Simple MultiParty Sessions:

a different perspective on multiparty session types:

Franco Barbanera

DMI - University of Catania

OVERVIEW

OVERVIEW

- Choreographic formalisms: the big picture;
-
-
-
-

OVERVIEW

- Choreographic formalisms: the big picture;
- Focusing on the Global-Local relation;
-
-
-

OVERVIEW

- Choreographic formalisms: the big picture;
- Focusing on the Global-Local relation;
- The SMPS Approach
  Since 2021, from an idea of F. Dagnino → Dagnino-Dezani-Giannini
  and then Barbanera, Bono, Castellani, de'Liguoro, Gheri, Lanese, Tuosto, Yoshida...
-
-

OVERVIEW

- Choreographic formalisms: the big picture;
- Focusing on the Global-Local relation;
- The SMPS Approach
  Since 2021, from an idea of F. Dagnino → Dagnino-Dezani-Giannini
  and then Barbanera, Bono, Castellani, de'Liguoro, Gheri, Lanese, Tuosto, Yoshida...
- SMPS and MPST
-

OVERVIEW

- Choreographic formalisms: the big picture;
- Focusing on the Global-Local relation;
- The SMPS Approach
  Since 2021, from an idea of F. Dagnino → Dagnino-Dezani-Giannini
  and then Barbanera, Bono, Castellani, de'Liguoro, Gheri, Lanese, Tuosto, Yoshida...
- SMPS and MPST
- Where we got, where we go

Description/Verification of concurrent systems

Global description

faithful/property-preserving

Implementation

Description/Verification of concurrent systems

Global description

Local (single component) behaviours

Description/Verification of concurrent systems

Global description

Local behaviours

Local behaviours

Description/Verification of concurrent systems

Global description

Local behaviours

Local behaviours

Local behaviours

Description/Verification of concurrent systems

Global description

Local behaviours

Local behaviours

Local behaviours

Description/Verification of concurrent systems

Global description

Local behaviours

abstraction step

Local behaviours

abstraction step

Local behaviours

Description/Verification of concurrent systems

Global description

dimension step

Local behaviours

abstraction step

Local behaviours

abstraction step

Local behaviours

Description/Verification of concurrent systems

Global description

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$ (A "Global-Local" relation)

Local behaviours

abstraction step

Local behaviours

abstraction step

Local behaviours

What expecting from $\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$?

Global description

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours

What expecting from $\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$?

Global description

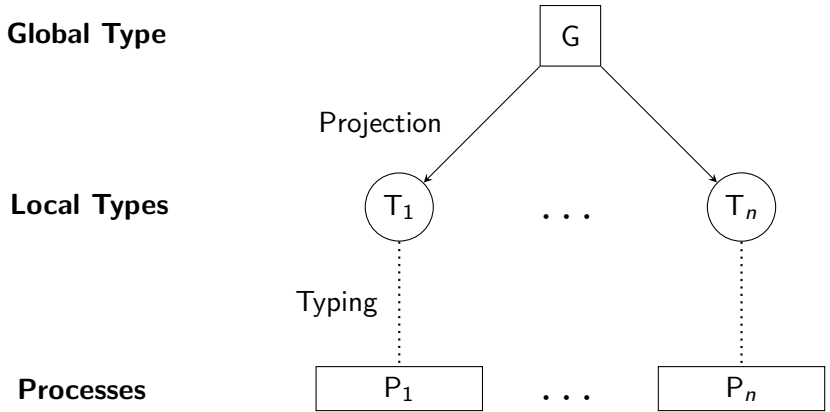
$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$ such that

- it faithfully represents the overall collective comportment of local behaviours
- its properties entail the collective properties of local behaviours.

MultiParty Session Types [Honda-Yoshida-Carbone 2008]



An example of the three layers for a protocol

The Shopaholic protocol (a.k.a. buyer-seller-carrier)

A buyer can keep on adding goods - sold by a seller - in his shopping cart an unbounded number of times, until he decides to buy the shopping cart's content. In the latter case, the seller informs the carrier for the shipment by sending him the list of goods.

The Shopaholic protocol (a.k.a. buyer-seller-carrier)

$$\mathsf{G} = \boldsymbol{\mu}\mathbf{t}.\ \mathsf{b} \to \mathsf{s} : \{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle).\mathsf{s} \to \mathsf{c} : \mathsf{ship}([\mathit{String}])\}$$

N. Yoshida, L. Gheri: *A Very Gentle Introduction to Multiparty Session Types.*

The Shopaholic protocol (a.k.a. buyer-seller-carrier)

$$\mathsf{G} = \boldsymbol{\mu}\mathbf{t}.\ \mathsf{b} \to \mathsf{s} : \{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle).\mathsf{s} \to \mathsf{c} : \mathsf{ship}([\mathit{String}])\}$$

$$\mathsf{T}_\mathsf{b} = \boldsymbol{\mu}\mathbf{t}.\ \oplus \mathsf{s}!\{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle)\} \qquad \mathsf{T}_\mathsf{s} = \boldsymbol{\mu}\mathbf{t}.\ \&\mathsf{b}?\{\ldots\} \qquad \mathsf{T}_\mathsf{c} = \ldots$$

The Shopaholic protocol (a.k.a. buyer-seller-carrier)

$$G = \boldsymbol{\mu t.}\ b \to s : \{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle).s \to c : \mathsf{ship}([\mathit{String}])\}$$

$$T_b = \boldsymbol{\mu t.}\ \oplus s!\{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle)\} \qquad T_s = \boldsymbol{\mu t.}\ \&b?\{\ldots\} \qquad T_c = \ldots$$

$$P_b = \boldsymbol{\mu X.}\ \underline{\text{if}}\ \mathit{Again?}\ \underline{\text{then}}\ s!\mathsf{add}\langle \mathit{pickone}\rangle.\mathbf{X}\ \underline{\text{else}}\ s!\mathsf{buy}\langle\,\rangle \qquad P_s = \ldots \qquad P_c = \ldots$$

Focusing on the Global-Local relation

$$\mathsf{G} = \boldsymbol{\mu}\mathbf{t}.\ \mathsf{b} \to \mathsf{s} : \{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle).\mathsf{s} \to \mathsf{c} : \mathsf{ship}([\mathit{String}])\}$$

Global description

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours

$$\mathsf{T}_\mathsf{b} = \boldsymbol{\mu}\mathbf{t}.\ \oplus \mathsf{s}!\{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \mathsf{buy}{:}(\langle\,\rangle)\} \qquad \mathsf{T}_\mathsf{s} = \boldsymbol{\mu}\mathbf{t}.\ \&\mathsf{b}?\{\ldots\} \qquad \mathsf{T}_\mathsf{c} = \ldots$$

$$\mathsf{P}_\mathsf{b} = \boldsymbol{\mu}\mathbf{t}.\ \underline{\text{if}}\ \mathit{Again?}\ \underline{\text{then}}\ \mathsf{s}!\mathsf{add}\langle \mathit{pickone}\rangle.\mathbf{t}\ \underline{\text{else}}\ \mathsf{s}!\mathsf{buy}\langle\,\rangle \qquad \mathsf{P}_\mathsf{s} = \ldots \qquad \mathsf{P}_\mathsf{c} = \ldots$$

Consider only one layer of local view

$$\mathsf{G} = \boldsymbol{\mu}\mathbf{t}.\ \mathsf{b} \to \mathsf{s} : \{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \ \mathsf{buy}{:}(\langle\,\rangle).\mathsf{s} \to \mathsf{c} : \mathsf{ship}([\mathit{String}])\}$$

Global description

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours

$$\mathsf{T}_\mathsf{b} = \boldsymbol{\mu}\mathbf{t}.\ \oplus\mathsf{s}!\{\mathsf{add}{:}(\mathit{String}).\mathbf{t},\ \mathsf{buy}{:}(\langle\,\rangle)\} \qquad \mathsf{T}_\mathsf{s} = \boldsymbol{\mu}\mathbf{t}.\ \&\mathsf{b}?\{\ldots\} \qquad \mathsf{T}_\mathsf{c} = \ldots$$

Disregarding values carried by messages

$$\mathsf{G} = \boldsymbol{\mu}\mathbf{t}.\ \mathsf{b} \to \mathsf{s} : \{\mathsf{add}.\mathbf{t},\ \mathsf{buy}.\ \mathsf{s} \to \mathsf{c} : \mathsf{ship}\}$$

Global description

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours

$$\mathsf{T}_\mathsf{b} = \boldsymbol{\mu}\mathbf{t}.\ \oplus \mathsf{s}!\{\mathsf{add}.\mathbf{t},\ \mathsf{buy}\} \qquad \mathsf{T}_\mathsf{s} = \boldsymbol{\mu}\mathbf{t}.\ \&\mathsf{b}?\{\ldots\} \qquad \mathsf{T}_\mathsf{c} = \ldots$$

Local Behaviours as abstractions for Processes

$$G = \mu \mathbf{t}.\ b \to s : \{\mathsf{add}.\mathbf{t},\ \mathsf{buy}.\ s \to c : \mathsf{ship}\}$$

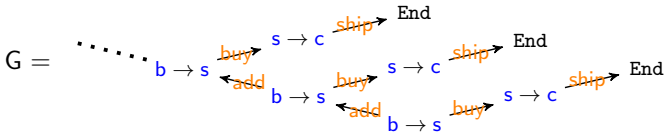
Global description

$\mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours

$P_b = \mu \mathbf{t}.\ \oplus s!\{\mathsf{add}.\mathbf{t},\ \mathsf{buy}\}$ $\quad$ $P_s = \mu \mathbf{t}.\ \&b?\{\ldots\}$ $\quad$ $P_c = \ldots$

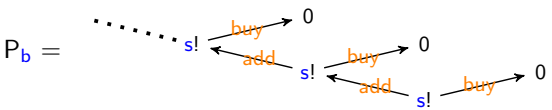
Disregarding actual representation of infinite structures



Global description

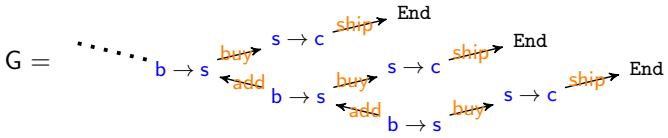
$\updownarrow \quad \mathcal{R}_{\mathcal{G}\text{-}\mathcal{L}}$

Local behaviours



$P_s = \ldots$ $P_c = \ldots$

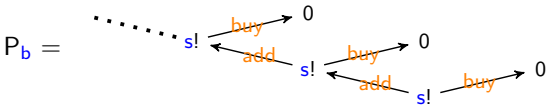
A different perspective on the Global-Local relation



Global description

Typing

Local behaviours



$P_s = \ldots$ $P_c = \ldots$

SMPS: Simple MultiParty Sessions

-
-
-
-
-
-

SMPS: Simple MultiParty Sessions

- Global descriptions $\rightsquigarrow$ Global types as in MPST
  but - no types of values carried by messages
  - (possibly) infinite terms.
-
-
-
-
-

SMPS: Simple MultiParty Sessions

- Global descriptions $\rightsquigarrow$ Global types as in MPST
  but - no types of values carried by messages
  - (possibly) infinite terms.
- Local behaviours $\rightsquigarrow$ Abstract processes (akin to MPST local types)
-
-
-
-

SMPS: Simple MultiParty Sessions

- Global descriptions $\rightsquigarrow$ Global types as in MPST
  but - no types of values carried by messages
  - (possibly) infinite terms.
- Local behaviours $\rightsquigarrow$ Abstract processes (akin to MPST local types)
- Global-Local relation $\rightsquigarrow$ Typing
-
-
-

SMPS: Simple MultiParty Sessions

- Global descriptions $\rightsquigarrow$ Global types as in MPST
  but - no types of values carried by messages
  - (possibly) infinite terms.
- Local behaviours $\rightsquigarrow$ Abstract processes (akin to MPST local types)
- Global-Local relation $\rightsquigarrow$ Typing
- Single channel, i.e. No "standard" delegation (internal delegation instead)
-
-

SMPS: Simple MultiParty Sessions

- Global descriptions $\rightsquigarrow$ Global types as in MPST
  but - no types of values carried by messages
  - (possibly) infinite terms.
- Local behaviours $\rightsquigarrow$ Abstract processes (akin to MPST local types)
- Global-Local relation $\rightsquigarrow$ Typing
- Single channel, i.e. No "standard" delegation (internal delegation instead)
- Communication model $\rightsquigarrow$ Synchronous or Asychronous
-

SMPS: Simple MultiParty Sessions

- Global descriptions $\rightsquigarrow$ Global types as in MPST
  but - no types of values carried by messages
  - (possibly) infinite terms.
- Local behaviours $\rightsquigarrow$ Abstract processes (akin to MPST local types)
- Global-Local relation $\rightsquigarrow$ Typing
- Single channel, i.e. No "standard" delegation (internal delegation instead)
- Communication model $\rightsquigarrow$ Synchronous or Asychronous
- Semantics of Global Types via a coinductive LTS

Processes and Multiparty Sessions

Processes

$$\mathsf{P} ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.\mathsf{P}_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.\mathsf{P}_i\}_{i\in I}$$

REGULAR terms only.

Processes and Multiparty Sessions

Processes

$$\mathsf{P} ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.\mathsf{P}_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.\mathsf{P}_i\}_{i\in I}$$

REGULAR terms only. (possible decidable and non-regular extensions)

Processes and Multiparty Sessions

Processes

$$\mathsf{P} ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.\mathsf{P}_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.\mathsf{P}_i\}_{i\in I}$$

REGULAR terms only.
Behaviours of components (participants) of systems.

Processes and Multiparty Sessions

Processes

$$\mathsf{P} ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.\mathsf{P}_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.\mathsf{P}_i\}_{i\in I}$$

REGULAR terms only.
Behaviours of components (participants) of systems.

Multiparty Sessions: parallel compositions of named processes.

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

Processes and Multiparty Sessions

Processes

$$\mathsf{P} ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.\mathsf{P}_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.\mathsf{P}_i\}_{i\in I}$$

REGULAR terms only.
Behaviours of components (participants) of systems.

Multiparty Sessions: parallel compositions of named processes.

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

Structural congruence: $\parallel$ is commutative, associative and with unit $\mathsf{p}[\mathbf{0}]$ (for any p)

Processes and Multiparty Sessions

Processes

$$\mathrm{P} ::=^{coind} \mathbf{0} \mid \mathtt{p}!\{\lambda_i.\mathrm{P}_i\}_{i\in I} \mid \mathtt{p}?\{\lambda_i.\mathrm{P}_i\}_{i\in I}$$

REGULAR terms only.
Behaviours of components (participants) of systems.

Multiparty Sessions: parallel compositions of named processes.

$$\mathbb{M} = \mathtt{p}_1[P_1] \parallel \cdots \parallel \mathtt{p}_n[P_n]$$

Synchronous Operational Semantics

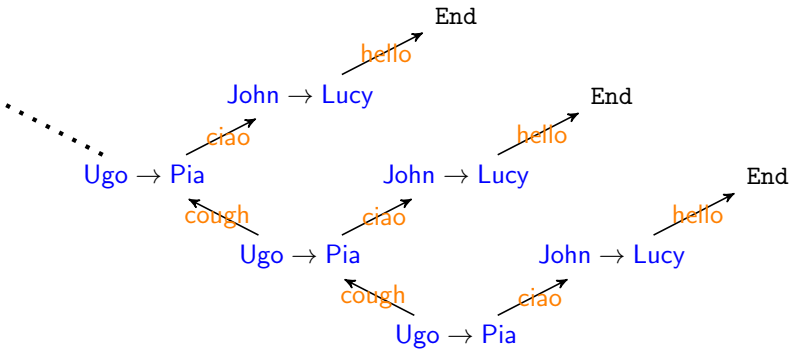
$$[\text{S-Comm}] \quad \frac{k \in I \subseteq J}{\mathtt{p}[\mathtt{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathtt{q}[\mathtt{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathtt{p}\lambda_k\mathtt{q}} \mathtt{p}[P_k] \parallel \mathtt{q}[Q_k] \parallel \mathbb{M}}$$

Global Types

$$\mathsf{G} ::=^{coind} \texttt{End} \;\mid\; \mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I}$$

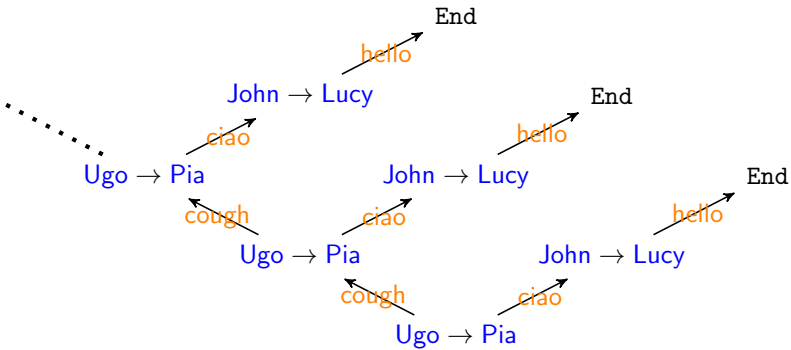
REGULAR terms only.

Global Types



Global Types

$$\mathsf{G} = \mathsf{Ugo} \to \mathsf{Pia} : \begin{cases} \mathsf{cough}.\mathsf{G} \\ \mathsf{ciao}.\ \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\ \mathtt{End} \end{cases}$$

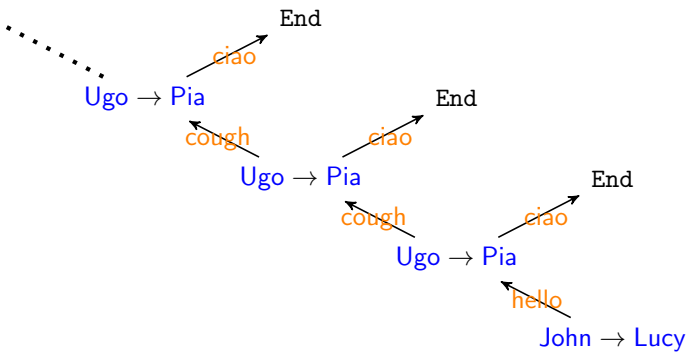


Global Types

$$\mathsf{G} = \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{G}' \qquad \mathsf{G}' = \mathsf{Ugo} \to \mathsf{Pia} : \begin{cases} \mathsf{cough}.\mathsf{G}' \\ \mathsf{ciao}.\ \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\ \mathtt{End} \end{cases}$$

Global Types

$$\mathsf{G} = \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{G}' \qquad \mathsf{G}' = \mathsf{Ugo} \to \mathsf{Pia} : \begin{cases} \mathsf{cough}.\mathsf{G}' \\ \mathsf{ciao}.\ \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\ \mathtt{End} \end{cases}$$



LTS for Global Types

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao}$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{John}\mathsf{hello}\mathsf{Lucy}}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{JohnhelloLucy}} \mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{JohnhelloLucy}} \mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{UgociaoPia}}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{JohnhelloLucy}} \mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{UgociaoPia}} \mathtt{End}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao}$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{UgociaoPia}}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{UgociaoPia}} \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{UgociaoPia}} \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello} \xrightarrow{\mathsf{JohnhelloLucy}}$$

LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j} \text{[External]}$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i \in I}} \text{[Internal]}$$

$$\mathsf{John} \to \mathsf{Lucy} : \mathsf{hello}.\mathsf{Ugo} \to \mathsf{Pia} : \mathsf{ciao} \xrightarrow{\mathsf{UgociaoPia}} \mathsf{John} \to \mathsf{Lucy} : \mathsf{hello} \xrightarrow{\mathsf{JohnhelloLucy}} \mathtt{End}$$

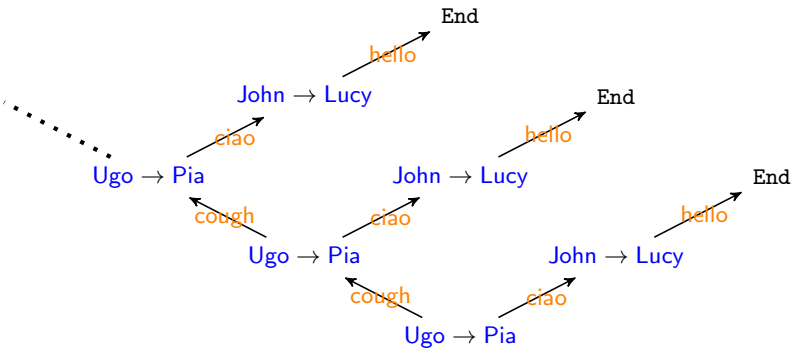
Drawback of inductively-defined LTS for Global Types

Drawback of inductively-defined LTS for Global Types

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

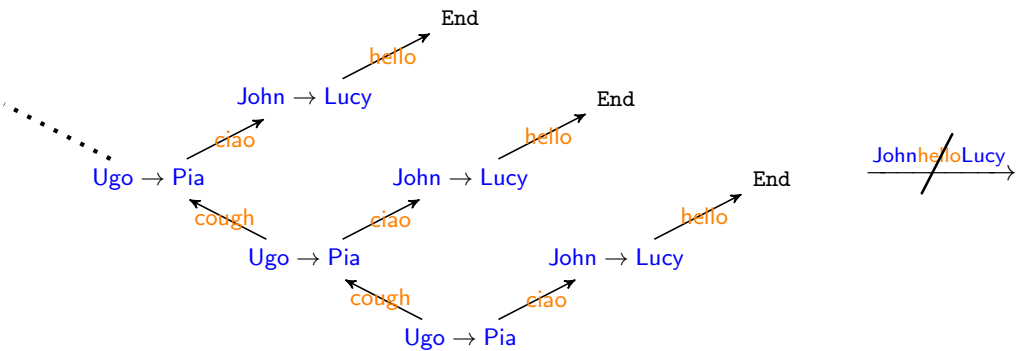
Drawback of inductively-defined LTS for Global Types

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}} \text{ [Internal]}$$



Drawback of inductively-defined LTS for Global Types

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}} \text{[Internal]}$$



A coinductive LTS for Global Types

A coinductive LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

A coinductive LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

A coinductive LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

$$\text{!!!}\quad \mathcal{D} = \frac{\mathcal{D}}{\mathsf{r} \to \mathsf{s}{:}\lambda.\mathsf{G} \xrightarrow{\mathsf{p}\lambda'\mathsf{q}} \mathsf{r} \to \mathsf{s}{:}\lambda.\mathsf{G}}\quad \text{!!!}$$

where $\mathsf{G} = \mathsf{r} \to \mathsf{s} : \lambda.\mathsf{G}$

A coinductive LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \qquad \mathsf{p}\lambda\mathsf{q} \in \bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

A coinductive LTS for Global Types

$$\frac{j \in I}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda_j\mathsf{q}} \mathsf{G}_j}\ [\textsc{External}]$$

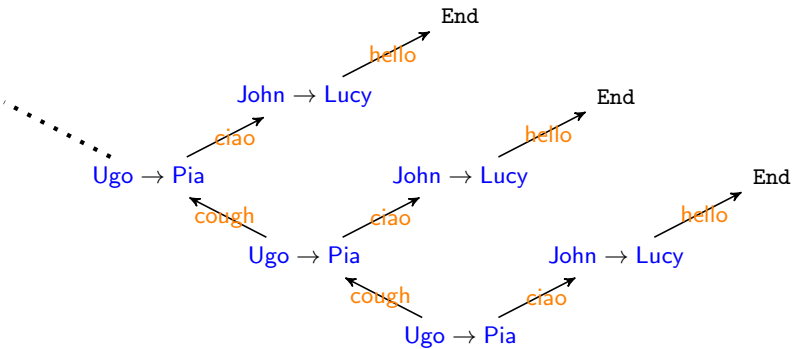
$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \qquad \mathsf{p}\lambda\mathsf{q} \in \bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}}\ [\textsc{Internal}]$$

where $\mathsf{cap}(\mathtt{End}) = \emptyset$

$\mathsf{cap}(\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I}) = \{\mathsf{p}\lambda_i\mathsf{q} \mid i \in I\} \cup \bigcup_{i\in I}\{\mathsf{cap}(\mathsf{G}_i)\}.$

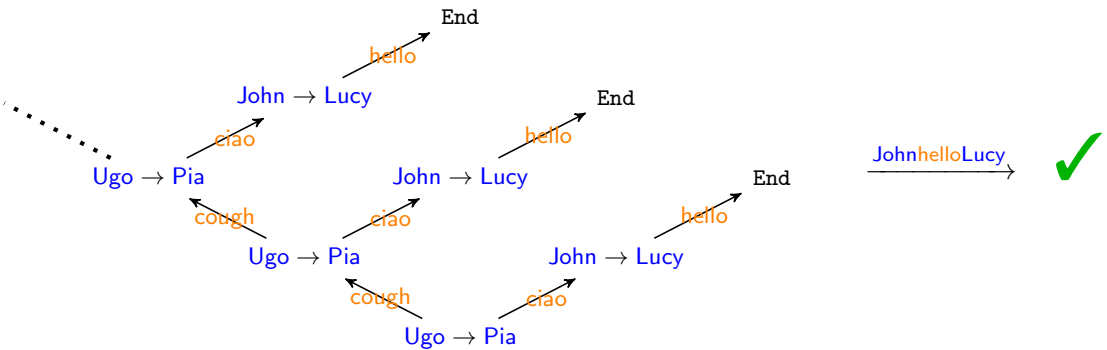
A coinductive LTS for Global Types

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \qquad \mathsf{p}\lambda\mathsf{q} \in \bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}} \text{ [Internal]}$$



A coinductive LTS for Global Types

$$\frac{\mathsf{G}_i \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}'_i \quad \forall i \in I \qquad \{\mathsf{p},\mathsf{q}\} \cap \{\mathsf{r},\mathsf{s}\} = \emptyset \qquad \mathsf{p}\lambda\mathsf{q} \in \bigcap_{i\in I} \mathsf{cap}(\mathsf{G}_i)}{\mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{r} \to \mathsf{s} : \{\lambda_i.\mathsf{G}'_i\}_{i\in I}} \text{ [INTERNAL]}$$



The SMPS Global-Local relation: Typing

Judgements of the form $\mathsf{G} \vdash \mathbb{M}$ are coinductively derived by the type system below, by considering sessions up to structural congruence:

$$\mathsf{End} \vdash \mathsf{p}[\mathbf{0}] \qquad [\text{T-End}]$$

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \qquad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j \in J}] \parallel \mathbb{M}} \; [\text{T-Comm}]$$

The SMPS type system

Judgements of the form $\mathsf{G} \vdash \mathbb{M}$ are coinductively derived by the type system below, by considering sessions up to structural congruence:

$$\mathsf{End} \vdash \mathsf{p}[\mathbf{0}] \qquad [\text{T-End}]$$

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \qquad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j\in J}] \parallel \mathbb{M}} \; [\text{T-Comm}]$$

more inputs in q allowed than branches in G
same outputs in p as branches in G (getting a strong version of Session Fidelity)

The SMPS type system

Judgements of the form $\mathsf{G} \vdash \mathbb{M}$ are coinductively derived by the type system below, by considering sessions up to structural congruence:

$$\mathsf{End} \vdash \mathsf{p}[\mathbf{0}] \qquad [\text{T-End}]$$

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \qquad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j \in J}] \parallel \mathbb{M}} \; [\text{T-Comm}]$$

The SMPS type system

Judgements of the form $\mathsf{G} \vdash \mathbb{M}$ are coinductively derived by the type system below, by considering sessions up to structural congruence:

$$\mathsf{End} \vdash \mathsf{p}[\mathbf{0}] \qquad [\text{T-End}]$$

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \qquad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j \in J}] \parallel \mathbb{M}} \; [\text{T-Comm}]$$

The SMPS type system

Judgements of the form $\mathsf{G} \vdash \mathbb{M}$ are coinductively derived by the type system below, by considering sessions up to structural congruence:

$$\mathsf{End} \vdash \mathsf{p}[\mathbf{0}] \qquad [\text{T-End}]$$

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad \mathsf{prt}(\mathsf{G}_i) \setminus \{\mathsf{p},\mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \qquad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j\in J}] \parallel \mathbb{M}} \; [\text{T-Comm}]$$

Otherwise $\mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathsf{r}[\mathsf{s}?\lambda']$ TYPABLE with $\mathsf{G} = \mathsf{q} \to \mathsf{p}{:}\lambda.\mathsf{G}$
where $P = \mathsf{q}?\lambda.P$ and $Q = \mathsf{p}!\lambda.Q$

Typing the shopaholic system

$$\mathsf{G} \vdash \mathsf{b}[B] \parallel \mathsf{s}[S] \parallel \mathsf{c}[C]$$

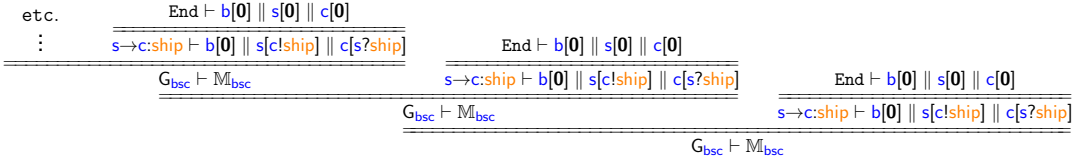
where

$\mathsf{G} = \mathsf{b} \to \mathsf{s} : \{\mathsf{add}.\mathsf{G},\ \mathsf{buy}.\mathsf{s} \to \mathsf{c} : \mathsf{ship}\}$

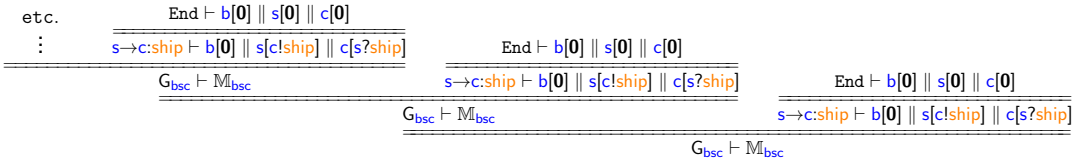
and

$B = \mathsf{s}!\{\mathsf{add}.B,\ \mathsf{buy}\} \qquad S = \mathsf{b}?\{\mathsf{add}.S,\ \mathsf{buy}.\mathsf{c}!\mathsf{ship}\} \qquad C = \mathsf{s}?\mathsf{ship}$

Typing the shopaholic system



Typing the shopaholic system, and more



The "benchmark" examples of Less-is-More [Scalas-Yoshida] are also typable

- OAuth2 Fragment
- Recursive Two-Buyers
- Recursive Map/Reduce
- Independent Multiparty Workers

Theorem: Session Fidelity and Subject Reduction

If

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

then

$$\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}' \implies \mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}' \quad \mathsf{G}' \vdash \mathbb{M}'$$

$$\mathsf{G} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathsf{G}' \quad \mathsf{G}' \vdash \mathbb{M}' \impliedby \mathbb{M} \xrightarrow{\mathsf{p}\lambda\mathsf{q}} \mathbb{M}'$$

If $\mathsf{G} \vdash \mathbb{M}$ then G and $\mathbb{M}$ are bisimilar (they have the same computational content).
If both G and G' type the same $\mathbb{M}$, then they are bisimilar.

Theorem: Typability implies Lock-freedom

If

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

then

$$\mathbb{M} \text{ is lock-free}$$

A session $\mathbb{M}$ is lock free if

$$\mathbb{M} \xrightarrow{\sigma} \mathbb{M}' \text{ and } \mathsf{p} \in \mathsf{prt}(\mathbb{M}')$$

$$\Downarrow$$

$$\mathbb{M}' \xrightarrow{\sigma' \cdot \Lambda} \text{ for some } \sigma' \text{ and } \Lambda \text{ such that } \mathsf{p} \notin \mathsf{prt}(\sigma') \text{ and } \mathsf{p} \in \mathsf{prt}(\Lambda).$$

Typability is independent from the interaction taken into account

$$\dfrac{\dfrac{\mathcal{D}_i}{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \parallel \mathsf{r}[\mathsf{s}!\{\lambda_h.R_h\}_{i\in H}] \parallel \mathsf{s}[\mathsf{r}?\{\lambda_k.S_k\}_{k\in K}]}\ \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j\in J}] \parallel \mathbb{M} \parallel \mathsf{r}[\mathsf{s}!\{\lambda'_h.R_h\}_{i\in H}] \parallel \mathsf{s}[\mathsf{r}?\{\lambda'_k.S_k\}_{k\in K}]}$$

$$\Downarrow$$

$$\dfrac{\dfrac{\mathcal{D}'_h}{\mathsf{G}'_i \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j\in J}] \parallel \mathbb{M} \parallel \mathsf{r}[R_h] \parallel \mathsf{s}[S_h]}\ \forall h \in H \subseteq K}{\mathsf{r} \to \mathsf{s} : \{\lambda'_h.\mathsf{G}'_h\}_{h\in H} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.P_j\}_{j\in J}] \parallel \mathbb{M} \parallel \mathsf{r}[\mathsf{s}!\{\lambda'_h.R_h\}_{i\in H}] \parallel \mathsf{s}[\mathsf{r}?\{\lambda'_k.S_k\}_{k\in K}]}$$

by Subject Reduction

Typability completely characterises Type Safety

- $\mathbb{M}$ **is stuck** if $\mathbb{M} \not\equiv \mathsf{p}[\mathbf{0}]$ and $\mathbb{M} \not\rightarrow$
- $\mathbb{M}$ **gets stuck** ($\mathtt{stuck}(\mathbb{M})$) if for no stuck $\mathbb{M}'$: $\mathbb{M} \rightarrow^* \mathbb{M}'$

Typability completely characterises Type Safety

- $\mathbb{M}$ **is stuck** if $\mathbb{M} \not\equiv \mathsf{p}[\mathbf{0}]$ and $\mathbb{M} \not\rightarrow$
- $\mathbb{M}$ **gets stuck** ($\mathtt{stuck}(\mathbb{M})$) if for no stuck $\mathbb{M}'$: $\mathbb{M} \rightarrow^* \mathbb{M}'$

$$\mathbb{M} \text{ typable} \quad \Leftrightarrow \quad \neg\mathtt{stuck}(\mathbb{M})$$

Typability fully characterises Type Safety

$$\mathbb{M} \text{ typable} \quad \Leftrightarrow \quad \neg \texttt{stuck}(\mathbb{M})$$

Typability fully characterises Type Safety

$$\mathbb{M} \text{ typable} \quad \Leftrightarrow \quad \neg \texttt{stuck}(\mathbb{M})$$

$\Rightarrow$): by Lock Freedom

$\Leftarrow$): 1. Any reduction strategy is mimicked by a type reconstruction;

2. Any type reconstruction of an untypable session fails in a point where we try to type a stuck session.

MPST $\leftarrow ? \rightarrow$ SMPS

MPST $\leftarrow ? \rightarrow$ SMPS

$$\frac{}{\mathsf{G}\!\upharpoonright\!\mathsf{p} = \mathbf{0}}\ \mathsf{p} \notin \mathsf{G}$$

$$\frac{\mathsf{G}_i\!\upharpoonright\!\mathsf{p} = P_i \quad \forall i \in I}{(\mathsf{p}\to\mathsf{s}{:}\{\lambda_i.\mathsf{G}_i\}_{i\in I})\!\upharpoonright\!\mathsf{p} = \mathsf{s}!\{\lambda_i.P_i\}_{i\in I}} \qquad \frac{\mathsf{G}_i\!\upharpoonright\!\mathsf{p} = P_i \quad \forall i \in I}{(\mathsf{r}\to\mathsf{p}{:}\{\lambda_i.\mathsf{G}_i\}_{i\in I})\!\upharpoonright\!\mathsf{p} = \mathsf{r}?\{\lambda_i.P_i\}_{i\in I}}$$

$$\frac{\mathsf{G}_i\!\upharpoonright\!\mathsf{p} = P_i \quad \forall i \in I \qquad \bigsqcap_{i\in I} P_i = P}{(\mathsf{r}\to\mathsf{s}{:}\{\lambda_i.\mathsf{G}_i\}_{i\in I})\!\upharpoonright\!\mathsf{p} = P}\ (\mathsf{p} \in \mathsf{G} \text{ and } \mathsf{p} \notin \{\mathsf{r},\mathsf{s}\})$$

where the (full) merge $P \sqcap Q$ is the partial operation coinductively defined by

$$\frac{}{\mathbf{0} \sqcap \mathbf{0} = \mathbf{0}} \qquad \frac{P_i \sqcap Q_i = R_i \quad \forall i \in I}{\mathsf{q}!\{\lambda_i.P_i\}_{i\in I} \sqcap \mathsf{q}!\{\lambda_i.Q_i\}_{i\in I} = \mathsf{q}!\{\lambda_i.R_i\}_{i\in I}}$$

$$\frac{P_h \sqcap Q_h = R_h \quad \forall h \in I \cap J}{\mathsf{q}?\{\lambda_i.P_i\}_{i\in I} \sqcap \mathsf{q}?\{\lambda_i.Q_j\}_{i\in J} = \mathsf{q}?\{\lambda_i.P_i, \lambda_j.Q_j, \lambda_h.R_h \mid i \in I \setminus J, j \in J \setminus I, h \in I \cap J\}}$$

MPST $\leftarrow ? \rightarrow$ SMPS

$$\mathcal{D} \;=\; \dfrac{\dfrac{\mathcal{D}}{\mathsf{G}_{\mathsf{bsc}}\upharpoonright \mathsf{c} = \mathsf{s}?\mathsf{ship}} \qquad \dfrac{}{\mathsf{End}\upharpoonright \mathsf{c} = \mathbf{0}} \qquad \dfrac{}{\mathbf{0} \sqcap \mathbf{0} = \mathbf{0}}}{\mathsf{G}_{\mathsf{bsc}}\upharpoonright \mathsf{c} = \mathsf{s}?\mathsf{ship}}$$

and moreover

$$\mathsf{G}_{\mathsf{bsc}} \vdash \mathsf{G}_{\mathsf{bsc}}\upharpoonright \mathsf{b} \parallel \mathsf{G}_{\mathsf{bsc}}\upharpoonright \mathsf{s} \parallel \mathsf{G}_{\mathsf{bsc}}\upharpoonright \mathsf{c} \equiv \mathbb{M}_{\mathsf{bsc}}$$

MPST $\leftarrow ? \rightarrow$ SMPS

The preorder $\leqslant$ on processes is coinductively defined by

$$[\text{SUB-}\mathbf{0}]\ \frac{}{\mathbf{0} \leqslant \mathbf{0}} \qquad [\text{SUB-OUT}]\ \frac{P_i \leqslant Q_i \qquad \forall i \in I}{\mathsf{q}!\{\lambda_i.P_i\}_{i\in I} \leqslant \mathsf{q}!\{\lambda_i.Q_i\}_{i\in I}}$$

$$[\text{SUB-IN}]\ \frac{P_i \leqslant Q_i \qquad \forall i \in I}{\mathsf{q}?\{\lambda_i.P_i\}_{i\in I\cup J} \leqslant \mathsf{q}?\{\lambda_i.Q_i\}_{i\in I}}$$

MPST $\leftarrow ? \rightarrow$ SMPS

Define

$$\mathbb{M} \leqslant \mathbb{M}' \quad \text{if} \quad \mathsf{prt}(\mathbb{M}) = \mathsf{prt}(\mathbb{M}') \text{ and } \forall \mathsf{p}.\ \mathsf{p}[P] \in \mathbb{M} \text{ and } \mathsf{p}[P'] \in \mathbb{M}' \text{ imply } P \leqslant P'$$

and

$$\mathsf{G} \triangleright \mathbb{M} \quad \text{if} \quad \mathsf{G}\!\downarrow \text{ is defined and } \mathbb{M} \leqslant \mathsf{G}\!\downarrow$$

MPST $\leftarrow ? \rightarrow$ SMPS

Define

$$\mathbb{M} \leqslant \mathbb{M}' \quad \text{if} \quad \mathsf{prt}(\mathbb{M}) = \mathsf{prt}(\mathbb{M}') \text{ and } \forall \mathsf{p}.\ \mathsf{p}[P] \in \mathbb{M} \text{ and } \mathsf{p}[P'] \in \mathbb{M}' \text{ imply } P \leqslant P'$$

and

$$\mathsf{G} \triangleright \mathbb{M} \quad \text{if} \quad \mathsf{G}\downarrow \text{ is defined and } \mathbb{M} \leqslant \mathsf{G}\downarrow$$

a simplified version of *association* (Hou-Yoshida-Kuhn: **Less is more - rivisited**)

MPST $\leftarrow ? \rightarrow$ SMPS

Define

$$\mathbb{M} \preccurlyeq \mathbb{M}' \quad \text{if} \quad \mathsf{prt}(\mathbb{M}) = \mathsf{prt}(\mathbb{M}') \text{ and } \forall \mathsf{p}.\ \mathsf{p}[P] \in \mathbb{M} \text{ and } \mathsf{p}[P'] \in \mathbb{M}' \text{ imply } P \preccurlyeq P'$$

and

$$\mathsf{G} \triangleright \mathbb{M} \quad \text{if} \quad \mathsf{G}\!\downarrow \text{ is defined and } \mathbb{M} \preccurlyeq \mathsf{G}\!\downarrow$$

Equivalence between typability and projectability:

$$\mathsf{G} \vdash \mathbb{M} \quad \Leftrightarrow \quad \mathsf{G} \triangleright \mathbb{M}$$

MPST $\leftarrow ? \rightarrow$ SMPS

Define

$$\mathbb{M} \preccurlyeq \mathbb{M}' \quad \text{if} \quad \mathsf{prt}(\mathbb{M}) = \mathsf{prt}(\mathbb{M}') \text{ and } \forall \mathsf{p}.\ \mathsf{p}[P] \in \mathbb{M} \text{ and } \mathsf{p}[P'] \in \mathbb{M}' \text{ imply } P \preccurlyeq P'$$

and

$$\mathsf{G} \triangleright \mathbb{M} \quad \text{if} \quad \mathsf{G}\!\downarrow \text{ is defined and } \mathbb{M} \preccurlyeq \mathsf{G}\!\downarrow$$

Equivalence between typability and projectability:

$$\mathsf{G} \vdash \mathbb{M} \quad \Leftrightarrow \quad \mathsf{G} \triangleright \mathbb{M}$$

Hence Type Safety is fully characterised by the existence of a projectable global type

Adding layers to the local view.

$\mathtt{p}!\{\lambda_i.\mathrm{P}_i\}_{i\in I}$ is an internal choice i.e. abstraction for conditionals.

Concrete Processes

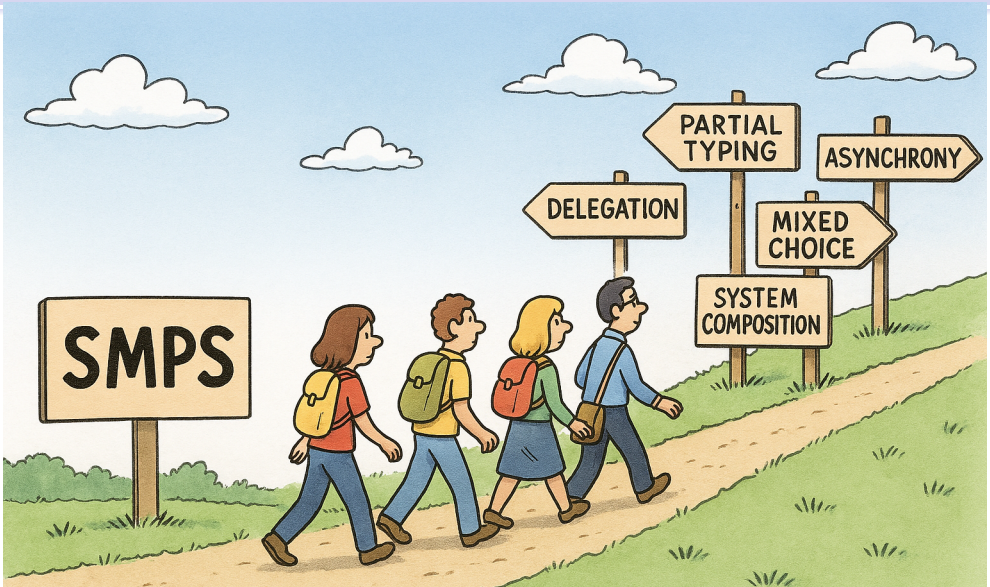
$$
\begin{array}{rcl}
\mathrm{P} & ::=^{coind} & \mathbf{0} \\
& | & \mathtt{p}!\lambda\langle e\rangle.\mathrm{P} \\
& | & \mathtt{p}?\{\lambda_i(x_i).\mathrm{P}_i\}_{i\in I} \\
& | & \texttt{case } e \texttt{ of } v_1\texttt{->}\mathtt{p}!\lambda_1\langle e_1\rangle.\mathrm{P}_1; \ldots, v_n\texttt{->}\mathtt{p}!\lambda_n\langle e_n\rangle.\mathrm{P}_n \texttt{ other } \mathtt{p}!\lambda_{n+1}\langle e_{n+1}\rangle.\mathrm{P}_{n+1}
\end{array}
$$

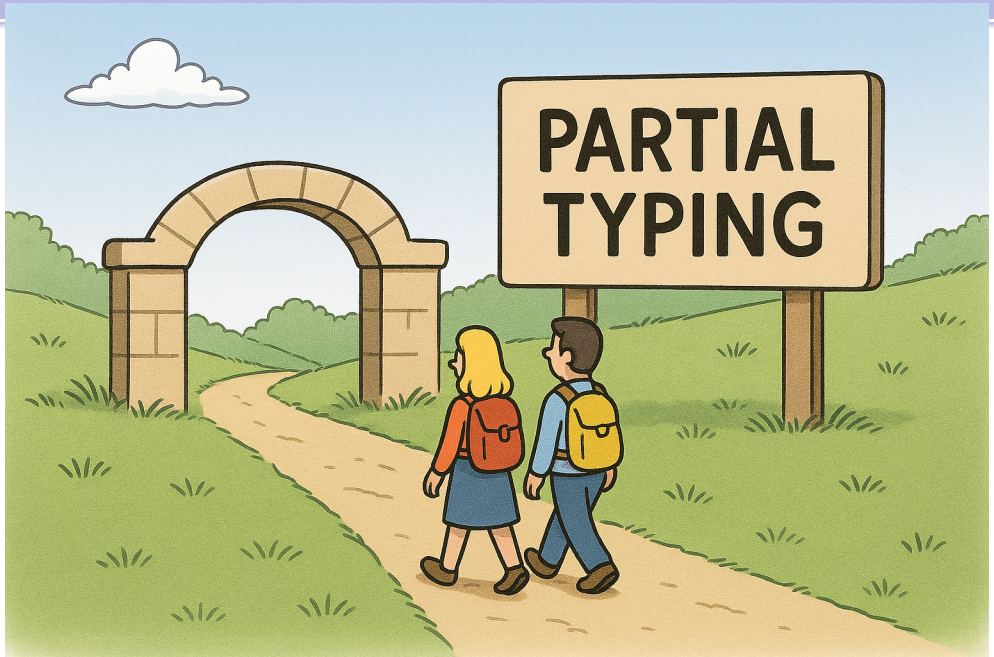
Multi-layered local view - as in MPST
Multiparty Sessions as MPST (simplified) Typing Contexts.
A type system for

$$\mathtt{p}_1[P_1] \parallel \cdots \parallel \mathtt{p}_n[P_n] \vdash \mathtt{p}_1[\mathrm{P}_1] \parallel \cdots \parallel \mathtt{p}_n[\mathrm{P}_n]$$

where communication properties are preserved.
F.Barbanera, U.de'Liguoro: *Multiparty Sessions Made Simple*, Springer BehAPI

A starting point




PARTIAL
TYPING

Definition (p-Lock)

$\mathbb{M}'$ is a p-lock if the participant p is willing to progress in $\mathbb{M}'$ but cannot do that in any continuation of $\mathbb{M}'$.

Definition (Lock-freedom)

$\mathbb{M}$ is lock-free if, for each participant p,

$$\mathbb{M} \rightarrow^* \mathbb{M}' \text{ implies } \mathbb{M}' \text{ is not a p-lock}$$

Isn't that too much in "real life"?

If

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

then

$\mathbb{M}$ is lock-free

Universal Declaration of system-Components' Rights

- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]

Universal Declaration of system-Components' Rights

- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]

Universal Declaration of system-Components' Rights

- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]

Universal Declaration of system-Components' Rights

- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]

Universal Declaration of system-Components' Rights

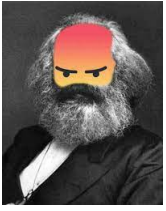
- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]

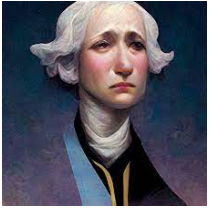
Universal Declaration of system-Components' Rights

- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without no distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]

Universal Declaration of system-Components' Rights

- **Article 1** All system components must be developed equal in dignity and rights. They are endowed with communication capabilities and should interact towards one another in a spirit of cooperation.
- **Article 2** Any system component is entitled to all the good communication properties like lock freedom, without no distinction of any kind, such as programming paradigm, language or interaction model.
- **Article 3** [...]





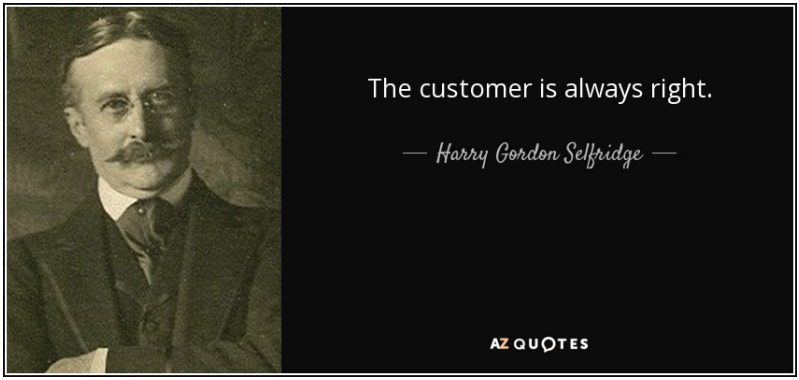
Non egalitarian systems

Non egalitarian systems

Any client-server setting is biased:

Non egalitarian systems

Any client-server setting is biased:



A non egalitarian system

A buyer can keep on adding goods - sold by a seller - in his shopping cart an unbounded number of times, until he decides to buy the shopping cart's content. In the latter case, the seller informs the carrier for the shipment, **where the carrier is able to serve an unbounded number of shipments.**

A non egalitarian system

A buyer can keep on adding goods - sold by a seller - in his shopping cart an unbounded number of times, until he decides to buy the shopping cart's content. In the latter case, the seller informs the carrier for the shipment, **where the carrier is able to serve an unbounded number of shipments.**

$$\mathbb{M} = \mathsf{b}[B] \parallel \mathsf{s}[S] \parallel \mathsf{c}[C]$$

with

$B = \mathsf{s}!\{\mathsf{add}.B,\ \mathsf{buy}\}$

$S = \mathsf{b}?\{\mathsf{add}.S,\ \mathsf{buy}.\mathsf{c}!\mathsf{ship}\}$

$C = \mathsf{s}?\mathsf{ship}.C$

A non egalitarian system

A buyer can keep on adding goods - sold by a seller - in his shopping cart an unbounded number of times, until he decides to buy the shopping cart's content. In the latter case, the seller informs the carrier for the shipment, **where the carrier is able to serve an unbounded number of shipments.**

$$\mathbb{M} = \mathsf{b}[B] \parallel \mathsf{s}[S] \parallel \mathsf{c}[C]$$

with
$B = \mathsf{s}!\{\mathsf{add}.B, \mathsf{buy}\}$
$S = \mathsf{b}?\{\mathsf{add}.S, \mathsf{buy}.\mathsf{c}!\mathsf{ship}\}$
$C = \mathsf{s}?\mathsf{ship}.C$

Not typable. In fact it is not c-lock free

$$\mathbb{M} \rightarrow^* \mathsf{b}[\mathbf{0}] \parallel \mathsf{s}[\mathbf{0}] \parallel \mathsf{c}[C]$$

A non egalitarian system

A buyer can keep on adding goods - sold by a seller - in his shopping cart an unbounded number of times, until he decides to buy the shopping cart's content. In the latter case, the seller informs the carrier for the shipment, **where the carrier is able to serve an unbounded number of shipments.**

$$\mathbb{M} = \mathsf{b}[B] \parallel \mathsf{s}[S] \parallel \mathsf{c}[C]$$

with
$B = \mathsf{s}!\{\mathsf{add}.B, \mathsf{buy}\}$
$S = \mathsf{b}?\{\mathsf{add}.S, \mathsf{buy}.\mathsf{c}!\mathsf{ship}\}$
$C = \mathsf{s}?\mathsf{ship}.C$

Not typable. In fact it is not c-lock free

$$\mathbb{M} \rightarrow^* \mathsf{b}[\mathbf{0}] \parallel \mathsf{s}[\mathbf{0}] \parallel \mathsf{c}[C]$$

Do we actually care about s and c?

Typing non egalitarian systems

We are interested in b's lock-freedom, not s's and c's

Franco Barbanera, Mariangiola Dezani:
Partially Typed Multiparty Sessions. ICE 2023
Franco Barbanera, Viviana Bono, Mariangiola Dezani:
Open compliance in multiparty sessions with partial typing. JLAMP, 2025

A type system such that if

$$\mathsf{G} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathbb{M}$$

then lock-freedom ensured only for participants other than s and c.
For our example this is possible for

$$\mathsf{G} = \mathsf{b} \to \mathsf{s}{:}\{\mathsf{add}.\,\mathsf{G},\ \mathsf{buy}\}$$

Typing non egalitarian systems

We are interested in b's lock-freedom, not s's and c's

Franco Barbanera, Mariangiola Dezani:
Partially Typed Multiparty Sessions. ICE 2023
Franco Barbanera, Viviana Bono, Mariangiola Dezani:
Open compliance in multiparty sessions with partial typing. JLAMP, 2025

A type system such that if

$$\mathsf{G} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathbb{M}$$

then lock-freedom ensured only for participants other than s and c.
For our example this is possible for

$$\mathsf{G} = \mathsf{b} \to \mathsf{s}:\{\mathsf{add}.\,\mathsf{G},\ \mathsf{buy}\}$$

Typing non egalitarian systems

We are interested in b's lock-freedom, not s's and c's

Franco Barbanera, Mariangiola Dezani:
Partially Typed Multiparty Sessions. ICE 2023
Franco Barbanera, Viviana Bono, Mariangiola Dezani:
Open compliance in multiparty sessions with partial typing. JLAMP, 2025

A type system such that if

$$\mathsf{G} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathbb{M}$$

then lock-freedom ensured only for participants other than s and c.
For our example this is possible for

$$\mathsf{G} = \mathsf{b} \to \mathsf{s}{:}\{\mathsf{add}.\,\mathsf{G},\ \mathsf{buy}\}$$

Typing non egalitarian systems

$$[\text{End}] \quad \frac{}{\text{End} \vdash_{\emptyset} \mathsf{p}[\mathbf{0}]} \; [\text{End}]$$

$$\frac{\mathsf{G}_i \vdash_{\mathcal{P}_i} \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad (\mathsf{prt}(\mathsf{G}_i) \cup \mathcal{P}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \quad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \vdash_{\bigcup_{i \in I} \mathcal{P}_i} \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.Q_j\}_{j \in J}] \parallel \mathbb{M}}$$

$$[\text{Weak}] \quad \frac{\mathsf{G} \vdash_{\mathcal{P}_1} \mathbb{M}_1}{\mathsf{G} \vdash_{\mathcal{P}_1 \cup \mathcal{P}_2} \mathbb{M}_1 \parallel \mathbb{M}_2} \quad \mathcal{P}_2 = \mathsf{prt}(\mathbb{M}_2) \neq \emptyset$$

Typing non egalitarian systems

$$[\text{End}]\quad \frac{}{\mathtt{End} \vdash_{\emptyset} \mathsf{p}[\mathbf{0}]}\ [\text{End}]$$

$$\frac{\mathsf{G}_i \vdash_{\mathcal{P}_i} \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad (\mathsf{prt}(\mathsf{G}_i) \cup \mathcal{P}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \quad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash_{\bigcup_{i\in I}\mathcal{P}_i} \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M}}$$

$$[\textsc{Weak}]\quad \frac{\mathsf{G} \vdash_{\mathcal{P}_1} \mathbb{M}_1}{\mathsf{G} \vdash_{\mathcal{P}_1 \cup \mathcal{P}_2} \mathbb{M}_1 \parallel \mathbb{M}_2}\ \mathcal{P}_2 = \mathsf{prt}(\mathbb{M}_2) \neq \emptyset$$

Typing non egalitarian systems

$$[\text{End}] \quad \frac{}{\mathbf{End} \vdash_{\emptyset} \mathsf{p}[\mathbf{0}]} \; [\text{End}]$$

$$[\text{Weak}] \quad \frac{\mathsf{G} \vdash_{\mathcal{P}_1} \mathbb{M}_1}{\mathsf{G} \vdash_{\mathcal{P}_1 \cup \mathcal{P}_2} \mathbb{M}_1 \parallel \mathbb{M}_2} \; \mathcal{P}_2 = \mathsf{prt}(\mathbb{M}_2) \neq \emptyset$$

$$\frac{\mathsf{G}_i \vdash_{\mathcal{P}_i} \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M} \qquad (\mathsf{prt}(\mathsf{G}_i) \cup \mathcal{P}_i) \setminus \{\mathsf{p}, \mathsf{q}\} = \mathsf{prt}(\mathbb{M}) \quad \forall i \in I \subseteq J}{\mathsf{p} \to \mathsf{q} : \{\lambda_i.\mathsf{G}_i\}_{i \in I} \vdash_{\bigcup_{i \in I} \mathcal{P}_i} \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.Q_j\}_{j \in J}] \parallel \mathbb{M}}$$

Typing non egalitarian systems

$$[\text{End}] \quad \frac{}{\mathbf{End} \vdash_{\emptyset} \mathsf{p}[\mathbf{0}]}\,[\text{End}]$$

$$[\textsc{Weak}] \quad \frac{\mathsf{G} \vdash_{\mathcal{P}_1} \mathbb{M}_1}{\mathsf{G} \vdash_{\mathcal{P}_1 \cup \mathcal{P}_2} \mathbb{M}_1 \parallel \mathbb{M}_2} \quad \mathcal{P}_2 = \mathsf{prt}(\mathbb{M}_2) \neq \emptyset$$

Typing non egalitarian systems

Theorem (Classist lock-freedom)

If $\mathsf{G} \vdash_{\mathcal{P}} \mathbb{M}$ *and* $\mathsf{p} \notin \mathcal{P}$ *then* $\mathbb{M}$ *is* p*-lock free*

Typing non egalitarian systems

$$\mathcal{D} \quad = \quad \dfrac{\mathcal{D} \qquad \dfrac{\dfrac{}{\mathsf{End} \vdash_{\emptyset} \mathsf{b}[\mathbf{0}]}\,[\mathrm{End}]}{\mathsf{End} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathsf{b}[\mathbf{0}] \parallel \mathsf{s}[\mathsf{c}!\mathsf{ship}] \parallel \mathsf{c}[C]}\,[\mathrm{WEAK}]}{\mathsf{G} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathsf{b}[B] \parallel \mathsf{s}[\mathsf{b}?\{\mathsf{add}.S,\ \mathsf{pay}.\mathsf{c}!\mathsf{ship}\}] \parallel \mathsf{c}[\mathsf{s}?\mathsf{ship}.C]}$$

where $\mathsf{G} = \mathsf{b} \to \mathsf{s}{:}\{\mathsf{add}.\,\mathsf{G},\ \mathsf{buy}\}$

$B = \mathsf{s}!\{\mathsf{add}.B,\ \mathsf{buy}\}$
$S = \mathsf{b}?\{\mathsf{add}.S,\ \mathsf{buy}.\mathsf{c}!\mathsf{ship}\}$
$C = \mathsf{s}?\mathsf{ship}.C$

Hence the Buyer-Seller-Carrier system is b-lock free

Typing non egalitarian systems

$$\mathcal{D} \quad = \quad \dfrac{\mathcal{D} \qquad \dfrac{\dfrac{}{\mathsf{End} \vdash_{\emptyset} \mathsf{b}[\mathbf{0}]}\,[\mathsf{End}]}{\mathsf{End} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathsf{b}[\mathbf{0}] \parallel \mathsf{s}[\mathsf{c}!\mathsf{ship}] \parallel \mathsf{c}[C]}\,[\mathrm{WEAK}]}{\mathsf{G} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathsf{b}[B] \parallel \mathsf{s}[\mathsf{b}?\{\mathsf{add}.S,\ \mathsf{pay}.\mathsf{c}!\mathsf{ship}\}] \parallel \mathsf{c}[\mathsf{s}?\mathsf{ship}.C]}$$

where $\mathsf{G} = \mathsf{b} \to \mathsf{s}{:}\{\mathsf{add}.\,\mathsf{G},\ \mathsf{buy}\}$

$B = \mathsf{s}!\{\mathsf{add}.B,\ \mathsf{buy}\}$
$S = \mathsf{b}?\{\mathsf{add}.S,\ \mathsf{buy}.\mathsf{c}!\mathsf{ship}\}$
$C = \mathsf{s}?\mathsf{ship}.C$

Hence the Buyer-Seller-Carrier system is b-lock free

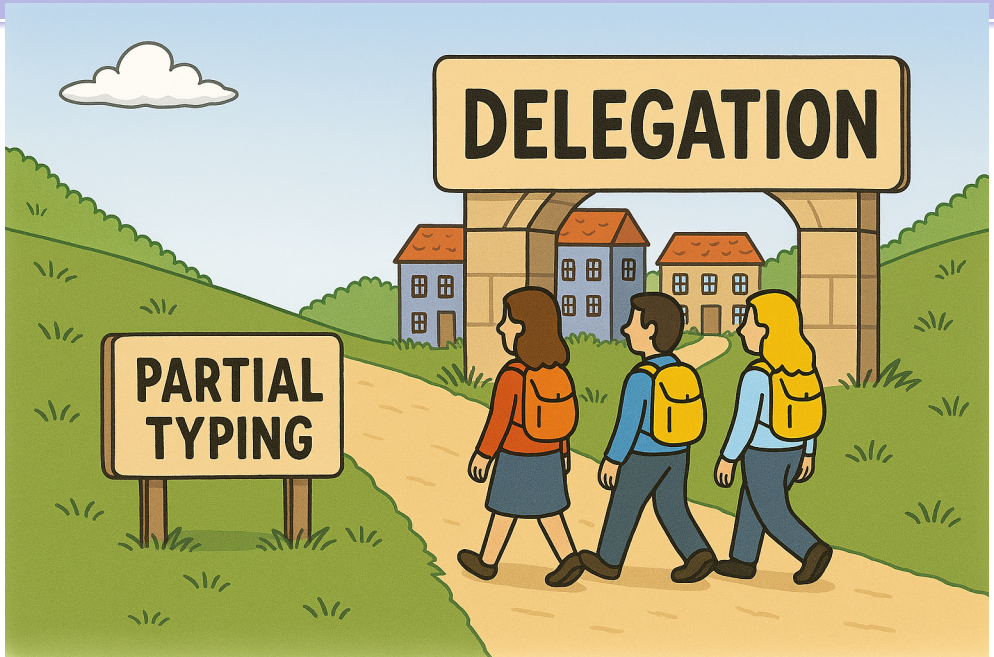
Typing non egalitarian systems

$$\mathcal{D} \quad = \quad \dfrac{\mathcal{D} \qquad \dfrac{\dfrac{}{\mathsf{End} \vdash_{\emptyset} \mathsf{b}[\mathbf{0}]}\,[\mathsf{End}]}{\mathsf{End} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathsf{b}[\mathbf{0}] \parallel \mathsf{s}[\mathsf{c}!\mathsf{ship}] \parallel \mathsf{c}[C]}\,[\textsc{weak}]}{\mathsf{G} \vdash_{\{\mathsf{s},\mathsf{c}\}} \mathsf{b}[B] \parallel \mathsf{s}[\mathsf{b}?\{\mathsf{add}.S,\ \mathsf{pay}.\mathsf{c}!\mathsf{ship}\}] \parallel \mathsf{c}[\mathsf{s}?\mathsf{ship}.C]}$$

where $\mathsf{G} = \mathsf{b} \to \mathsf{s}{:}\{\mathsf{add}.\,\mathsf{G},\ \mathsf{buy}\}$

$B = \mathsf{s}!\{\mathsf{add}.B,\ \mathsf{buy}\}$
$S = \mathsf{b}?\{\mathsf{add}.S,\ \mathsf{buy}.\mathsf{c}!\mathsf{ship}\}$
$C = \mathsf{s}?\mathsf{ship}.C$

Hence the Buyer-Seller-Carrier system is b-lock free


DELEGATION
PARTIAL TYPING

DELEGATION

THE ART OF GETTING THINGS DONE THROUGH OTHERS

DiPLO
www.diplomacy.edu

SMPS with internal delegation

Processes:

$$P ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.P_i\}_{i\in I} \mid \Subset\mathsf{p}; P \mid \ni\mathsf{p}; P \mid \mathsf{p}\Subset; P \mid \mathsf{p}\ni; P$$

$\Subset\mathsf{p}$	delegation request	$\mathsf{p}\Subset$	delegation end
$\mathsf{p}\Subset$	delegation acceptance	$\Subset\mathsf{p}$	resume from delegation

SMPS with internal delegation

Processes:

$$P ::=^{coind} \mathbf{0} \mid \mathtt{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathtt{p}?\{\lambda_i.P_i\}_{i\in I} \mid \Subset\mathtt{p};P \mid \ni\mathtt{p};P \mid \mathtt{p}\Subset;P \mid \mathtt{p}\ni;P$$

$\Subset\mathtt{p}$	delegation request	$\mathtt{p}\Subset$	delegation end
$\mathtt{p}\Subset$	delegation acceptance	$\Subset\mathtt{p}$	resume from delegation

Synchronous Operational Semantics:

$$[\textsc{Comm}] \quad \mathtt{p}[\mathtt{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathtt{q}[\mathtt{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathtt{p}\lambda_h\mathtt{q}} \mathtt{p}[P_h] \parallel \mathtt{q}[Q_h] \parallel \mathbb{M} \quad h\in I\subseteq J$$

$$[\textsc{DelStart}] \quad \mathtt{p}[\Subset\mathtt{q};P] \parallel \mathtt{q}[\mathtt{p}\Subset;Q] \parallel \mathbb{M} \xrightarrow{\mathtt{p}\Subset\mathtt{q}} \overset{*}{\mathtt{q}}\,[P] \parallel \mathtt{p}[Q] \parallel \mathbb{M}$$

$$[\textsc{DelEnd}] \quad \overset{*}{\mathtt{q}}\,[\ni\mathtt{p};P] \parallel \mathtt{p}[\mathtt{q}\ni Q] \parallel \mathbb{M} \xrightarrow{\mathtt{q}\ni\mathtt{p}} \mathtt{p}[P] \parallel \mathtt{q}[Q] \parallel \mathbb{M}$$

SMPS with internal delegation

A customer order an item to a seller and pays by sending the credit card number (ccn). The seller, after receiving the order, delegates to the bank the reception of ccn.

SMPS with internal delegation

A customer order an item to a seller and pays by sending the credit card number (ccn). The seller, after receiving the order, delegates to the bank the reception of ccn.

$$c[s!item.s!ccn] \quad \| \quad s[c?item.\Subset b;\ \Supset s] \quad \| \quad b[s\Subset;\ c?ccn.b\Supset]$$

SMPS with internal delegation

A customer order an item to a seller and pays by sending the credit card number (ccn). The seller, after receiving the order, delegates to the bank the reception of ccn.

$$c[s!item.s!ccn] \quad \| \quad s[c?item.\Subset b; \ni s] \quad \| \quad b[s \in; c?ccn.b \ni]$$

$$\downarrow citems$$

$$c[s!ccn] \quad \| \quad s[\Subset b; \ni s] \quad \| \quad b[s \in; c?ccn.b \ni]$$

SMPS with internal delegation

A customer order an item to a seller and pays by sending the credit card number (ccn). The seller, after receiving the order, delegates to the bank the reception of ccn.

$$c[s!item.s!ccn] \quad \| \quad s[c?item.\Subset b;\ \ni s] \quad \| \quad b[s\in;\ c?ccn.b\ni]$$

$$\downarrow citems$$

$$c[s!ccn] \quad \| \quad s[\Subset b;\ \ni s] \quad \| \quad b[s\in;\ c?ccn.b\ni]$$

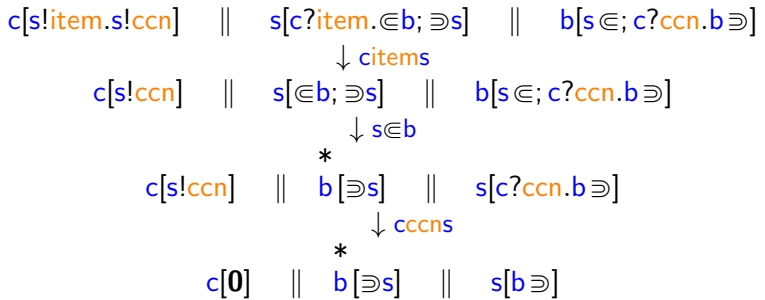
$$\downarrow s\Subset b$$

$$*$$

$$c[s!ccn] \quad \| \quad b\,[\ni s] \quad \| \quad s[c?ccn.b\ni]$$

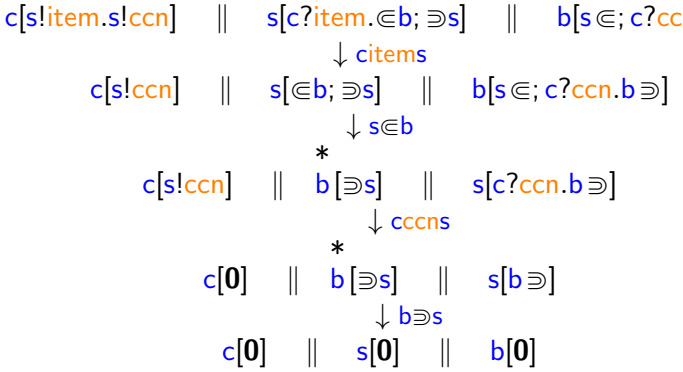
SMPS with internal delegation

A customer order an item to a seller and pays by sending the credit card number (ccn). The seller, after receiving the order, delegates to the bank the reception of ccn.



SMPS with internal delegation

A customer order an item to a seller and pays by sending the credit card number (ccn). The seller, after receiving the order, delegates to the bank the reception of ccn.



SMPS with internal delegation and partial typing

A partial type system

$$\mathsf{G} \vdash_{\mathcal{P}} \mathbb{M}$$

guaranteeing

- ($\mathcal{P}$-biased) Subject Reduction,
- Session Fidelity
- ($\mathcal{P}$-biased) Lock Freedom

SMPS with internal delegation and partial typing

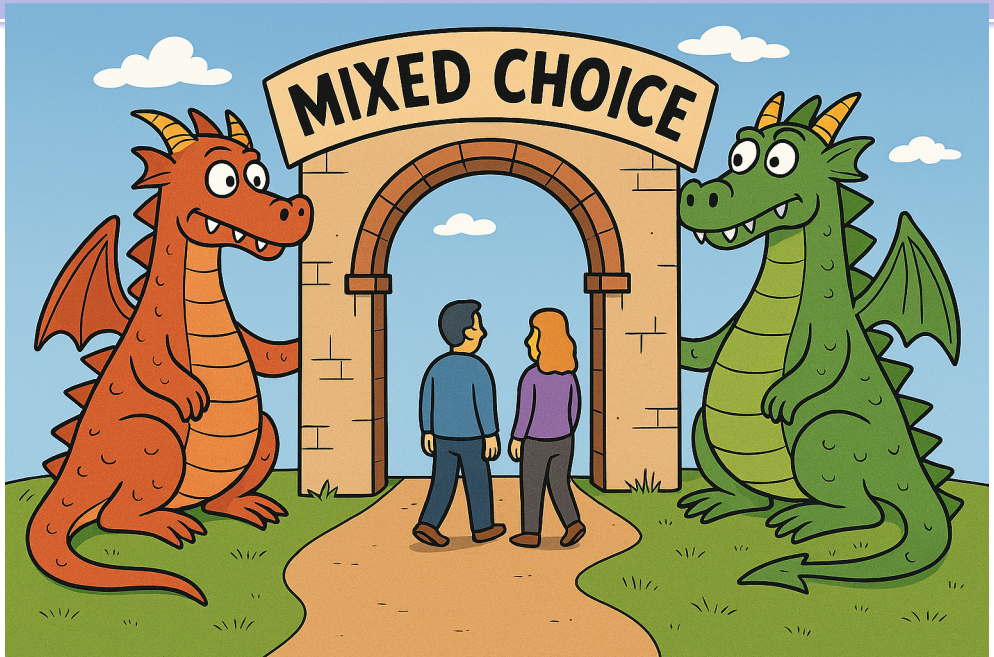
A partial type system

$$\mathsf{G} \vdash_{\mathcal{P}} \mathbb{M}$$

guaranteeing

- ($\mathcal{P}$-biased) Subject Reduction,
- Session Fidelity
- ($\mathcal{P}$-biased) Lock Freedom

Recall: Normal typing $\subseteq$ Partial typing


MIXED CHOICE

Processes **without** mixed choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.P_i\}_{i\in I}$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\ell \in I \subseteq J$$

$$\mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\lambda_\ell\mathsf{q}} \mathsf{p}[P_\ell] \parallel \mathsf{q}[Q_\ell] \parallel \mathbb{M}$$

Processes **without** mixed choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \mathtt{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathtt{p}?\{\lambda_i.P_i\}_{i\in I}$$

Multiparty Sessions

$$\mathbb{M} = \mathtt{p}_1[P_1] \parallel \cdots \parallel \mathtt{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\ell \in I \subseteq J$$

$$\mathtt{p}[\mathtt{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathtt{q}[\mathtt{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathtt{p}\lambda_\ell\mathtt{q}} \mathtt{p}[P_\ell] \parallel \mathtt{q}[Q_\ell] \parallel \mathbb{M}$$

Processes **without** mixed choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \mathtt{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathtt{p}?\{\lambda_i.P_i\}_{i\in I}$$

Multiparty Sessions

$$\mathbb{M} = \mathtt{p}_1[P_1] \parallel \cdots \parallel \mathtt{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\ell \in I \subseteq J$$

$$\mathtt{p}[\mathtt{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathtt{q}[\mathtt{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathtt{p}\lambda_\ell\mathtt{q}} \mathtt{p}[P_\ell] \parallel \mathtt{q}[Q_\ell] \parallel \mathbb{M}$$

Processes **without** mixed choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.P_i\}_{i\in I}$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\frac{\ell \in I \subseteq J}{\mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\lambda_\ell\mathsf{q}} \mathsf{p}[P_\ell] \parallel \mathsf{q}[Q_\ell] \parallel \mathbb{M}}$$

Processes **without** mixed choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \mathsf{p}!\{\lambda_i.P_i\}_{i\in I} \mid \mathsf{p}?\{\lambda_i.P_i\}_{i\in I}$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\ell \in I \subseteq J$$

$$\mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\lambda_\ell\mathsf{q}} \mathsf{p}[P_\ell] \parallel \mathsf{q}[Q_\ell] \parallel \mathbb{M}$$

Type disciplines for multiparty sessions **without** mixed choice

Type disciplines for multiparty sessions **without** mixed choice

Processes **with** Mixed Choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \Sigma_{i \in I} \pi_i.P_i \qquad \pi ::= \mathsf{p}?\mathsf{m}\lambda \mid \mathsf{p}!\mathsf{m}\lambda$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\mathsf{p}[\mathsf{q}!\mathsf{m}\lambda.P + P'] \parallel \mathsf{q}[\mathsf{p}?\mathsf{m}\lambda.Q + Q'] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\mathsf{m}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}$$

Processes **with** Mixed Choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \Sigma_{i \in I} \pi_i.P_i \qquad \pi ::= \mathsf{p}?\mathsf{m}\lambda \mid \mathsf{p}!\mathsf{m}\lambda$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\mathsf{p}[\mathsf{q}!\mathsf{m}\lambda.P + P'] \parallel \mathsf{q}[\mathsf{p}?\mathsf{m}\lambda.Q + Q'] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\mathsf{m}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}$$

Processes **with** Mixed Choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \Sigma_{i \in I} \pi_i.P_i \qquad \pi ::= \mathsf{p}?\mathsf{m}\lambda \mid \mathsf{p}!\mathsf{m}\lambda$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\mathsf{p}[\mathsf{q}!\mathsf{m}\lambda.P + P'] \parallel \mathsf{q}[\mathsf{p}?\mathsf{m}\lambda.Q + Q'] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\mathsf{m}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}$$

Processes **with** Mixed Choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \Sigma_{i \in I} \pi_i.P_i \qquad \pi ::= \mathsf{p}?\mathsf{m}\lambda \mid \mathsf{p}!\mathsf{m}\lambda$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\mathsf{p}[\mathsf{q}!\mathsf{m}\lambda.P + P'] \parallel \mathsf{q}[\mathsf{p}?\mathsf{m}\lambda.Q + Q'] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\mathsf{m}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}$$

Processes **with** Mixed Choice

Processes

$$P ::=^{coind} \mathbf{0} \mid \Sigma_{i \in I} \pi_i.P_i \qquad \pi ::= \mathsf{p}?\mathsf{m}\lambda \mid \mathsf{p}!\mathsf{m}\lambda$$

Multiparty Sessions

$$\mathbb{M} = \mathsf{p}_1[P_1] \parallel \cdots \parallel \mathsf{p}_n[P_n]$$

(synchronous) Operational Semantics

$$\mathsf{p}[\mathsf{q}!\mathsf{m}\lambda.P + P'] \parallel \mathsf{q}[\mathsf{p}?\mathsf{m}\lambda.Q + Q'] \parallel \mathbb{M} \xrightarrow{\mathsf{p}\mathsf{m}\lambda\mathsf{q}} \mathsf{p}[P] \parallel \mathsf{q}[Q] \parallel \mathbb{M}$$

Try and discipline that!

Much expressive power

Kirstin Peters, Nobuko Yoshida:
Separation and Encodability in Mixed Choice Multiparty Sessions. LICS 2024.

Usual approach to typing does not work

Usual approach to typing does not work

Example from by

Kirstin Peters, Nobuko Yoshida:
Separation and Encodability in Mixed Choice Multiparty Sessions. LICS, 2024.

inspired by

Catuscia Palamidessi:
Comparing the expressive power of the synchronous and asynchronous pi-calculi.
MSCS, 2003.

inspired by

Luc Bouge:
On the existence of symmetric algorithms to find leaders in networks of communicating sequential processes. Acta Inf., 1988.

Usual approach to typing does not work

Leader election

$$p[q!leader + r?leader.s!elect + s?del]$$

$$s[p?elect.q!del + q?electr!del + r?elect.p!del]$$

$$r[p!leader + q?leader.s!elect + s?del] \qquad q[r!leader + p?leader.s!elect + s?del]$$

Usual approach to typing does not work

Leader election

$$p[\mathbf{0}]$$

$$s[p?elect.q!del + q?electr!del + r?elect.p!del]$$

$$r[p!leader + q?leader.s!elect + s?del] \qquad q[s!elect]$$

Usual approach to typing does not work

Leader election

$p[\mathbf{0}]$

$s[r!del]$

$r[p!leader + q?leader.s!elect + s?del]$ $q[\mathbf{0}]$

Usual approach to typing does not work

Leader election

p[**0**]

s[**0**]

r[**0**] q[**0**]

A naive mixed-choice version of standard SMPS typing

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M}}{\mathsf{p} \to \mathsf{q} : \{\mathsf{m}\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\mathsf{q}!\{\mathsf{m}\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\mathsf{m}\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M}}$$

A naive mixed-choice version of standard SMPS typing

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M}}{\mathsf{p} \to \mathsf{q} : \{\mathsf{m}\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\mathsf{q}!\{\mathsf{m}\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\mathsf{m}\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M}}$$

essentially corresponds to

$$\frac{\mathsf{G}_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i \quad \{\Lambda_i\}_{i\in I} = \text{ reductions involving } \mathsf{p} \in \mathsf{prt}(\mathbb{M})}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \vdash \mathbb{M}}$$

A naive mixed-choice version of standard SMPS typing

$$\frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathsf{q}[Q_i] \parallel \mathbb{M}}{\mathsf{p} \to \mathsf{q} : \{\mathsf{m}\lambda_i.\mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\mathsf{q}!\{\mathsf{m}\lambda_i.P_i\}_{i\in I}] \parallel \mathsf{q}[\mathsf{p}?\{\mathsf{m}\lambda_j.Q_j\}_{j\in J}] \parallel \mathbb{M}}$$

essentially corresponds to

$$\frac{\mathsf{G}_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i \quad \{\Lambda_i\}_{i\in I} = \text{ reductions involving } \mathsf{p} \in \mathsf{prt}(\mathbb{M})}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \vdash \mathbb{M}}$$

NO WAY! $\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i$ would not capture the overall behaviour of the Election Session.

$$\frac{G_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i}{\Sigma_{i \in I} \Lambda_i . G_i \vdash \mathbb{M}}$$

$$\frac{G_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i}{\Sigma_{i \in I} \Lambda_i . G_i \vdash \mathbb{M}}$$

It would work for the Election Session

$$\frac{\mathsf{G}_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i}{\Sigma_{i \in I} \Lambda_i.\mathsf{G}_i \vdash \mathbb{M}}$$

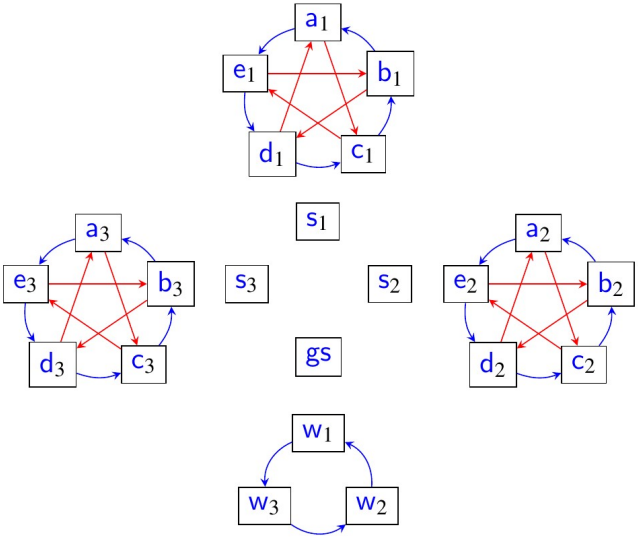
It would work for the Election Session

BUT

A global type would not “factorise” anything.
It would correspond to the complete reduction-tree.

Modular Election

Modular Election



Modular Election

$$\mathbb{E}^{gl} \;=\; \mathbb{E}_1 \parallel \mathbb{E}_2 \parallel \mathbb{E}_3 \parallel \mathbb{G}$$

where, for $1 \leq i \leq 3$,

$$\mathbb{E}_i \;=\; \mathsf{a}_i[P_{\mathsf{a}_i}] \parallel \mathsf{b}_i[P_{\mathsf{b}_i}] \parallel \mathsf{c}_i[P_{\mathsf{c}_i}] \parallel \mathsf{d}_i[P_{\mathsf{d}_i}] \parallel \mathsf{e}_i[P_{\mathsf{e}_i}] \parallel \mathsf{s}_i[P_{\mathsf{s}_i}]$$

with $P_{\mathsf{a}_i} = \mathsf{e}_i!leader + \mathsf{b}_i?leader.\left(\mathsf{c}_i!leader + \mathsf{d}_i?leader.\mathsf{s}_i!elect.\left(\mathsf{s}_i?gleader + \mathsf{s}_i?no\right)\right) + \mathsf{s}_i?del$

and $P_{\mathsf{s}_i} = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}?elect.\left(\mathsf{gs}?gleader.\mathsf{x}!gleader.Q_i + \mathsf{gs}?no.\mathsf{x}!no.Q_i\right)$
with $Q_i = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}!del$

$$\mathbb{G} \;=\; \mathsf{w}_1[P_{\mathsf{w}_1}] \parallel \mathsf{w}_2[P_{\mathsf{w}_2}] \parallel \mathsf{w}_3[P_{\mathsf{w}_3}] \parallel \mathsf{gs}[P_{\mathsf{gs}}]$$

with $P_{\mathsf{w}_i} = \mathsf{w}_{i+2}!leader + \mathsf{w}_{i+1}?leader.\mathsf{gs}!gleader + \mathsf{gs}?del$

and $P_{\mathsf{gs}} = \Sigma_{i\in\{1,2,3\}}\mathsf{w}_i?gleader.\mathsf{s}_i!gleader.\mathsf{s}_{i+1}!no.\mathsf{s}_{i+2}!no.\left(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!del\right)$

Modular Multiparty Sessions

Partitionable sessions such that

- Unrestricted mixed-choice can be used inside the modules;
- Inter-module communication through **connectors**;
- Connectors can interact using **one-to-one** mixed choice.

Modular Multiparty Sessions

Partitionable sessions such that

- Unrestricted mixed-choice can be used inside the modules;
- Inter-module communication through **connectors**;
- Connectors can interact using **one-to-one** mixed choice.

Modular Multiparty Sessions

Partitionable sessions such that

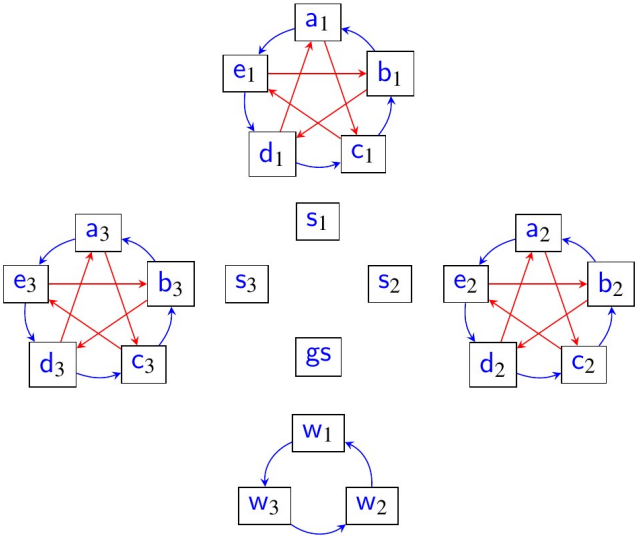
- Unrestricted mixed-choice can be used inside the modules;
- Inter-module communication through **connectors**;
- Connectors can interact using **one-to-one** mixed choice.

Modular Multiparty Sessions

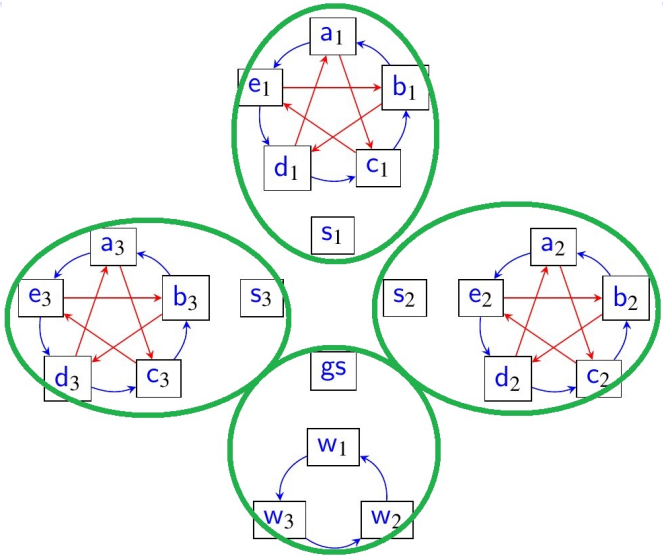
Partitionable sessions such that

- Unrestricted mixed-choice can be used inside the modules;
- Inter-module communication through **connectors**;
- Connectors can interact using **one-to-one** mixed choice.

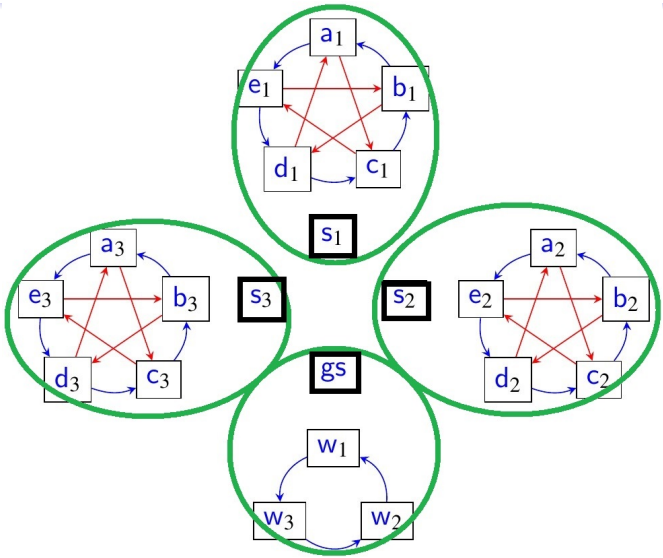
Modular Election



Modular Election



Modular Election



Modular Election

$$\mathbb{E}^{gl} \;=\; \mathbb{E}_1 \parallel \mathbb{E}_2 \parallel \mathbb{E}_3 \parallel \mathbb{G}$$

where, for $1 \leq i \leq 3$,

$$\mathbb{E}_i \;=\; \mathsf{a}_i[P_{\mathsf{a}_i}] \parallel \mathsf{b}_i[P_{\mathsf{b}_i}] \parallel \mathsf{c}_i[P_{\mathsf{c}_i}] \parallel \mathsf{d}_i[P_{\mathsf{d}_i}] \parallel \mathsf{e}_i[P_{\mathsf{e}_i}] \parallel \mathsf{s}_i[P_{\mathsf{s}_i}]$$

with $P_{\mathsf{a}_i} = \mathsf{e}_i!leader + \mathsf{b}_i?leader.\left(\mathsf{c}_i!leader + \mathsf{d}_i?leader.\mathsf{s}_i!elect.\left(\mathsf{s}_i?gleader + \mathsf{s}_i?no\right)\right) + \mathsf{s}_i?del$

and $P_{\mathsf{s}_i} = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}?elect.\left(\mathsf{gs}?gleader.\mathsf{x}!gleader.Q_i + \mathsf{gs}?no.\mathsf{x}!no.Q_i\right)$
with $Q_i = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}!del$

$$\mathbb{G} \;=\; \mathsf{w}_1[P_{\mathsf{w}_1}] \parallel \mathsf{w}_2[P_{\mathsf{w}_2}] \parallel \mathsf{w}_3[P_{\mathsf{w}_3}] \parallel \mathsf{gs}[P_{\mathsf{gs}}]$$

with $P_{\mathsf{w}_i} = \mathsf{w}_{i+2}!leader + \mathsf{w}_{i+1}?leader.\mathsf{gs}!gleader + \mathsf{gs}?del$

and $P_{\mathsf{gs}} = \Sigma_{i\in\{1,2,3\}}\mathsf{w}_i?gleader.\mathsf{s}_i!gleader.\mathsf{s}_{i+1}!no.\mathsf{s}_{i+2}!no.\left(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!del\right)$

Modular Election

$$\mathbb{E}^{gl} \;=\; \mathbb{E}_1 \,\|\, \mathbb{E}_2 \,\|\, \mathbb{E}_3 \,\|\, \mathbb{G}$$

where, for $1 \leq i \leq 3$,

$$\mathbb{E}_i \;=\; \mathsf{a}_i[P_{\mathsf{a}_i}] \,\|\, \mathsf{b}_i[P_{\mathsf{b}_i}] \,\|\, \mathsf{c}_i[P_{\mathsf{c}_i}] \,\|\, \mathsf{d}_i[P_{\mathsf{d}_i}] \,\|\, \mathsf{e}_i[P_{\mathsf{e}_i}] \;\|\; \mathsf{s}_i[P_{\mathsf{s}_i}]$$

$$\text{with} \quad P_{\mathsf{a}_i} \;=\; \mathsf{e}_i!\mathit{leader} + \mathsf{b}_i?\mathit{leader}.\Big(\mathsf{c}_i!\mathit{leader} + \mathsf{d}_i?\mathit{leader}.\mathsf{s}_i!\mathit{elect}.\big(\mathsf{s}_i?\mathit{gleader} + \mathsf{s}_i?\mathit{no}\big)\Big) + \mathsf{s}_i?\mathit{del}$$

$$\text{and} \quad P_{\mathsf{s}_i} \;=\; \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}?\mathit{elect}.\big(\mathsf{gs}?\mathit{gleader}.\mathsf{x}!\mathit{gleader}.Q_i + \mathsf{gs}?\mathit{no}.\mathsf{x}!\mathit{no}.Q_i\big) \quad \text{with } Q_i = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}!\mathit{del}$$

$$\mathbb{G} \;=\; \mathsf{w}_1[P_{\mathsf{w}_1}] \,\|\, \mathsf{w}_2[P_{\mathsf{w}_2}] \,\|\, \mathsf{w}_3[P_{\mathsf{w}_3}] \;\|\; \mathsf{gs}[P_{\mathsf{gs}}]$$

$$\text{with} \quad P_{\mathsf{w}_i} \;=\; \mathsf{w}_{i+2}!\mathit{leader} + \mathsf{w}_{i+1}?\mathit{leader}.\mathsf{gs}!\mathit{gleader} + \mathsf{gs}?\mathit{del}$$

$$\text{and} \quad P_{\mathsf{gs}} \;=\; \Sigma_{i\in\{1,2,3\}}\mathsf{w}_i?\mathit{gleader}.\mathsf{s}_i!\mathit{gleader}.\mathsf{s}_{i+1}!\mathit{no}.\mathsf{s}_{i+2}!\mathit{no}.\big(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!\mathit{del}\big)$$

Modular Election

$$\mathbb{E}^{gl} = \mathbb{E}_1 \| \mathbb{E}_2 \| \mathbb{E}_3 \| \mathbb{G}$$

where, for $1 \leq i \leq 3$,

$$\mathbb{E}_i = \mathsf{a}_i[P_{\mathsf{a}_i}] \,\|\, \mathsf{b}_i[P_{\mathsf{b}_i}] \,\|\, \mathsf{c}_i[P_{\mathsf{c}_i}] \,\|\, \mathsf{d}_i[P_{\mathsf{d}_i}] \,\|\, \mathsf{e}_i[P_{\mathsf{e}_i}] \,\|\, \mathsf{s}_i[P_{\mathsf{s}_i}]$$

with $P_{\mathsf{a}_i} = \mathsf{e}_i!leader + \mathsf{b}_i?leader.\left(\mathsf{c}_i!leader + \mathsf{d}_i?leader.\mathsf{s}_i!elect.\left(\mathsf{s}_i?gleader + \mathsf{s}_i?no\right)\right) + \mathsf{s}_i?del$

and $P_{\mathsf{s}_i} = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}?elect.\left(\mathsf{gs}?gleader.\mathsf{x}!gleader.Q_i + \mathsf{gs}?no.\mathsf{x}!no.Q_i\right)$ with $Q_i = \Sigma_{\mathsf{x}\in\{\mathsf{a}_i,\mathsf{b}_i,\mathsf{c}_i,\mathsf{d}_i,\mathsf{e}_i\}}\mathsf{x}!del$

$$\mathbb{G} = \mathsf{w}_1[P_{\mathsf{w}_1}] \,\|\, \mathsf{w}_2[P_{\mathsf{w}_2}] \,\|\, \mathsf{w}_3[P_{\mathsf{w}_3}] \,\|\, \mathsf{gs}[P_{\mathsf{gs}}]$$

with $P_{\mathsf{w}_i} = \mathsf{w}_{i+2}!leader + \mathsf{w}_{i+1}?leader.\mathsf{gs}!gleader + \mathsf{gs}?del$

and $P_{\mathsf{gs}} = \Sigma_{i\in\{1,2,3\}}\mathsf{w}_i?gleader.\mathsf{s}_i!gleader.\mathsf{s}_{i+1}!no.\mathsf{s}_{i+2}!no.\left(\Sigma_{i\in\{1,2,3\}}\mathsf{w}_i!del\right)$

Modular typing

$$\frac{\mathsf{G}_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i}{\Sigma_{i \in I} \Lambda_i.\mathsf{G}_i \vdash \mathbb{M}}$$

Modular typing

$$\frac{G_i \vdash \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i \quad \{\Lambda_i\}_{i\in I} = \text{ reductions involving } p \in \text{prt}(\mathbb{M})}{\Sigma_{i\in I}\Lambda_i.G_i \vdash \mathbb{M}}$$

Modular typing

$$\frac{\mathsf{G}_i \vdash^{\mathcal{P}} \mathbb{M}_i \qquad \mathbb{M} \xrightarrow{\Lambda_i} \mathbb{M}_i \qquad \{\Lambda_i\}_{i\in I} = \text{ reductions in a module of modularisation } \mathcal{P}}{\Sigma_{i\in I}\Lambda_i.\mathsf{G}_i \vdash^{\mathcal{P}} \mathbb{M}}$$

Some properties of modularised typing

- Modularisation is preserved by reduction;
- Typability does depend **neither** on the module one starts with **nor** on the chosen modularisation.

Some properties of modularised typing

- Modularisation is preserved by reduction;
- Typability does depend **neither** on the module one starts with **nor** on the chosen modularisation.

Some properties of modularised typing

- Modularisation is preserved by reduction;
- Typability does depend **neither** on the module one starts with
  nor on the chosen modularisation.

What do we get by typing?

- Subject Reduction;
- Session Fidelity;
- Lock Freedom.

What do we get by typing?

- Subject Reduction;
- Session Fidelity;
- Lock Freedom.

What do we get by typing?

- Subject Reduction;
- Session Fidelity;
- Lock Freedom.

What do we get by typing?

- Subject Reduction;
- Session Fidelity;
- Lock Freedom.


COMPOSITION

A DRAWBACK of standard MPST and SMPS

They are frameworks for CLOSED systems!!

A DRAWBACK of standard MPST and SMPS

They are frameworks for CLOSED systems!!

Good composition methods

They should be

- CONSERVATIVE
- FLEXIBLE
- SAFE

Good composition methods

They should be

- CONSERVATIVE
  Altering as less as possible the single systems
- FLEXIBLE
- SAFE

Good composition methods

They should be

- CONSERVATIVE
- FLEXIBLE

  i.e. "system independent": the composition mechanism
  - is not part of the system
  -allows to consider **any** system as potentially **open**
- SAFE

Good composition methods

They should be

- CONSERVATIVE
- FLEXIBLE
- SAFE

  Guaranteeing not to “break” relevant properties of the single systems we compose.

Good composition methods

They should be

- CONSERVATIVE
- FLEXIBLE
- SAFE

Composition Methods for MPST: Some proposals

Lorenzo Gheri, Nobuko Yoshida:
Hybrid Multiparty Session Types: Compositionality for Protocol Specification through Endpoint Projection. OOPSLA, 2023.

Claude Stolze, Marino Miculan, Pietro Di Gianantonio:
Composable partial multiparty session types for open systems. Softw.Syst.Model. 2023.

Composition Methods: the "participants-as-interfaces" (PaI) approach

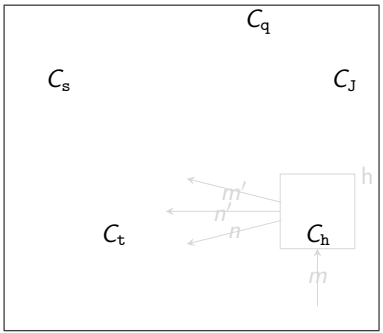
A very general idea applicable on systems with message-passing, when

participant behaviour  **system interface**

where interface = description of possible interactions with an outer system.

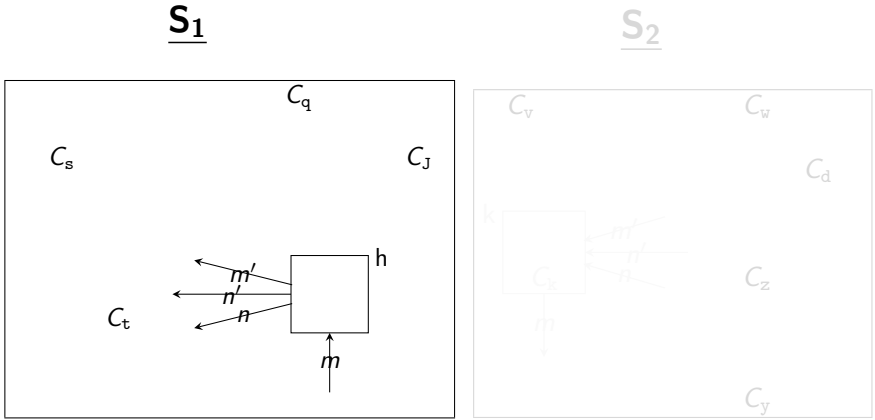
The "participants-as-interfaces" (PaI) approach

S1



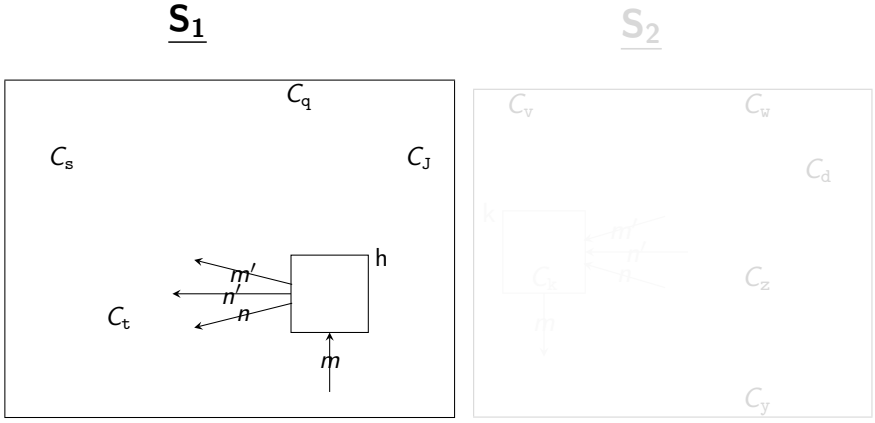
S2

The "participants-as-interfaces" (PaI) approach



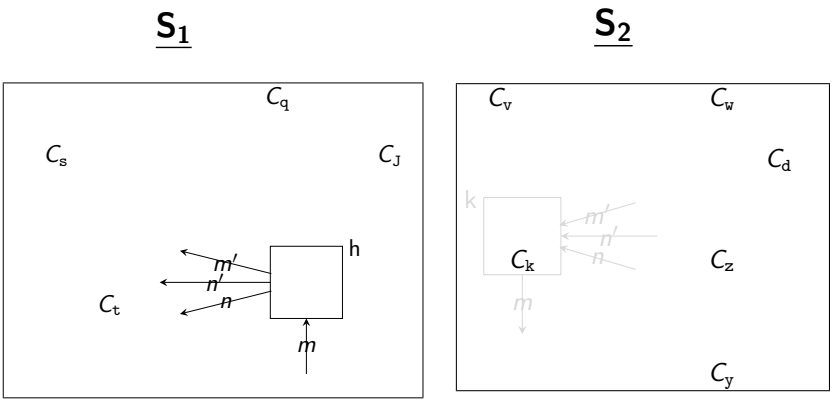
We abstract here from the way communications are performed and from the logical order of the exchanged messages

The "participants-as-interfaces" (PaI) approach

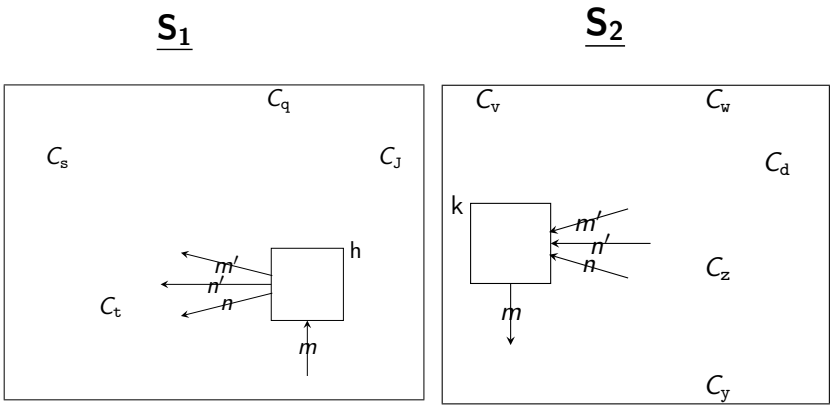


C_h's behaviour can be looked at as what can be offered by an outer system

The "participants-as-interfaces" (PaI) approach

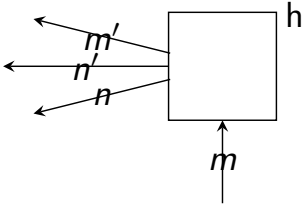


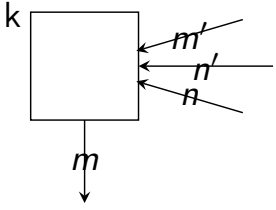
The "participants-as-interfaces" (PaI) approach



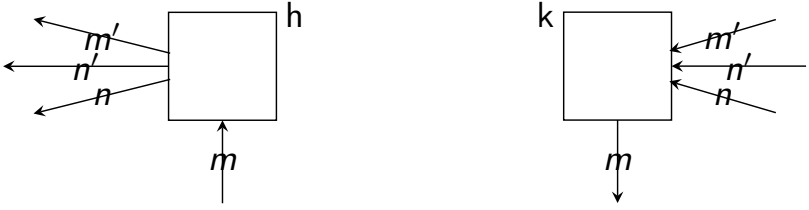
C_k offers what C_h requires

The "participants-as-interfaces" (PaI) approach



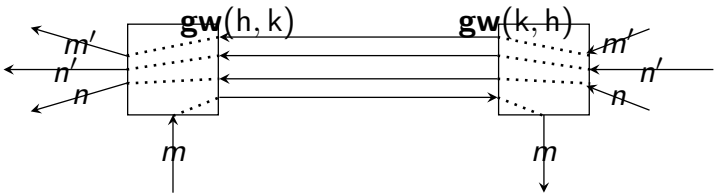


The "participants-as-interfaces" (PaI) approach

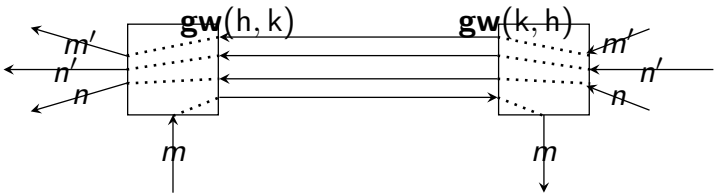


C_h and C_k are "complementary" interfaces: an h's input is a k's output, and vice versa

The "participants-as-interfaces" (PaI) approach



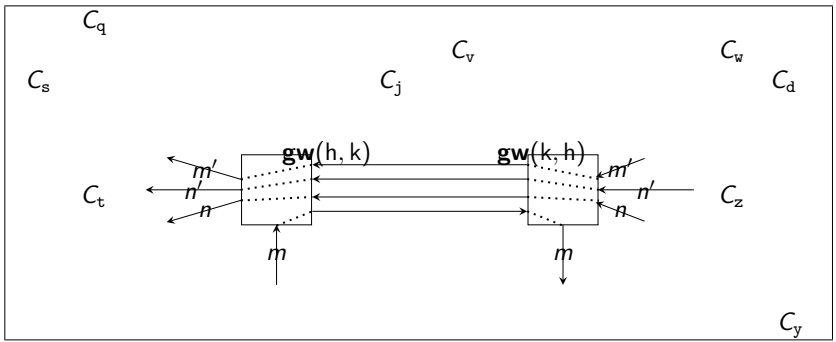
The "participants-as-interfaces" (PaI) approach



Composition via gateways:
the interface participants become forwarders dubbed gateways

The "participants-as-interfaces" (PaI) approach

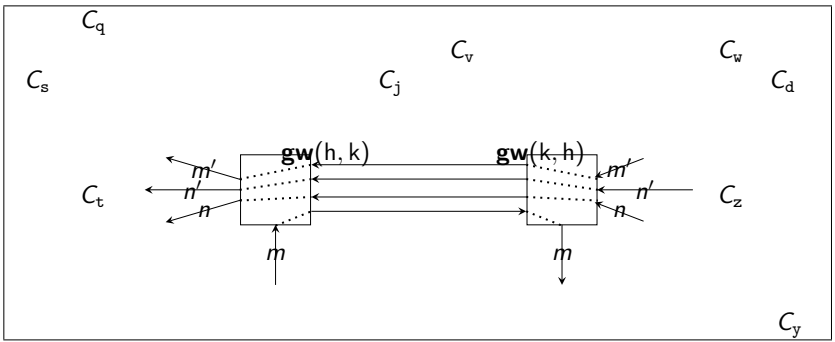
$$\underline{\mathbf{S_1}}^{\mathsf{h}\leftrightarrow\mathsf{k}}\underline{\mathbf{S_2}}$$



the compatible interfaces have been replaced by the corresponding gateways

The "participants-as-interfaces" (PaI) approach

$$\underline{\mathbf{S_1}}^{\mathsf{h}\leftrightarrow\mathsf{k}}\underline{\mathbf{S_2}}$$



the compatible interfaces have been replaced by the corresponding gateways

The "participants-as-interfaces" (PaI) approach

- **Conservative**
  Just one participant per system becomes a forwarder
- **Flexible**
  formalism independent, as far as based on message-passing
- **Safe?**
  Safety is formalism dependent.

  Safe for SMPS ✓

The "participants-as-interfaces" (PaI) approach

- Conservative
  Just one participant per system becomes a forwarder
- Flexible
  formalism independent, as far as based on message-passing
- Safe?
  Safety is formalism dependent.

  Safe for SMPS ✓

The "participants-as-interfaces" (PaI) approach

- Conservative
  Just one participant per system becomes a forwarder
- Flexible
  formalism independent, as far as based on message-passing
- Safe**?**
  Safety is formalism dependent.

Safe for SMPS ✓

The "participants-as-interfaces" (PaI) approach

- Conservative
  Just one participant per system becomes a forwarder
- Flexible
  formalism independent, as far as based on message-passing
- Safe**?**
  Safety is formalism dependent.

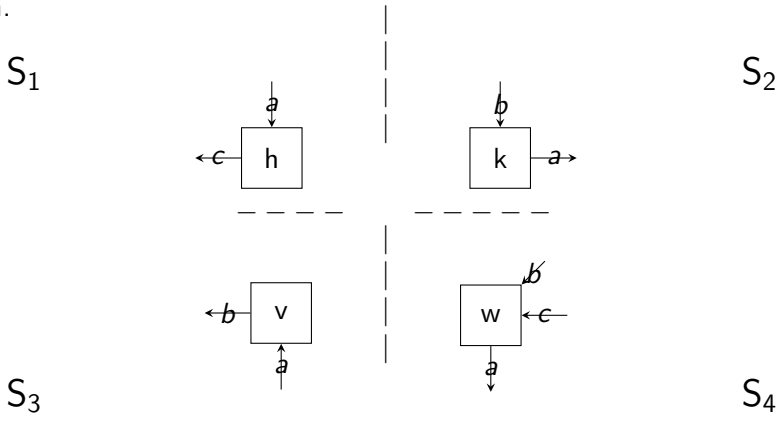
Safe for SMPS ✓

Simultaneous multiple connections

The composition via gateways trivially extends to simultaneous multiple system composition.

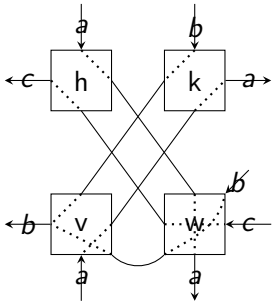
Simultaneous multiple connections

The composition via gateways trivially extends to simultaneous multiple system composition.



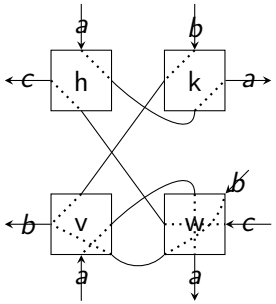
Simultaneous multiple connections

The composition via gateways trivially extends to simultaneous multiple system composition.



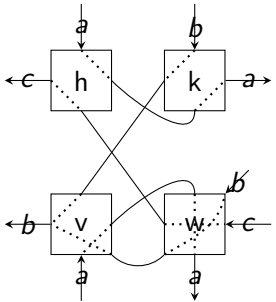
Simultaneous multiple connections

The composition via gateways trivially extends to simultaneous multiple system composition.



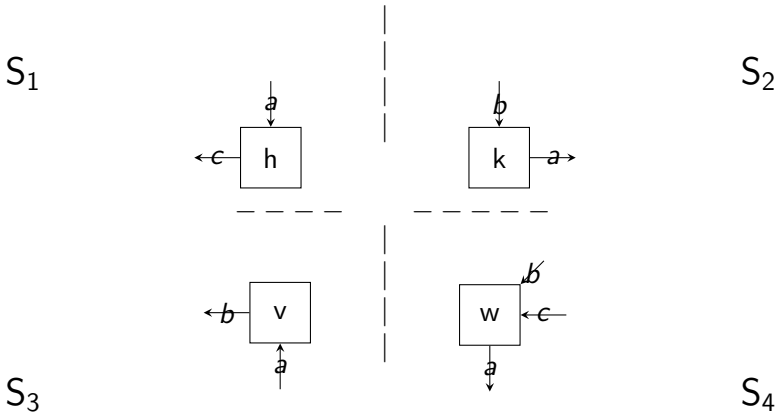
Simultaneous multiple connections

The composition via gateways trivially extends to simultaneous multiple system composition.

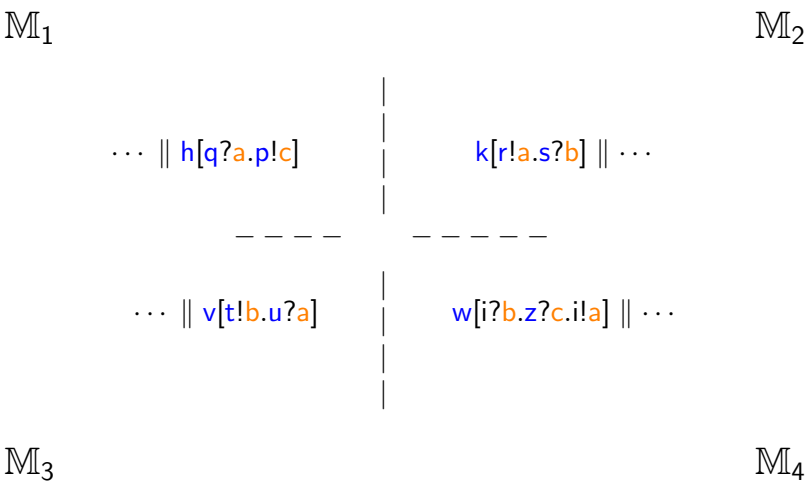


Issues: • Many different "connection policies".
• Which connection policies get safe composition?

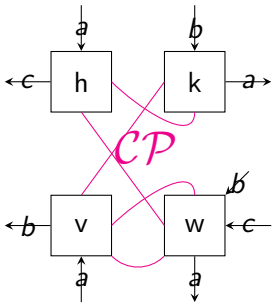
Systems represented as sessions



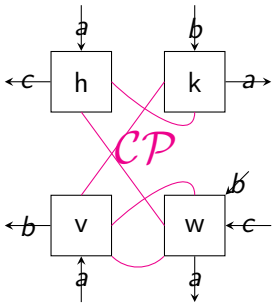
Systems represented as sessions



Connection Policies formalised as sessions



Connection Policies formalised as sessions



$\mathcal{CP} = \mathsf{h}[\mathsf{k}!a.\mathsf{w}?c] \;\|\; \mathsf{k}[\mathsf{h}?a.\mathsf{v}!b] \;\|\; \mathsf{v}[\mathsf{w}?b.\mathsf{w}!a] \;\|\; \mathsf{w}[\mathsf{v}!b.\mathsf{h}!c.\mathsf{v}?a]$

Connection Policies formalised as sessions

$\mathcal{CP} = \mathsf{h}[\mathsf{k}!a.\mathsf{w}?c] \;\|\; \mathsf{k}[\mathsf{h}?a.\mathsf{v}!b] \;\|\; \mathsf{v}[\mathsf{w}?b.\mathsf{w}!a] \;\|\; \mathsf{w}[\mathsf{v}!b.\mathsf{h}!c.\mathsf{v}?a]$

$\mathcal{CP}$ is typable

Connection Policies formalised as sessions

$\mathcal{CP} = \mathsf{h}[\mathsf{k}!a.\mathsf{w}?c] \;\|\; \mathsf{k}[\mathsf{h}?a.\mathsf{v}!b] \;\|\; \mathsf{v}[\mathsf{w}?b.\mathsf{w}!a] \;\|\; \mathsf{w}[\mathsf{v}!b.\mathsf{h}!c.\mathsf{v}?a]$

$\mathcal{CP}$ is typable, i.e. in the type system we can derive

$$\mathsf{G} \vdash \mathcal{CP}$$

where $\mathsf{G} = \mathsf{h} \to \mathsf{k}{:}a.\, \mathsf{w} \to \mathsf{v}{:}b.\, \mathsf{k} \to \mathsf{v}{:}b.\, \mathsf{w} \to \mathsf{h}{:}c.\, \mathsf{v} \to \mathsf{w}{:}a$

Connection Policies formalised as sessions

$\mathcal{CP} = \mathsf{h}[\mathsf{k}!a.\mathsf{w}?c] \;\|\; \mathsf{k}[\mathsf{h}?a.\mathsf{v}!b] \;\|\; \mathsf{v}[\mathsf{w}?b.\mathsf{w}!a] \;\|\; \mathsf{w}[\mathsf{v}!b.\mathsf{h}!c.\mathsf{v}?a]$

$\mathcal{CP}$ is typable, i.e. in the type system we can derive

$$\mathsf{G} \vdash \mathcal{CP}$$

where $\mathsf{G} = \mathsf{h} \to \mathsf{k}{:}a.\, \mathsf{w} \to \mathsf{v}{:}b.\, \mathsf{k} \to \mathsf{v}{:}b.\, \mathsf{w} \to \mathsf{h}{:}c.\, \mathsf{v} \to \mathsf{w}{:}a$

We can get a safe composition via gateways out of a typable connection policy.

Safety of PaI Composition of SMPS systems

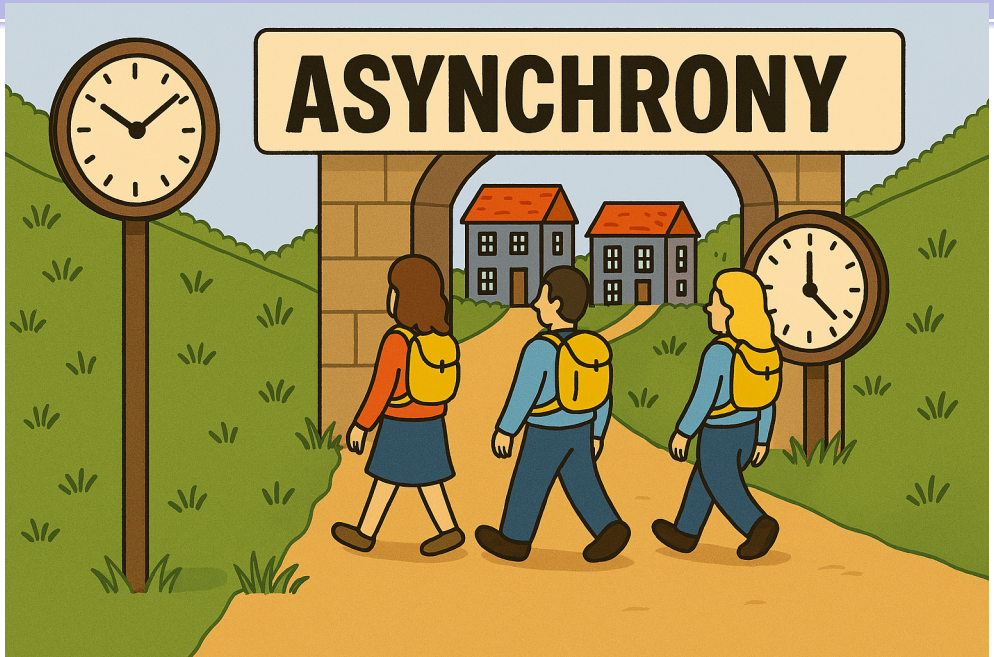
Let $\mathbb{M}_1, \ldots, \mathbb{M}_n$ be typable (and hence lock-free) sessions, and let $\mathcal{CP}$ be a typable connection policy.

Then the composition of $\mathbb{M}_1, \ldots, \mathbb{M}_n$ via the connection policy $\mathcal{CP}$ is typable (and hence lock-free).

Safety of PaI Composition of SMPS systems

Let $\mathbb{M}_1, \ldots, \mathbb{M}_n$ be typable (and hence lock-free) sessions, and let $\mathcal{CP}$ be a typable connection policy.

Then the composition of $\mathbb{M}_1, \ldots, \mathbb{M}_n$ via the connection policy $\mathcal{CP}$ is typable (and hence lock-free).


ASYNCHRONY

Formalising Asynchronous communications in SMPS

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

Formalising Asynchronous communications in SMPS

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

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

Formalising Asynchronous communications in SMPS

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

Formalising Asynchronous communications in SMPS

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

$$\mathsf{p}[\mathsf{q}?\{\lambda_i.P_i\}_{i\in I}] \parallel \mathbb{M} \parallel \langle \mathsf{q}, \lambda_h, \mathsf{p} \rangle \cdot \mathcal{Q} \xrightarrow{\mathsf{pq}?\lambda_h} \mathsf{p}[P_h] \parallel \mathbb{M} \parallel \mathcal{Q} \quad h \in I \quad [\textsc{Rcv}]$$

Formalising Asynchronous communications in SMPS

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

$$\mathsf{p}[\mathsf{q}?\{\lambda_i.P_i\}_{i\in I}] \parallel \mathbb{M} \parallel \langle \mathsf{q}, \lambda_h, \mathsf{p} \rangle \cdot \mathcal{Q} \xrightarrow{\mathsf{pq}?\lambda_h} \mathsf{p}[P_h] \parallel \mathbb{M} \parallel \mathcal{Q} \quad h \in I \quad [\textsc{Rcv}]$$

Asynchronous Global Types

$$\mathsf{G} \quad :::=^{coind} \quad \mathsf{pq}!\{\lambda_i.\mathsf{G}_i\}_{i\in I} \mid \mathsf{pq}?\lambda.\mathsf{G} \mid \texttt{End}$$

Splitting communications in global types

Asynchronous Global Types

$$\mathsf{G} \quad :::=^{coind} \quad \mathsf{pq}!\{\lambda_i.\mathsf{G}_i\}_{i\in I} \mid \mathsf{pq}?\lambda.\mathsf{G} \mid \texttt{End}$$

Splitting communications in global types

F.Dagnino, P. Giannini, M. Dezani:
Global types and event structure semantics for asynchronous multiparty sessions CoRR (2021), Fundamenta Informaticae (2024)

Asynchronous Global Types

$$\mathsf{G} \quad :::=^{coind} \quad \mathsf{pq}!\{\lambda_i.\mathsf{G}_i\}_{i\in I} \mid \mathsf{pq}?\lambda.\mathsf{G} \mid \mathtt{End}$$

Splitting communications in global types

Type Configurations $\mathsf{G} \parallel \mathcal{Q}$

Subject Reduction and Session Fidelity relate reductions on Session to reductions on Type Configurations.

Asynchronous Global Types

$$\mathsf{G} \quad :::=^{coind} \quad \mathsf{pq}!\{\lambda_i.\mathsf{G}_i\}_{i\in I} \mid \underline{\underline{\mathsf{pq}?\lambda.\mathsf{G}}} \mid \mathtt{End}$$

Splitting communications in global types

Type Configurations $\mathsf{G} \parallel \mathcal{Q}$

Subject Reduction and Session Fidelity relate reductions on Session to reductions on Type Configurations.

Type System(s)

$$[\text{T-End}] \quad \mathsf{End} \vdash \mathsf{p}[\mathbf{0}]$$

$$[\text{T-Send}] \quad \frac{\mathsf{G}_i \vdash \mathsf{p}[P_i] \parallel \mathbb{M} \parallel \mathcal{Q} \cdot \langle \mathsf{p}, \lambda_i, \mathsf{q} \rangle \qquad \forall i \in I}{\mathsf{pq}!\{\lambda_i.\mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathbb{M} \parallel \mathcal{Q}}$$

$$[\text{T-Rcv}] \quad \frac{\mathsf{G} \vdash \mathsf{p}[P_k] \parallel \mathbb{M} \parallel \mathcal{Q} \qquad k \in I}{\mathsf{pq}?\lambda_k.\mathsf{G} \vdash \mathsf{p}[\mathsf{q}!\{\lambda_i.P_i\}_{i \in I}] \parallel \mathbb{M} \parallel \langle \mathsf{p}, \lambda_k, \mathsf{q} \rangle \cdot \mathcal{Q}}$$

with suitable conditions on participants due to infinite derivations.

Extensions dealing with PARTIALITY and (internal) DELEGATION

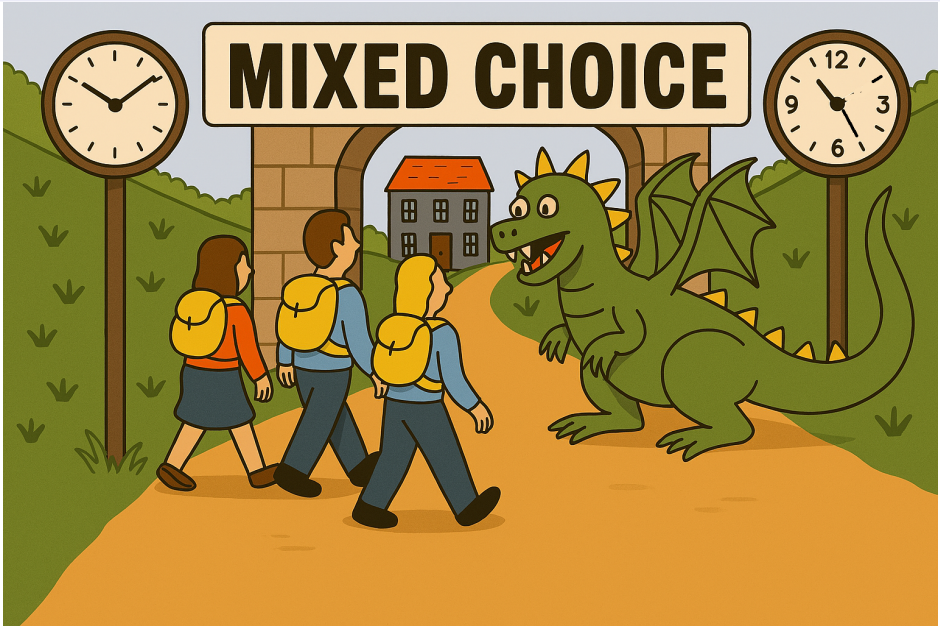
Decidable restrictions

Forcing:

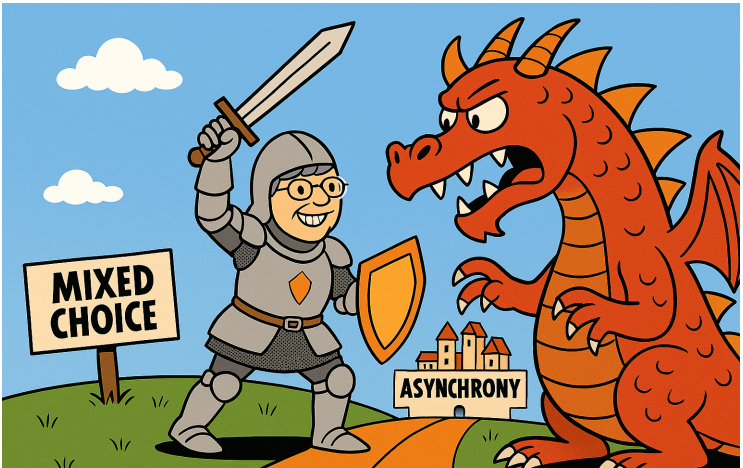
- Existential B-boundedness
- k-synchronisability
- ...

Asynch.-MPST $\leftarrow ? \rightarrow$ Asynch.-SMPS

Asynch.-MPST $\leftarrow ? \rightarrow$ Asynch.-SMPS


MIXED CHOICE

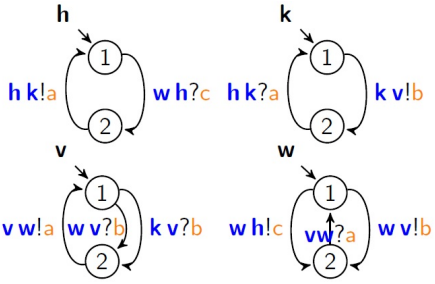
PLACES ‘26 - M.Dezani’s talk:
Asynchronous Multiparty Sessions with Mixed Choice



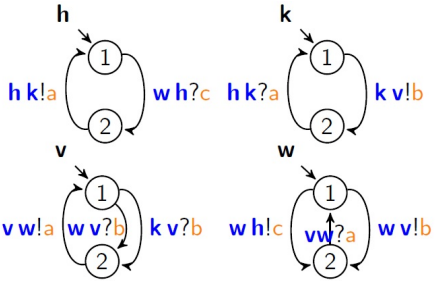
Possible use of mixed choice: Fault Tolerance


Communicating
Finite State Machines
Asynchronous
SMPS

CFSM ↘ ↙ Asynch-SMPS

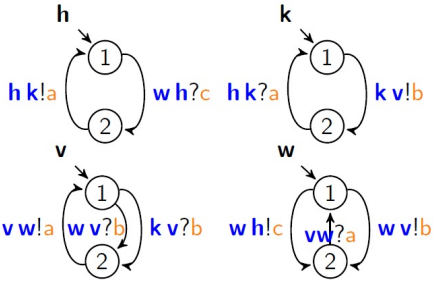


CFSM ↘ ↙ Asynch-SMPS



A formalism for Local Behaviours

CFSM ↘↙ Asynch-SMPS



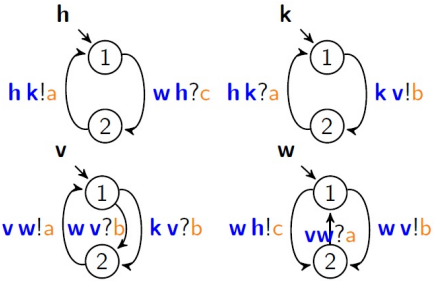
Many works on CFSM-MPST correspondence.

For instance

Julien Lange, Nobuko Yoshida:

Verifying Asynchronous Interactions via Communicating Session Automata. CAV (1) 2019

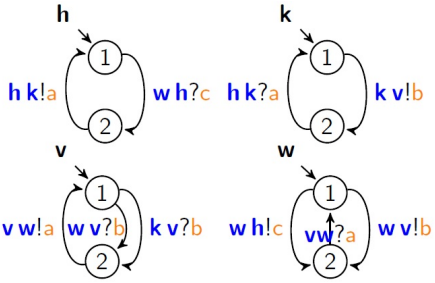
CFSM ↘ ↙ Asynch-SMPS



Many works on CFSM-MPST correspondence.

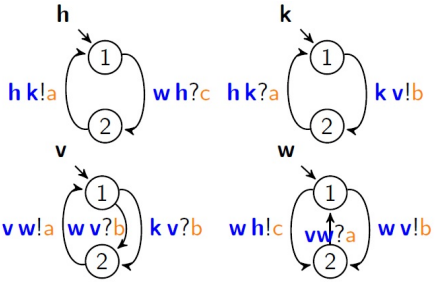
Roughly: Getting requirements (ensuring good properties) on CFSM from MPST theory.

CFSM ↘ ↙ Asynch-SMPS



What about CFSM-SMPS?

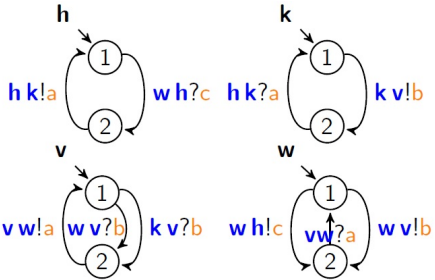
CFSM ↘ ↙ Asynch-SMPS



What about CFSM-SMPS?

CFSM systems $\equiv$ Asynchronous Multiparty Sessions

CFSM ↘ ↓ Asynch-SMPS



What about CFSM-SMPS?

$$\mathsf{h}[H] \parallel \mathsf{k}[K] \parallel \mathsf{v}[V] \parallel \mathsf{w}[W]$$

where

$$H = \mathsf{w}?\mathsf{c}.\mathsf{k}!\mathsf{a}.H \quad K = \mathsf{v}!\mathsf{b}.\mathsf{h}?\mathsf{a}.K \quad V = (\mathsf{w}?\mathsf{b} + \mathsf{k}?\mathsf{b}).\mathsf{w}!\mathsf{a}.V \quad W = (\mathsf{h}?\mathsf{c} + \mathsf{v}?\mathsf{b}).\mathsf{v}?\mathsf{a}.V$$

What is an SMPS asynchronous global type in the CFSM setting?

What is an SMPS asynchronous global type in the CFSM setting?

F. Stutz, E. D'Osualdo:
An Automata-theoretic Basis for Specification and Type Checking of Multiparty Protocols.
ESOP, 2025

What is an SMPS asynchronous global type in the CFSM setting?

F. Stutz, E. D'Osualdo:
An Automata-theoretic Basis for Specification and Type Checking of Multiparty Protocols.
ESOP, 2025

They propose a new general formalism for (finite-control) **global description** of protocols: Protocol State Machines (PSMs).

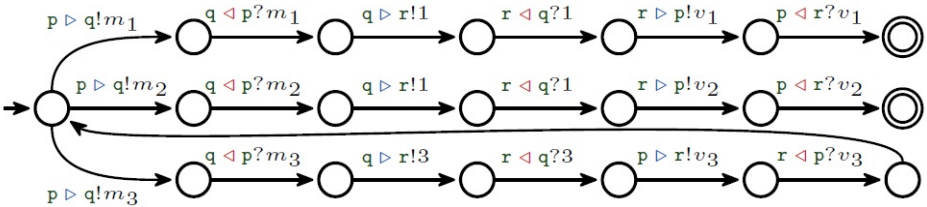
What is an SMPS asynchronous global type in the CFSM setting?

F. Stutz, E. D'Osualdo:
An Automata-theoretic Basis for Specification and Type Checking of Multiparty Protocols.
ESOP, 2025

They propose a new general formalism for (finite-control) **global description** of protocols: Protocol State Machines (PSMs).

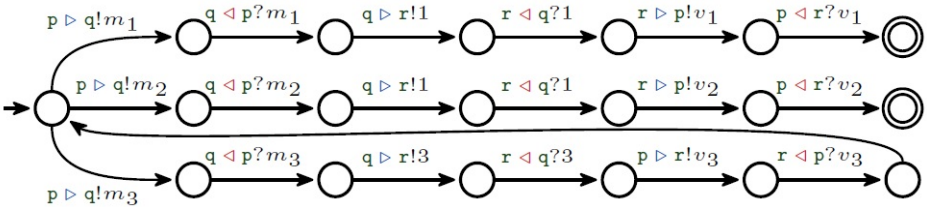
PSMs $\equiv$ Asynch.Global Types

CFSM Asynch-SMPS



A PSM

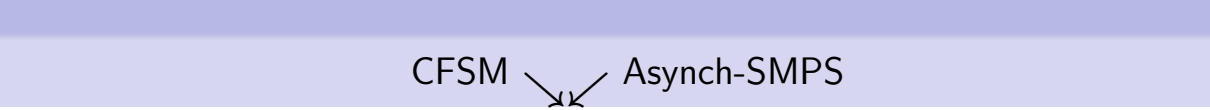
CFSM ↘ ↙ Asynch-SMPS



A PSM

$$\mathsf{G} = \mathsf{pq!}\begin{cases} m_1.\mathsf{qp?}m_1.\mathsf{qr!1.rq?1.rp!}v_1.\mathsf{pr?}v_1.\mathtt{End} \\ m_2.\mathsf{qp?}m_2.\mathsf{qr!1.rq?1.rp!}v_2.\mathsf{pr?}v_2.\mathtt{End} \\ m_3.\mathsf{qp?}m_3.\mathsf{qr!3.rq?3.rp!}v_3.\mathsf{rp?}v_3.\mathsf{G} \end{cases}$$

The corresponding asynch. global type



Vast prairies yet to be explored!

Anything IN-BETWEEN?

Anything IN-BETWEEN?

David Castro-Perez, Francisco Ferreira, Sung-Shik Jongmans:
A Synthetic Reconstruction of MultiParty Session Types. POPL, 2026

Anything IN-BETWEEN?

David Castro-Perez, Francisco Ferreira, Sung-Shik Jongmans:
A Synthetic Reconstruction of MultiParty Session Types. POPL, 2026

A typing approach where single participants are typed

Anything IN-BETWEEN?

David Castro-Perez, Francisco Ferreira, Sung-Shik Jongmans:
A Synthetic Reconstruction of MultiParty Session Types. POPL, 2026

A typing approach where single participants are typed

A nice step towards a compositional verification approach via typing
(drawback: all participants are typed with the same G)

Anything IN-BETWEEN?

David Castro-Perez, Francisco Ferreira, Sung-Shik Jongmans:
A Synthetic Reconstruction of MultiParty Session Types. POPL, 2026

A typing approach where single participants are typed

A nice step towards a compositional verification approach via typing
(drawback: all participants are typed with the same G)

A conjecture: participants separately typable **iff** whole session typable in SMPS
(by abstracting their processes)

Anything IN-BETWEEN?

David Castro-Perez, Francisco Ferreira, Sung-Shik Jongmans:
A Synthetic Reconstruction of MultiParty Session Types. POPL, 2026

A typing approach where single participants are typed

A nice step towards a compositional verification approach via typing
(drawback: all participants are typed with the same G)

A conjecture: participants separately typable **iff** whole session typable in SMPS
(by abstracting their processes)

A compositiona verification in MPST

L. Gheri, N. Yoshida: *Hybrid Multiparty Session Types: Compositionality for Protocol Specification through Endpoint Projection, OOPSLA (2023)*

Anything IN-BETWEEN?

David Castro-Perez, Francisco Ferreira, Sung-Shik Jongmans:
A Synthetic Reconstruction of MultiParty Session Types. POPL, 2026

A typing approach where single participants are typed

A nice step towards a compositional verification approach via typing
(drawback: all participants are typed with the same G)

A conjecture: participants separately typable **iff** whole session typable in SMPS
(by abstracting their processes)

A compositiona verification in MPST

L. Gheri, N. Yoshida: *Hybrid Multiparty Session Types: Compositionality for Protocol Specification through Endpoint Projection, OOPSLA (2023)*
Possible direction: SMPS for Gheri-Nobuko hybrid types

Presentation Partially Based On

] **Dagnino, F.**, Giannini, P., Dezani-Ciancaglini, M., 2023. Deconfined global types for asynchronous sessions. Logical Methods in Computer Science Volume 19, Issue 1.

] **Castellani, I.**, Dezani-Ciancaglini, M., **Giannini, P.**, 2022. Asynchronous sessions with input races, in: Carbone, M., Neykova, R. (Eds.), PLACES, Open Publishing Association. pp. 12–23.

] Barbanera, F., Bono, V., **Dezani-Ciancaglini, M.**, 2025a. Open compliance in multiparty sessions with partial typing. Journal of Logical and Algebraic Methods in Programming 144, 101046.

] Barbanera, F., **Bono, V.**, Dezani-Ciancaglini, M., 2025b. Partially typed multiparty sessions with internal delegation. J. Log. Algebraic Methods Program. 142, 101018.

] **Barbanera, F.**, Dezani-Ciancaglini, M., 2023. Partially typed multiparty sessions, in: Aubert, C., Di Giusto, C., Fowler, S., Safina, L. (Eds.), ICE, Open Publishing Association. pp. 15–34.

] Barbanera, F., Dezani-Ciancaglini, M., 2024. Asynchronous multiparty sessions with internal delegation, in: Essays Dedicated to Rocco De Nicola on The Occasion of His 70th Birthday, Springer. pp. 322–339.

] Barbanera, F., Dezani-Ciancaglini, M., 2025. Modular multiparty sessions with mixed choice, in: Aubert, C., Giusto, C.D., Fowler, S., Pun, V.K.I. (Eds.), ICE, pp. 2–20.

] Barbanera, F., Dezani-Ciancaglini, M., **de'Liguoro, U.**, 2022. Open compliance in multiparty sessions, in: Tarifa, S.L.T., Proença, J. (Eds.), FACS, pp. 222–243.

] Barbanera, F., Dezani-Ciancaglini, M., de'Liguoro, U., 2024a. Partial typing for asynchronous multiparty sessions, in: DCM'23, Open Publishing Association. pp. 1–20.

] Barbanera, F., Dezani-Ciancaglini, M., de'Liguoro, U., 2024b. Un-projectable global types for multiparty sessions, in: Bruni, A., Momigliano, A., Pradella, M., Rossi, M. (Eds.), PPDP, ACM Press. pp. 15:1–15:13.

] Barbanera, F., Dezani-Ciancaglini, M., **Gheri, L.**, **Yoshida, N.**, 2023. Multicompatibility for multiparty-session composition, in: Santiago Escobar, V.T.V. (Ed.), PPDP, Association for Computing Machinery. pp. 1–15.

] Barbanera, F., Dezani-Ciancaglini, M., **Lanese, I.**, **Tuosto, E.**, 2021. Composition and decomposition of multiparty sessions. J. Log. Algebraic Methods Program. 119, article 100620.

Conclusions

Conclusions


"That's all Folks!"


Thank
you