Deconfined Global Types for Asynchronous Sessions

Francesco Dagnino(1), Paola Giannini(2) and
Mariangiola Dezani-Ciancaglini(3)

(1) DIBRIS - Università di Genova
(2) DiSIT - Università del Piemonte Orientale, Italy
(3) Dipartimento di Informatica - Università di Torino, Italy

April 22, 2021

Global Types

ensure global properties of networks

Global Types

ensure global properties of networks

Structure of typing

global type	
↓	projection
local types	
↓	subtyping
refined local types	
↓	type assignement
processes	

Asynchronous communications

in traditional global types

- communications are atomic operations
- outputs and inputs are specified together

Asynchronous communications

in traditional global types

- communications are atomic operations
- outputs and inputs are specified together

- asynchronous communications are handled by subtyping
- asynchrnous subtyping allows to anticipate outputs/postpone inputs

Asynchronous communications

in traditional global types

- communications are atomic operations
- outputs and inputs are specified together

- asynchronous communications are handled by subtyping
- asynchrnous subtyping allows to anticipate outputs/postpone inputs

Issue

- asynchronous subtyping is undecidable
- the gap between global types and processes is too large!

Our approach

reduce the gap between global types and processes

Our approach

reduce the gap between global types and processes

- split outputs and inputs in global types

Our approach

reduce the gap between global types and processes

- split outputs and inputs in global types
- use simple subtyping

Our approach

reduce the gap between global types and processes

- split outputs and inputs in global types
- use simple subtyping
- well-formedness conditions on global types to ensure good properties

Outline

Processes

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

processes are regular terms

Processes

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

processes are regular terms

- $\mathsf{p}!\{\lambda_i; P_i\}_{i\in I}$ = internal choice/output
- $\mathsf{p}?\{\lambda_i; P_i\}_{i\in I}$ = external choice/input
- $\mathbf{0}$ = inactive process

Processes

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

processes are regular terms

- $\mathsf{p}!\{\lambda_i; P_i\}_{i\in I}$ = internal choice/output
- $\mathsf{p}?\{\lambda_i; P_i\}_{i\in I}$ = external choice/input
- $\mathbf{0}$ = inactive process

$$P = \mathsf{q}!\lambda_1; \mathsf{q}?\lambda_2; P \qquad Q = \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q$$

Networks and Queues

Network

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

where $\mathsf{p}_i \neq \mathsf{p}_j$ if $i \neq j$

Networks and Queues

Network

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

where $\mathsf{p}_i \neq \mathsf{p}_j$ if $i \neq j$

Queue

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

where $\langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is a message from p to q with label λ

we consider queues modulo an equivalence $\equiv$

$$\mathcal{M}_1 \cdot \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \cdot \langle \mathsf{r}, \lambda_2, \mathsf{s} \rangle \cdot \mathcal{M}_2 \equiv \mathcal{M}_1 \cdot \langle \mathsf{r}, \lambda_2, \mathsf{s} \rangle \cdot \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \cdot \mathcal{M}_2 \quad \text{if } \mathsf{p} \neq \mathsf{r} \text{ or } \mathsf{q} \neq \mathsf{s}$$

Networks and Queues

Network

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

where $\mathsf{p}_i \neq \mathsf{p}_j$ if $i \neq j$

Queue

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

where $\langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ is a message from p to q with label λ

we consider queues modulo an equivalence $\equiv$

$$\mathcal{M}_1 \cdot \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \cdot \langle \mathsf{r}, \lambda_2, \mathsf{s} \rangle \cdot \mathcal{M}_2 \equiv \mathcal{M}_1 \cdot \langle \mathsf{r}, \lambda_2, \mathsf{s} \rangle \cdot \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \cdot \mathcal{M}_2 \quad \text{if } \mathsf{p} \neq \mathsf{r} \text{ or } \mathsf{q} \neq \mathsf{s}$$

Session

network + queue $\quad \mathbb{N} \parallel \mathcal{M}$

LTS for asynchronous sessions

LTS for asynchronous sessions

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

LTS for asynchronous sessions

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

$$\mathsf{q}[\![\,\mathsf{p}?\{\lambda_j;Q_j\}_{j\in J}\,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p},\lambda_h,\mathsf{q}\rangle\cdot\mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\,Q_h\,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad \text{where } h\in J$$

LTS for asynchronous sessions

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

[

$$\mathsf{q}[\![\, \mathsf{p}?\{\lambda_j; Q_j\}_{j\in J} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad \text{where } h \in J$$

Example

$$P = \mathsf{q}!\lambda 1; \mathsf{q}?\lambda_2; P \quad Q = \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q$$

$$\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \emptyset$$

...

LTS for asynchronous sessions

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

[

$$\mathsf{q}[\![\, \mathsf{p}?\{\lambda_j; Q_j\}_{j\in J} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad \text{where } h \in J$$

Example

$$P = \mathsf{q}!\lambda 1; \mathsf{q}?\lambda_2; P \qquad Q = \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q$$

$$\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \emptyset \xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_1} \mathsf{p}[\![\, \mathsf{q}?\lambda_2; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle$$

$$\ldots$$

LTS for asynchronous sessions

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

$$\mathsf{q}[\![\,\mathsf{p}?\{\lambda_j;Q_j\}_{j\in J}\,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p},\lambda_h,\mathsf{q}\rangle\cdot\mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\,Q_h\,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad \text{where } h\in J$$

Example

$$P = \mathsf{q}!\lambda 1;\mathsf{q}?\lambda_2;P \qquad Q = \mathsf{p}!\lambda_2;\mathsf{p}?\lambda_1;Q$$

$$\begin{aligned}\mathsf{p}[\![\,P\,]\!] \parallel \mathsf{q}[\![\,Q\,]\!] \parallel \emptyset &\xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_1} \mathsf{p}[\![\,\mathsf{q}?\lambda_2;P\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}!\lambda_2;\mathsf{p}?\lambda_1;Q\,]\!] \parallel \langle \mathsf{p},\lambda_1,\mathsf{q}\rangle \\ &\xrightarrow{\mathsf{q}\,\mathsf{p}!\lambda_2} \mathsf{p}[\![\,\mathsf{q}?\lambda_2;P\,]\!] \parallel \mathsf{q}[\![\,\mathsf{p}?\lambda_1;Q\,]\!] \parallel \langle \mathsf{p},\lambda_1,\mathsf{q}\rangle\cdot\langle \mathsf{q},\lambda_2,\mathsf{p}\rangle\end{aligned}$$

...

LTS for asynchronous sessions

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

$$\mathsf{q}[\![\, \mathsf{p}?\{\lambda_j; Q_j\}_{j\in J} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad \text{where } h \in J$$

Example

$$P = \mathsf{q}!\lambda 1; \mathsf{q}?\lambda_2; P \qquad Q = \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q$$

$$\begin{aligned}
\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \emptyset &\xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_1} \mathsf{p}[\![\, \mathsf{q}?\lambda_2; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \\
&\xrightarrow{\mathsf{q}\,\mathsf{p}!\lambda_2} \mathsf{p}[\![\, \mathsf{q}?\lambda_2; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \cdot \langle \mathsf{q}, \lambda_2, \mathsf{p} \rangle \\
&\xrightarrow{\mathsf{q}\,\mathsf{p}?\lambda_2} \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{q}, \lambda_2, \mathsf{p} \rangle
\end{aligned}$$

$\ldots$

LTS for asynchronous sessions

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

[

$$\mathsf{q}[\![\, \mathsf{p}?\{\lambda_j; Q_j\}_{j\in J} \,]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}, \lambda_h, \mathsf{q} \rangle \cdot \mathcal{M} \xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_h} \mathsf{q}[\![\, Q_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad \text{where } h \in J$$

Example

$$P = \mathsf{q}!\lambda 1; \mathsf{q}?\lambda_2; P \qquad Q = \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q$$

$$\begin{aligned}
\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \emptyset &\xrightarrow{\mathsf{p}\,\mathsf{q}!\lambda_1} \mathsf{p}[\![\, \mathsf{q}?\lambda_2; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}!\lambda_2; \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \\
&\xrightarrow{\mathsf{q}\,\mathsf{p}!\lambda_2} \mathsf{p}[\![\, \mathsf{q}?\lambda_2; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{p}, \lambda_1, \mathsf{q} \rangle \cdot \langle \mathsf{q}, \lambda_2, \mathsf{p} \rangle \\
&\xrightarrow{\mathsf{q}\,\mathsf{p}?\lambda_2} \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}?\lambda_1; Q \,]\!] \parallel \langle \mathsf{q}, \lambda_2, \mathsf{p} \rangle \\
&\xrightarrow{\mathsf{p}\,\mathsf{q}?\lambda_1} \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \emptyset \\
&\ldots
\end{aligned}$$

Asynchronous Global Types

Asynchronous Global Types

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

processes are regular terms

Asynchronous Global Types

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

processes are regular terms

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

Asynchronous Global Types

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

processes are regular terms

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

Example

$$G = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{q}\,\mathsf{p}!\lambda_2; \mathsf{q}\,\mathsf{p}?\lambda_2; \mathsf{p}\,\mathsf{q}?\lambda_1; G$$

Projection

Judgement: $\mathsf{G} \upharpoonright \mathsf{p} \mapsto P$

Projection

Judgement: $\mathsf{G} \upharpoonright \mathsf{p} \mapsto P$

$$[\text{EXT}]\ \dfrac{}{\mathsf{G} \upharpoonright \mathsf{p} \mapsto \mathbf{0}}\ \mathsf{p} \notin \mathsf{players}(\mathsf{G})$$

Projection

Judgement: $\mathsf{G} \upharpoonright \mathsf{p} \mapsto P$

$$[\textsc{Ext}]\ \frac{}{\mathsf{G} \upharpoonright \mathsf{p} \mapsto \mathbf{0}}\ \ \mathsf{p} \notin \mathsf{players}(\mathsf{G})$$

$$[\textsc{In-Rcv}]\ \frac{\mathsf{G} \upharpoonright \mathsf{q} \mapsto P}{(\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G}) \upharpoonright \mathsf{q} \mapsto \mathsf{p}?\lambda; P} \qquad [\textsc{In-Ext}]\ \frac{\mathsf{G} \upharpoonright \mathsf{s} \mapsto P}{(\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G}) \upharpoonright \mathsf{s} \mapsto P}\ \ \begin{array}{l}\mathsf{s} \neq \mathsf{q}\\ \mathsf{s} \in \mathsf{players}(\mathsf{G})\end{array}$$

Projection

Judgement: $\mathsf{G} \upharpoonright \mathsf{p} \mapsto P$

$$[\textsc{Ext}]\ \frac{}{\mathsf{G} \upharpoonright \mathsf{p} \mapsto \mathbf{0}} \quad \mathsf{p} \notin \mathsf{players}(\mathsf{G})$$

$$[\textsc{In-Rcv}]\ \frac{\mathsf{G} \upharpoonright \mathsf{q} \mapsto P}{(\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G}) \upharpoonright \mathsf{q} \mapsto \mathsf{p}?\lambda; P} \qquad [\textsc{In-Ext}]\ \frac{\mathsf{G} \upharpoonright \mathsf{s} \mapsto P}{(\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G}) \upharpoonright \mathsf{s} \mapsto P} \quad \begin{array}{l}\mathsf{s} \neq \mathsf{q}\\ \mathsf{s} \in \mathsf{players}(\mathsf{G})\end{array}$$

$$[\textsc{Out-Snd}]\ \frac{\mathsf{G}_i \upharpoonright \mathsf{p} \mapsto P_i \quad \forall i \in I}{(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}) \upharpoonright \mathsf{p} \mapsto \mathsf{q}!\{\lambda_i; P_i\}_{i\in I}}$$

Projection

Judgement: $\mathsf{G} \restriction \mathsf{p} \mapsto P$

$$\mathcal{C} ::= [\,]_n \quad | \quad \mathsf{p}?\{\lambda_i; \mathcal{C}_i\}_{i \in I} \quad | \quad \mathsf{p}!\{\lambda_i; \mathcal{C}_i\}_{i \in I} \quad | \quad P$$

$$[\textsc{Out-Rcv}] \frac{\mathsf{G}_i \restriction \mathsf{q} \mapsto \mathcal{C}[\mathsf{p}?\lambda_i; P_{i,j}]_{j \in J} \quad \forall i \in I}{(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I}) \restriction \mathsf{q} \mapsto \mathcal{C}[\mathsf{p}?\{\lambda_i; P_{i,j}\}_{i \in I}]_{j \in J}} \quad \mathsf{q} \in \mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I})$$

Projection

Judgement: $\mathsf{G} \restriction \mathsf{p} \mapsto P$

$$\mathcal{C} ::= [\,]_n \quad | \quad \mathsf{p}?\{\lambda_i; \mathcal{C}_i\}_{i \in I} \quad | \quad \mathsf{p}!\{\lambda_i; \mathcal{C}_i\}_{i \in I} \quad | \quad P$$

$$[\text{Out-Rcv}]\ \frac{\mathsf{G}_i \restriction \mathsf{q} \mapsto \mathcal{C}[\mathsf{p}?\lambda_i; P_{i,j}]_{j \in J} \quad \forall i \in I}{(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I}) \restriction \mathsf{q} \mapsto \mathcal{C}[\mathsf{p}?\{\lambda_i; P_{i,j}\}_{i \in I}]_{j \in J}}\ \mathsf{q} \in \mathsf{players}(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I})$$

$$[\text{Out-Ext}]\ \frac{\mathsf{G}_i \restriction \mathsf{s} \mapsto \mathcal{C}[\mathsf{r}?\lambda'_i; R_{i,j}]_{j \in J} \quad \forall i \in I}{(\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I}) \restriction \mathsf{s} \mapsto \mathcal{C}[\mathsf{r}?\{\lambda'_i; R_{i,j}\}_{i \in I}]_{j \in J}}\ \begin{array}{l}\mathsf{s} \notin \{\mathsf{p}, \mathsf{q}\} \\ \mathsf{s} \in \mathsf{players}(\mathsf{G}_i)\ \forall i \in I\end{array}$$

Projection is a function

$$\mathsf{G} \upharpoonright \mathsf{p} \mapsto P_1 \textit{ and } \mathsf{G} \upharpoonright \mathsf{p} \mapsto P_2 \textit{ imply } P_1 = P_2$$

proved under the assumption that G is bounded

An example

$$
\begin{aligned}
\mathsf{G} &= \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{G}_1, \quad \lambda_2; \mathsf{G}_2\} \\
\mathsf{G}_i &= \mathsf{q}\,\mathsf{p}!\{\lambda_3; \mathsf{p}\,\mathsf{q}?\lambda_i; \mathsf{q}\,\mathsf{p}?\lambda_3; \mathsf{End}, \\
&\qquad\qquad \lambda_4; \mathsf{p}\,\mathsf{q}?\lambda_i; \mathsf{q}\,\mathsf{p}?\lambda_4; \mathsf{G}\}
\end{aligned}
$$

An example

$$\begin{aligned}
\mathsf{G} &= \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{G}_1, \quad \lambda_2; \mathsf{G}_2\} \\
\mathsf{G}_i &= \mathsf{q}\,\mathsf{p}!\{\lambda_3; \mathsf{p}\,\mathsf{q}?\lambda_i; \mathsf{q}\,\mathsf{p}?\lambda_3; \mathsf{End}, \\
&\qquad\qquad \lambda_4; \mathsf{p}\,\mathsf{q}?\lambda_i; \mathsf{q}\,\mathsf{p}?\lambda_4; \mathsf{G}\}
\end{aligned}$$

$$\mathsf{G} \restriction \mathsf{p} = P \qquad \begin{aligned} P &= \mathsf{q}!\{\lambda_1; P_1, \quad \lambda_2; P_1\} \\ P_1 &= \mathsf{q}?\{\lambda_3; \mathbf{0}, \quad \lambda_4; P\} \end{aligned} \qquad \mathcal{C} = []$$

An example

$$\mathsf{G} = \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{G}_1, \quad \lambda_2; \mathsf{G}_2\}$$

$$\begin{aligned}\mathsf{G}_i = \mathsf{q}\,\mathsf{p}!\{&\lambda_3; \mathsf{p}\,\mathsf{q}?\lambda_i; \mathsf{q}\,\mathsf{p}?\lambda_3; \mathsf{End},\\ &\lambda_4; \mathsf{p}\,\mathsf{q}?\lambda_i; \mathsf{q}\,\mathsf{p}?\lambda_4; \mathsf{G}\}\end{aligned}$$

$$\mathsf{G} \restriction \mathsf{p} = P \quad \begin{aligned} P &= \mathsf{q}!\{\lambda_1; P_1, \quad \lambda_2; P_1\} \quad \mathcal{C} = []\\ P_1 &= \mathsf{q}?\{\lambda_3; \mathbf{0}, \quad \lambda_4; P\}\end{aligned}$$

$$\mathsf{G} \restriction \mathsf{q} = Q \quad \begin{aligned} Q &= \mathsf{p}!\{\lambda_3; Q_3, \quad \lambda_4; Q_2\} \quad \mathcal{C} = \mathsf{p}!\{\lambda_3; []_1, \quad \lambda_4; []_2\}\\ Q_1 &= \mathsf{p}?\{\lambda_1; \mathbf{0}, \lambda_2; \mathbf{0}\}\\ Q_2 &= \mathsf{p}?\{\lambda_1; Q, \quad \lambda_2; Q\}\end{aligned}$$

From global types to networks

Type assignement $\mathbb{N} : \mathsf{G}$

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

From global types to networks

Type assignement $\mathbb{N} : \mathsf{G}$

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

- $\mathsf{players}(\mathsf{G}) \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$
- for all $i \in 1..n$, $P_i \leq \mathsf{G} \upharpoonright \mathsf{p}_i$

From global types to networks

Type assignement $\mathbb{N} : \mathsf{G}$

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

- $\mathsf{players}(\mathsf{G}) \subseteq \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$
- for all $i \in 1..n$, $P_i \leq \mathsf{G} \upharpoonright \mathsf{p}_i$

where subtyping $\leq$ is the simple one:
it allows more inputs and less outputs

Progress property

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

Progress property

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

- no deadlocks
  all non-terminated derivatives of $\mathbb{N} \parallel \mathcal{M}$ are live

Progress property

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

- no deadlocks
  all non-terminated derivatives of $\mathbb{N} \parallel \mathcal{M}$ are live
- no locked inputs
  all inputs will eventually be satisfied

Progress property

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

- no deadlocks
  all non-terminated derivatives of $\mathbb{N} \parallel \mathcal{M}$ are live
- no locked inputs
  all inputs will eventually be satisfied
- no orphan messages
  all messages in the queue will eventually be read

formalised using parallel reduction

$$\mathbb{N} \parallel \mathcal{M} \stackrel{\Delta}{\Rightarrow} \mathbb{N}' \parallel \mathcal{M}'$$

Δ is a maximal set of possible coherent actions

Well-formedness

assigning a global type to a network does not ensure progress

there are global types specifying “wrong” protocols:

Well-formedness

assigning a global type to a network does not ensure progress

there are global types specifying "wrong" protocols:

- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}$ deadlocked!

Well-formedness

assigning a global type to a network does not ensure progress

there are global types specifying "wrong" protocols:

- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}$ deadlocked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_1; \mathsf{q}\,\mathsf{r}!\lambda_3; \mathsf{q}\,\mathsf{r}?\lambda_3; \mathsf{G},\ \lambda_2; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}\}$ r is locked!

Well-formedness

assigning a global type to a network does not ensure progress

there are global types specifying "wrong" protocols:

- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}$ deadlocked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_1; \mathsf{q}\,\mathsf{r}!\lambda_3; \mathsf{q}\,\mathsf{r}?\lambda_3; \mathsf{G},\ \lambda_2; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}\}$ r is locked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{q}\,\mathsf{p}!\lambda_2; \mathsf{q}\,\mathsf{p}?\lambda_2; \mathsf{G}$ λ_1 is orphan!

Well-formedness

assigning a global type to a network does not ensure progress

there are global types specifying "wrong" protocols:

- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}$ deadlocked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_1; \mathsf{q}\,\mathsf{r}!\lambda_3; \mathsf{q}\,\mathsf{r}?\lambda_3; \mathsf{G},\ \lambda_2; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}\}$ r is locked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{q}\,\mathsf{p}!\lambda_2; \mathsf{q}\,\mathsf{p}?\lambda_2; \mathsf{G}$ λ_1 is orphan!

We need well-formedness conditions on global types

Well-formedness

assigning a global type to a network does not ensure progress

there are global types specifying "wrong" protocols:

- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}$ deadlocked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\{\lambda_1; \mathsf{p}\,\mathsf{q}?\lambda_1; \mathsf{q}\,\mathsf{r}!\lambda_3; \mathsf{q}\,\mathsf{r}?\lambda_3; \mathsf{G},\ \lambda_2; \mathsf{p}\,\mathsf{q}?\lambda_2; \mathsf{G}\}$ r is locked!
- $\mathsf{G} = \mathsf{p}\,\mathsf{q}!\lambda_1; \mathsf{q}\,\mathsf{p}!\lambda_2; \mathsf{q}\,\mathsf{p}?\lambda_2; \mathsf{G}$ λ_1 is orphan!

We need well-formedness conditions on global types

well-formednes depends on the message queue $\mathcal{M}$:
$\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{End}$ well-formed if $\mathcal{M} = \langle \mathsf{p}, \lambda, \mathsf{q} \rangle$ but ill formed if $\mathcal{M} = \emptyset$

Well-formedness

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

- G is projectable for all participants
- G is bounded
  (all players occur at bounded depth in all paths of G)
- $G \parallel \mathcal{M}$ is input/output matching

Typing

$$\frac{\mathbb{N} : \mathsf{G} \qquad \mathsf{G} \parallel \mathcal{M} \text{ is well formed}}{\mathcal{M} \vdash \mathbb{N} : \mathsf{G}}$$

Typing

$$\frac{\mathbb{N} : \mathsf{G} \qquad \mathsf{G} \parallel \mathcal{M} \text{ is well formed}}{\mathcal{M} \vdash \mathbb{N} : \mathsf{G}}$$

Properties

- Subject Reduction
- Session Fidelity
- Progress

Input/Output Matching

$$[\textsc{End}]\ \dfrac{}{\vdash_{\mathsf{iom}} \mathsf{End} \parallel \emptyset} \qquad [\textsc{In}]\ \dfrac{\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}}{\vdash_{\mathsf{iom}} \mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G} \parallel \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}}$$

$$[\textsc{Out}]\ \dfrac{\vdash_{\mathsf{iom}} \mathsf{G}_i \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_i, \mathsf{q}\rangle \quad \forall i \in I}{\vdash_{\mathsf{iom}} \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i \in I} \parallel \mathcal{M}}$$

Input/Output Matching

$$[\textsc{End}]\ \frac{}{\vdash_{\mathsf{iom}} \mathsf{End} \parallel \emptyset} \qquad [\textsc{In}]\ \frac{\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}}{\vdash_{\mathsf{iom}} \mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G} \parallel \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}}\ \vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M})$$

$$[\textsc{Out}]\ \frac{\vdash_{\mathsf{iom}} \mathsf{G}_i \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_i, \mathsf{q}\rangle \quad \forall i \in I}{\vdash_{\mathsf{iom}} \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \parallel \mathcal{M}}\ \vdash_{\mathsf{read}} (\mathsf{G}_i, \mathcal{M} \cdot \langle \mathsf{p}, \lambda_i, \mathsf{q}\rangle)\ \forall i \in I$$

$\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M})$ checks that G reads all messages in $\mathcal{M}$

Input/Output Matching

$$[\textsc{End}]\ \frac{}{\vdash_{\mathsf{iom}} \mathsf{End} \parallel \emptyset} \qquad [\textsc{In}]\ \frac{\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}}{\vdash_{\mathsf{iom}} \mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G} \parallel \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}}\ \vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M})$$

$$[\textsc{Out}]\ \frac{\vdash_{\mathsf{iom}} \mathsf{G}_i \parallel \mathcal{M} \cdot \langle \mathsf{p}, \lambda_i, \mathsf{q}\rangle \quad \forall i \in I}{\vdash_{\mathsf{iom}} \mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I} \parallel \mathcal{M}}\ \vdash_{\mathsf{read}} (\mathsf{G}_i, \mathcal{M} \cdot \langle \mathsf{p}, \lambda_i, \mathsf{q}\rangle)\ \forall i \in I$$

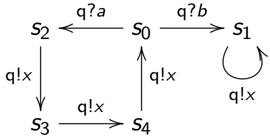
$\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M})$ checks that G reads all messages in $\mathcal{M}$

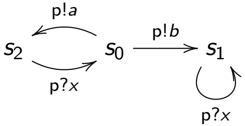
$$[\textsc{Empty-R}]\ \frac{}{\vdash_{\mathsf{read}} (\mathsf{G}, \emptyset)} \qquad [\textsc{Out-R}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}_i, \mathcal{M}) \quad (\forall i \in I)}{\vdash_{\mathsf{read}} (\mathsf{p}\,\mathsf{q}!\{\lambda_i; \mathsf{G}_i\}_{i\in I}, \mathcal{M})}$$

$$[\textsc{In-R1}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M})}{\vdash_{\mathsf{read}} (\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G}, \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M})}$$

$$[\textsc{In-R2}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M})}{\vdash_{\mathsf{read}} (\mathsf{p}\,\mathsf{q}?\lambda; \mathsf{G}, \mathcal{M})}\ \mathcal{M} \not\equiv \langle \mathsf{p}, \lambda, \mathsf{q}\rangle \cdot \mathcal{M}'$$

A well-formed type





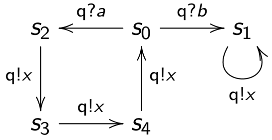
$$P = \mathsf{q}?\{a; \mathsf{q}!x; \mathsf{q}!x; \mathsf{q}!x; P,\ \ b; P_1\}$$

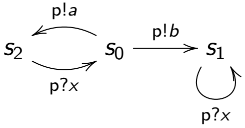
$$P_1 = \mathsf{q}!x; P_1$$

$$Q = \mathsf{p}!\{a; \mathsf{p}?x; Q,\ \ b; Q_1\}$$

$$Q_1 = \mathsf{p}?x; Q_1$$

A well-formed type





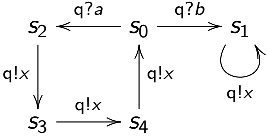
$P = \mathsf{q}?\{a; \mathsf{q}!x; \mathsf{q}!x; \mathsf{q}!x; P,\ \ b; P_1\}$ $\qquad$ $Q = \mathsf{p}!\{a; \mathsf{p}?x; Q,\ \ b; Q_1\}$

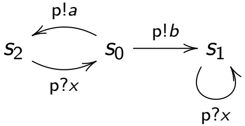
$P_1 = \mathsf{q}!x; P_1$ $\qquad$ $Q_1 = \mathsf{p}?x; Q_1$

$$\mathsf{G} = \mathsf{q}\,\mathsf{p}!\{a; \mathsf{q}\,\mathsf{p}?a; \mathsf{p}\,\mathsf{q}!x; \mathsf{p}\,\mathsf{q}!x; \mathsf{p}\,\mathsf{q}!x; \mathsf{p}\,\mathsf{q}?x; \mathsf{G},\ \ b; \mathsf{q}\,\mathsf{p}?b; \mathsf{G}_1\}$$

$$\mathsf{G}_1 = \mathsf{p}\,\mathsf{q}!x; \mathsf{p}\,\mathsf{q}?x; \mathsf{G}_1$$

A well-formed type





$$P = \mathsf{q}?\{a; \mathsf{q}!x; \mathsf{q}!x; \mathsf{q}!x; P,\ \ b; P_1\} \qquad Q = \mathsf{p}!\{a; \mathsf{p}?x; Q,\ \ b; Q_1\}$$

$$P_1 = \mathsf{q}!x; P_1 \qquad Q_1 = \mathsf{p}?x; Q_1$$

$$\mathsf{G} = \mathsf{q\,p}!\{a; \mathsf{q\,p}?a; \mathsf{p\,q}!x; \mathsf{p\,q}!x; \mathsf{p\,q}!x; \mathsf{p\,q}?x; \mathsf{G},\ \ b; \mathsf{q\,p}?b; \mathsf{G}_1\}$$

$$\mathsf{G}_1 = \mathsf{p\,q}!x; \mathsf{p\,q}?x; \mathsf{G}_1$$

$\mathsf{G} \parallel \emptyset$ is well-formed

A sound algorithm

$\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ is coinductively defined: it may be undecidable

A sound algorithm

$\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ is coinductively defined: it may be undecidable

a sound algorithm based on cycle detection

Judgement: $\mathcal{H} \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ where $\mathcal{H}$ keeps track of type configurations encountered during the proof

A sound algorithm

$\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ is coinductively defined: it may be undecidable

a sound algorithm based on cycle detection

Judgement: $\mathcal{H} \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ where $\mathcal{H}$ keeps track of type configurations encountered during the proof

Key rule

$$[\textsc{Cycle}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M}) \quad \vdash_{\mathsf{agr}} (\mathsf{G}, \mathcal{M}'') \quad \vdash_{\mathsf{dread}} (\mathsf{G}, \mathcal{M}'')}{\mathcal{H}_1, (\mathsf{G}, \mathcal{M}), \mathcal{H}_2 \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}'}\ \ \mathcal{M}' \equiv \mathcal{M} \cdot \mathcal{M}''$$

A sound algorithm

$\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ is coinductively defined: it may be undecidable

a sound algorithm based on cycle detection

Judgement: $\mathcal{H} \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ where $\mathcal{H}$ keeps track of type configurations encountered during the proof

Key rule

$$[\textsc{Cycle}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M}) \quad \vdash_{\mathsf{agr}} (\mathsf{G}, \mathcal{M}'') \quad \vdash_{\mathsf{dread}} (\mathsf{G}, \mathcal{M}'')}{\mathcal{H}_1, (\mathsf{G}, \mathcal{M}), \mathcal{H}_2 \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}'} \quad \mathcal{M}' \equiv \mathcal{M} \cdot \mathcal{M}''$$

- in a cycle the queue grows

A sound algorithm

$\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ is coinductively defined: it may be undecidable

a sound algorithm based on cycle detection

Judgement: $\mathcal{H} \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ where $\mathcal{H}$ keeps track of type configurations encountered during the proof

Key rule

$$[\text{Cycle}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M}) \quad \vdash_{\mathsf{agr}} (\mathsf{G}, \mathcal{M}'') \quad \vdash_{\mathsf{dread}} (\mathsf{G}, \mathcal{M}'')}{\mathcal{H}_1, (\mathsf{G}, \mathcal{M}), \mathcal{H}_2 \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}'} \quad \mathcal{M}' \equiv \mathcal{M} \cdot \mathcal{M}''$$

- in a cycle the queue grows
- $\vdash_{\mathsf{agr}} (\mathsf{G}, \mathcal{M}'') = \mathcal{M}''$ does not interferes with G

A sound algorithm

$\vdash_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ is coinductively defined: it may be undecidable

a sound algorithm based on cycle detection

Judgement: $\mathcal{H} \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}$ where $\mathcal{H}$ keeps track of type configurations encountered during the proof

Key rule

$$[\textsc{Cycle}]\ \frac{\vdash_{\mathsf{read}} (\mathsf{G}, \mathcal{M}) \quad \vdash_{\mathsf{agr}} (\mathsf{G}, \mathcal{M}'') \quad \vdash_{\mathsf{dread}} (\mathsf{G}, \mathcal{M}'')}{\mathcal{H}_1, (\mathsf{G}, \mathcal{M}), \mathcal{H}_2 \vdash^{\mathcal{I}}_{\mathsf{iom}} \mathsf{G} \parallel \mathcal{M}'} \quad \mathcal{M}' \equiv \mathcal{M} \cdot \mathcal{M}''$$

- in a cycle the queue grows
- $\vdash_{\mathsf{agr}} (\mathsf{G}, \mathcal{M}'') = \mathcal{M}''$ does not interferes with G
- $\vdash_{\mathsf{dread}} (\mathsf{G}, \mathcal{M}'') = \mathcal{M}''$ is consumed in all paths of G

Conclusion and future work

Contributions

- a new formalism of global types splitting outputs and inputs
- a decidable type-checking for asynchronous sessions
- an algorithm ensuring well-formedness of global types
- prototype implementation in co-logic programming (available on GitHub)

Conclusion and future work

Contributions

- a new formalism of global types splitting outputs and inputs
- a decidable type-checking for asynchronous sessions
- an algorithm ensuring well-formedness of global types
- prototype implementation in co-logic programming (available on GitHub)

Further work

- formal comparison with traditional global types + asynchronous subtyping
- proof/counterexamples of completeness of the algorithm for well-formedness
- ...

References

- Ilaria Castellani, Mariangiola Dezani-Ciancaglini and Paola Giannini. "Global types and event structure semantics for asynchronous multiparty sessions", *(submitted to LMCS)*
- Francesco Dagnino, Paola Giannini and Mariangiola Dezani-Ciancaglini. "Deconfined global types for asynchronous sessions", *(Coordination 2021)*
- Riccardo Bianchini and Francesco Dagnino. "Asynchronous global types in co-logic programming", *(Coordination 2021)*

Questions?

Thank you!