Asynchronous Sessions with Input Races

Paola Giannini*

joint work with Ilaria Castellani‡ and Mariangiola Dezani†

∗ DiSSTE, Università del Piemonte Orientale
‡ INRIA Sophia Antipolis, France
† Dipartimento di Informatica, Università di Torino

INRIA, Sophia Antipolis - March 26, 2024

Overall goal

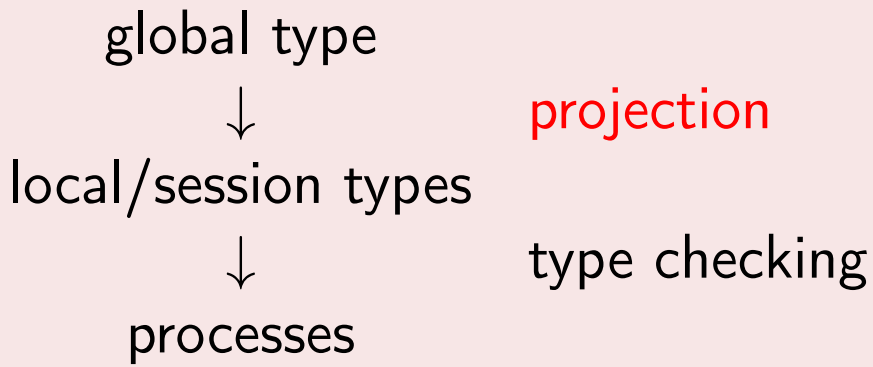
Define a permissive type system for multiparty asynchronous sessions,

- allowing syntactic input races in processes (= choices of inputs from different senders)
- provided they do not hinder lock-freedom.

Global Types

Global types ensure global properties of multiparty sessions

Classical Structure of Typing



- Projectability is the main tool to enforce good properties,
- Projectability imposes restrictions on the choices that can be used in a protocol.
- It is difficult to accommodate protocols with input races!

Example 1: A "fake" input race

$$\boxed{\texttt{Alice}}\ \texttt{Bob!arr;Bob?arr;(}\ \texttt{Bob!tellC;Carol!wArr} \oplus \texttt{Bob!yTellC}\ \texttt{)}$$

$$\boxed{\texttt{Bob}}\ \texttt{Alice!arr;Alice?arr;(}\ \texttt{Alice?tellC} + \texttt{Alice?yTellC;Carol!wArr}\ \texttt{)}$$

$$\boxed{\texttt{Carol}}\ \texttt{(}\ \texttt{Alice?wArr} + \texttt{Bob?wArr}\ \texttt{)}$$

Example 2: A "confluent" input race

Alice `Carol!arr;...`

Bob `Carol!arr;...`

Carol `( Alice?arr;Bob?arr;... + Bob?arr;Alice?arr;... )`

Objective and Approach

Define global types for multiparty sessions such that

- typable multiparty sessions enjoy lock freedom
- protocols such as the ones of the previous examples be typable

Approach

- define more expressive global types: split outputs and inputs
- bypass projection by matching directly the global type with the multiparty session imposing restrictions on the global type

Processes

$$P ::=_{\rho} \oplus_{i \in I} \mathsf{p}_i!\ell_i; P_i \mid \Sigma_{i \in I} \mathsf{p}_i?\ell_i; P_i \mid 0$$

coinductive definition, processes are regular terms

- $\oplus_{i \in I} \mathsf{p}_i!\ell_i; P_i$ = internal choice/output
- $\Sigma_{i \in I} \mathsf{p}_i?\ell_i; P_i$ = external choice/input
- 0 = inactive process

Example

$$P = \mathsf{q}\,!\,\ell_1; \mathsf{q}\,?\,\ell_2; P \qquad Q = \mathsf{p}\,!\,\ell_2; \mathsf{p}\,?\,\ell_1; Q$$

Multiparty Sessions

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 $i \neq j$. $\mathsf{Players}(\mathbb{N}) = \{\mathsf{p} \mid \mathsf{p}[\![P]\!] \in \mathbb{N} \ \wedge \ P \neq 0\}$

Queue

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

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

we consider queues modulo an equivalence $\equiv$

$$\mathcal{M}_1 \cdot \langle \mathsf{p}, \ell_1, \mathsf{q} \rangle \cdot \langle \mathsf{r}, \ell_2, \mathsf{s} \rangle \cdot \mathcal{M}_2 \equiv \mathcal{M}_1 \cdot \langle \mathsf{r}, \ell_2, \mathsf{s} \rangle \cdot \langle \mathsf{p}, \ell_1, \mathsf{q} \rangle \cdot \mathcal{M}_2 \quad \text{if } \mathsf{p} \neq \mathsf{r} \text{ or } \mathsf{q} \neq \mathsf{s}$$

Session

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

LTS for Asynchronous Sessions

Rules

[Send]

$$\mathsf{p}[\![\oplus_{i\in I} \mathsf{q}_i!\ell_i; P_i]\!] \parallel \mathbb{N} \parallel \mathcal{M} \xrightarrow{\mathsf{p}!\mathsf{q}_h.\ell_h} \mathsf{p}[\![P_h]\!] \parallel \mathbb{N} \parallel \mathcal{M}\cdot\langle \mathsf{p}, \ell_h, \mathsf{q}_h\rangle \quad h \in I$$

[Rcv]

$$\mathsf{q}[\![\Sigma_{j\in J} \mathsf{p}_j?\ell_j; Q_j]\!] \parallel \mathbb{N} \parallel \langle \mathsf{p}_h, \ell_h, \mathsf{q}\rangle\cdot\mathcal{M} \xrightarrow{\mathsf{q}?\mathsf{p}_h.\ell_h} \mathsf{q}[\![Q_h]\!] \parallel \mathbb{N} \parallel \mathcal{M} \quad h \in J$$

Example

$$P = \mathsf{q}\,!\,\ell_1; \mathsf{q}\,?\,\ell_2; P \quad Q = \mathsf{p}\,!\,\ell_2; \mathsf{p}\,?\,\ell_1; Q$$

$$\begin{aligned}
\mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \emptyset &\xrightarrow{\mathsf{p}!\mathsf{q}.\ell_1} \mathsf{p}[\![\mathsf{q}\,?\,\ell_2; P]\!] \parallel \mathsf{q}[\![\mathsf{p}\,!\,\ell_2; \mathsf{p}\,?\,\ell_1; Q]\!] \parallel \langle \mathsf{p}, \ell_1, \mathsf{q}\rangle \\
&\xrightarrow{\mathsf{q}!\mathsf{p}.\ell_2} \mathsf{p}[\![\mathsf{q}\,?\,\ell_2; P]\!] \parallel \mathsf{q}[\![\mathsf{p}\,?\,\ell_1; Q]\!] \parallel \langle \mathsf{p}, \ell_1, \mathsf{q}\rangle \cdot \langle \mathsf{q}, \ell_2, \mathsf{p}\rangle \\
&\xrightarrow{\mathsf{p}?\mathsf{q}.\ell_2} \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![\mathsf{p}\,?\,\ell_1; Q]\!] \parallel \langle \mathsf{p}, \ell_1, \mathsf{q}\rangle \\
&\xrightarrow{\mathsf{q}?\mathsf{p}.\ell_1} \mathsf{p}[\![P]\!] \parallel \mathsf{q}[\![Q]\!] \parallel \emptyset \\
&\ldots
\end{aligned}$$

Asynchronous Global Types

$$G ::=_\rho \mathsf{p}!\{\mathsf{q}_i.\ell_i; G_i\}_{i\in I} \mid \mathsf{p}?\{\mathsf{q}_i.\ell_i; G_i\}_{i\in I} \mid \mathsf{End}$$

global types are regular terms

- $\mathsf{p}!\{\mathsf{q}_i.\ell_i; G_i\}_{i\in I}$ = output choice (p sends to q_i a label ℓ_i)
- $\mathsf{p}?\{\mathsf{q}_i.\ell_i; G_i\}_{i\in I}$ = input choice (p receives from q_i label ℓ_i)
- End = termination

Example

$$G = \mathsf{p}!\mathsf{q}.\ell_1; \mathsf{q}!\mathsf{p}.\ell_2; \mathsf{p}?\mathsf{q}.\ell_2; \mathsf{q}?\mathsf{p}.\ell_1; G$$

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

$$P = \mathsf{q}\,!\,\ell_1; \mathsf{q}\,?\,\ell_2; P \qquad Q = \mathsf{p}\,!\,\ell_2; \mathsf{p}\,?\,\ell_1; Q$$

What could we do with standard global types?

$$G_1 = \mathsf{p} \to \mathsf{q}.\ell_1; \mathsf{q} \to \mathsf{p}.\ell_2; G_1 \quad \text{or} \quad G_2 = \mathsf{q} \to \mathsf{p}.\ell_2; \mathsf{p} \to \mathsf{q}.\ell_1; G_2$$

But neither of them projected on p and q is a type for $\mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!]$.
Need to add asynchronous subtyping.

Typing (1)

Typing judgement

$$\mathsf{G} \vdash \mathbb{N} \parallel \mathcal{M}$$

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

$$[\text{Out}]\ \frac{\mathsf{G}_i \vdash \mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N} \parallel \mathcal{M} \cdot \langle \mathsf{p}, \ell_i, \mathsf{q}_i \rangle \quad \forall i \in I}{\mathsf{p}!\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i \in I} \vdash \mathsf{p}[\![\bigoplus_{i \in I} \mathsf{q}_i!\ell_i; P_i \,]\!] \parallel \mathbb{N} \parallel \mathcal{M}}\ (*)$$

the side condition $(*)$ is

$$\mathsf{Players}(\mathsf{G}_i) = \mathsf{Players}(\mathsf{p}[\![\, P_i \,]\!] \parallel \mathbb{N}) \quad \forall i \in I$$

where

$$\mathsf{Players}(\mathsf{End}) = \emptyset$$

$$\mathsf{Players}(\mathsf{p}!\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i \in I}) = \mathsf{Players}(\mathsf{p}?\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i \in I}) = \{\mathsf{p}\} \cup \bigcup_{i \in I} \mathsf{Players}(\mathsf{G}_i)$$

Typing (2)

We do not want

$$\mathsf{G} \vdash \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!] \parallel \emptyset$$

where

$$P = \mathsf{q}\,!\,\ell; P \qquad Q = \mathsf{p}\,?\,\ell; Q \qquad \mathsf{G} = \mathsf{p}!\mathsf{q}.\ell; \mathsf{q}?\mathsf{p}.\ell; \mathsf{G}$$

and R could be anything.

Without the side condition

$$\dfrac{\dfrac{\dfrac{\vdots}{\mathsf{G} \vdash \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!] \parallel \emptyset}}{\mathsf{q}?\mathsf{p}.\ell; \mathsf{G} \vdash \mathsf{p}[\![\, P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\ell; Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!] \parallel \langle \mathsf{p}, \ell, \mathsf{q} \rangle}\,[\mathsf{In}]}{\mathsf{p}!\mathsf{q}.\ell; \mathsf{q}?\mathsf{p}.\ell; \mathsf{G} \vdash \mathsf{p}[\![\, \mathsf{q}\,!\,\ell; P \,]\!] \parallel \mathsf{q}[\![\, \mathsf{p}\,?\,\ell; Q \,]\!] \parallel \mathsf{r}[\![\, R \,]\!] \parallel \emptyset}\,[\mathsf{Out}]$$

Typing Rules: the Difficult Rule

$$[\mathsf{In}]\ \frac{\mathsf{G}_j \vdash \mathsf{p}[\![\, P_j \,]\!] \parallel \mathbb{N} \parallel \mathcal{M}_j \quad \iota(\mathsf{G}_j, \mathbb{M}) \quad \forall j \in J}{\mathsf{p}?\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I} \vdash \mathsf{p}[\![\, \sum_{h\in H} \mathsf{q}_h?\ell_h; P_h \,]\!] \parallel \mathbb{N} \parallel \mathcal{M}} \qquad \overset{(*)}{J = \mathsf{rm}(\{\mathsf{m}_i\}_{i\in I}, \mathcal{M}) = \mathsf{rm}(\{\mathsf{m}_h\}_{h\in H}, \mathcal{M}) \neq \emptyset}$$

- Let $\mathsf{m}_k = \langle \mathsf{q}_k, \ell_k, \mathsf{p} \rangle$.
- The indexes of ready messages of a set of messages $\{\mathsf{m}_i\}_{i\in I}$ w.r.t a queue $\mathcal{M}$ is
$$\mathsf{rm}(\{\mathsf{m}_i\}_{i\in I}, \mathcal{M}) = \{i \in I \mid \mathcal{M} \equiv \mathsf{m}_i \cdot \mathcal{M}_i\}.$$
- Branches indexed by J are live branches and the others are dead branches
- Execution of live branches should not resuscitate (unchecked) dead branches
- G is inactive for $\mathbb{M}$ (does not produce a message in $\mathbb{M}$)

$$\iota(\mathsf{End}, \mathbb{M}) \qquad \iota(\mathsf{p}?\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I}, \mathbb{M}) \quad \text{if } \iota(\mathsf{G}_i, \mathbb{M}) \ \ \forall i \in I$$
$$\iota(\mathsf{p}!\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I}, \mathbb{M}) \quad \text{if } \langle \mathsf{p}, \ell_i, \mathsf{q}_i \rangle \notin \mathbb{M} \text{ and } \iota(\mathsf{G}_i, \mathbb{M}) \ \ \forall i \in I$$

Here $\mathbb{M} = \{\langle \mathsf{q}_k, \ell_k, \mathsf{p} \rangle \mid k \in (I \cup H) \setminus J \ \&\ \mathsf{q}_k \neq \mathsf{q}_j \ \forall j \in J\}$.

We need $\iota(G, \mathbb{M})$!

Consider the multiparty session $\mathbb{N} \parallel \langle p, \ell, r \rangle$ where

$$\mathbb{N} = s[\![\, r!\ell' \,]\!] \parallel r[\![\, p?\ell; s?\ell' + s?\ell'; p?\ell' \,]\!]$$

we expect $\mathbb{N} \parallel \langle p, \ell, r \rangle$ not to be typable, since it can reach a deadlock.

$$\begin{array}{rcl} \mathbb{N} \parallel \langle p, \ell, r \rangle & \xrightarrow{s!r.\ell'} & r[\![\, p?\ell; s?\ell' + s?\ell'; p?\ell' \,]\!] \parallel \langle p, \ell, r \rangle \cdot \langle s, \ell', r \rangle \\ & \xrightarrow{r?s.\ell'} & r[\![\, p?\ell' \,]\!] \parallel \langle p, \ell, r \rangle \end{array}$$

Let $G = r?\{p.\ell; s!r.\ell'; r?s.\ell' \;,\; s.\ell'; s!r.\ell'; r?p.\ell'\}$.

$$\dfrac{\dfrac{\dfrac{}{\mathsf{End} \vdash r[\![\, 0 \,]\!] \parallel \emptyset}\,[\mathsf{In}]}{r?s.\ell' \vdash r[\![\, s?\ell' \,]\!] \parallel \langle s, \ell', r \rangle}\,[\mathsf{Out}]}{s!r.\ell'; r?s.\ell' \vdash s[\![\, r!\ell' \,]\!] \parallel r[\![\, s?\ell' \,]\!] \parallel \emptyset}\,[\mathsf{In}]}{r?\{p.\ell; s!r.\ell'; r?s.\ell' \;,\; s.\ell'; s!r.\ell'; r?p.\ell'\} \vdash \mathbb{N} \parallel \langle p, \ell, r \rangle}$$

Checking $\iota(s!r.\ell'; r?p.\ell', \{\langle s, \ell', r \rangle\})$ we fail to apply [In]

LTS for Type Configurations

$$\mathsf{G} \parallel \mathcal{M} \xrightarrow{\beta} \mathsf{G}' \parallel \mathcal{M}' \qquad \beta = \mathsf{p}!\mathsf{q}.\ell \;\text{ or }\; \beta = \mathsf{q}?\mathsf{p}.\ell$$

Rules

$$[\text{Top-Out}]\ \frac{}{\mathsf{p}!\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I} \parallel \mathcal{M} \xrightarrow{\mathsf{p}!\mathsf{q}_h.\ell_h} \mathsf{G}_h \parallel \mathcal{M} \cdot \langle \mathsf{p}, \ell_h, \mathsf{q}_h\rangle}\ h \in I$$

$$[\text{Top-In}]\ \frac{}{\mathsf{p}?\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I} \parallel \langle \mathsf{q}_h, \ell_h, \mathsf{p}\rangle \cdot \mathcal{M} \xrightarrow{\mathsf{q}_h?\mathsf{p}.\ell_h} \mathsf{G}_h \parallel \mathcal{M}}\ h \in I$$

$$[\text{Inside-Out}]\ \frac{\mathsf{G}_i \parallel \mathcal{M} \cdot \langle \mathsf{p}, \ell_i, \mathsf{q}_i\rangle \xrightarrow{\beta} \mathsf{G}_i' \parallel \mathcal{M}' \cdot \langle \mathsf{p}, \ell_i, \mathsf{q}_i\rangle \quad \forall i \in I}{\mathsf{p}!\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I} \parallel \mathcal{M} \xrightarrow{\beta} \mathsf{p}!\{\mathsf{q}_i.\ell_i; \mathsf{G}_i'\}_{i\in I} \parallel \mathcal{M}'}\ \mathsf{p} \neq \mathsf{play}(\beta)$$

$$[\text{Inside-In}]\ \frac{\mathsf{G}_j \parallel \mathcal{M} \xrightarrow{\beta} \mathsf{G}_j' \parallel \mathcal{M}' \quad \forall j \in J}{\mathsf{p}?\{\mathsf{q}_i.\ell_i; \mathsf{G}_i\}_{i\in I} \parallel \mathcal{M} \xrightarrow{\beta} \mathsf{p}?\{\mathsf{q}_i.\ell_i; \mathsf{G}_i'\}_{i\in I} \parallel \mathcal{M}'}$$

$$\begin{array}{l} \mathsf{p} \neq \mathsf{play}(\beta) \\ J = \mathsf{rm}(\{\langle \mathsf{q}_i, \ell_i, \mathsf{p}\rangle\}_{i\in I}, \mathcal{M}) \neq \emptyset \\ \beta \neq \mathsf{q}_l!\mathsf{p}.\ell_l \quad \forall l \in I \setminus J \\ \mathsf{G}_l' = \mathsf{G}_k' \quad k \in J \quad \forall l \in I \setminus J \end{array}$$

Properties

If $G \vdash \mathbb{N} \parallel \mathcal{M}$ and G is bounded (each $p \in \text{Players}(G)$ occurs at bounded depth in all paths of G), then we can prove the properties

- Session Fidelity
- Subject Reduction
- Lock Freedom

However

- The type system is undecidable.

A decidable restriction can be defined giving a (standard) inductive definition.

Related Work

- Our global types extend those of [1] by allowing input races.
- In [2] and [3] sessions are synchronous.
- None of the papers below can type the session of Example 1.
- Example 1 omitting the initial exchange between Alice and Bob can be typed in [2] and [4] but not in [3]
- Example 2 is typed in [3] but not in [2] and [4].

---

[1] F. Dagnino, M. D.-C. & P.G. (2021): Deconfined Global Types for Asynchronous Sessions.
[2] R. van Glabbeek, P. Höffner & R. Horne (2021): Assuming Just Enough Fairness to make Session Types Complete for Lock-freedom.
[3] S. Jongmans & N. Yoshida (2020): ExploringType-Level Bisimilarity towards More Expressive Multiparty Session Types.
[4] R. Majumdar, M. Mukund, F. Stutz & D. Zufferey (2021): Generalising Projection in Asynchronous Multiparty Session Types.

Conclusions and Future Work

- We defined a type system for multiparty asynchronous sessions, allowing syntactic input races and ensuring lock-freedom.
- There are several directions for future work:
  - weaken the predicate $\iota(\mathsf{G}, \mathbb{M})$ taking into account the order of messages sent by G
  - strengthen our typing in order to forbid orphan messages
  - weaken our typing in order to allow optional participants in the branches of choices, following some previous work on "connecting communications"
  - and more....


obrigado
Dank U
Merci
mahalo
Köszi
спасибо
Grazie
Thank you
mauruuru
Takk
Gracias
Dziękuję
Děkuju
danke
Kiitos