Towards Formalising Trace Equivalence
for Global and Local Types

David Castro-Perez     Francisco Ferreira     Lorenzo Gheri
Nobuko Yoshida

Imperial College London

Verification of Session Types Workshop - 4th June 2020
Certifying the Semantics of Communication

This work is part of a bigger project for certifying, and reasoning about, programs in distributed systems.

\[
\begin{align*}
\Red{G} \xrightarrow{\rho} \Red{G^c} \xrightarrow{\text{LTS}} \Red{\text{Traces}} \\
\Red{L} \xrightarrow{\rho_L} \Red{L^c} \xrightarrow{\text{LTS}} \Red{\text{Traces}}
\end{align*}
\]

- We formalize the meta-theory of multiparty session types\(^1\).
- We use the Coq\(^2\) Proof Assistant.

---


\(^2\) https://coq.inria.fr/
Global and Local Types

Inductively defined by the following syntaxes:

\[
G ::= \text{end} \quad \text{end type} \\
| X \quad \text{variable} \\
| \mu X.G \quad \text{recursion} \\
| p \to q : \{\ell_i(S_i).G_i\}_{i \in I} \quad \text{message}
\]

\[
L ::= \text{end} \quad \text{end type} \\
| X \quad \text{variable} \\
| \mu X.L \quad \text{recursion} \\
| ![q]; \{\ell_i(S_i).L_i\}_{i \in I} \quad \text{send type} \\
| ??[p]; \{\ell_i(S_i).L_i\}_{i \in I} \quad \text{receive type}
\]

Types are assumed *closed* (no free variables) and recursion variables are always assumed *guarded* in types (namely types like $\mu X.X$ are not allowed).
Projection Rules:

- \textbf{end}|r = \text{end}; [\text{PROJ-END}]
- X|r = X; [\text{PROJ-VAR}]
- (\mu X.G)|r = \mu X.(G|r) [\text{PROJ-REC}]
- r = p \text{ implies } p \rightarrow q : \{\ell_i(S_i).G_i\}_{i \in I}|r = !\{q\}; \{\ell_i(S_i).G_i|r\}_{i \in I}; [\text{PROJ-SEND}]
- r = q \text{ implies } p \rightarrow q : \{\ell_i(S_i).G_i\}_{i \in I}|r = ?\{p\}; \{\ell_i(S_i).G_i|r\}_{i \in I}; [\text{PROJ-RECV}]
- r \neq p, r \neq q \text{ and, for all } i, j \in I, G_i|r = G_j|r; \text{ implies } p \rightarrow q : \{\ell_i(S_i).G_i\}_{i \in I}|r = G_1|r; [\text{PROJ-CONT}]
- undefined otherwise.
Warm-Up!

A global type for a simple protocol:

\[ G = p \rightarrow q : \ell(S).q \rightarrow r : \ell'(S').\text{end} \]

and its projection on participant q:

\[ G|q = {?[p];\ell(S).!{r};\ell'(S')}\text{.end} \]

Do these global types have well defined projections?

\[ p \rightarrow q : \{ \ell_1(\text{oranges}).\text{end} \}, \ell_2(\text{bananas}).\text{end} \]

\[ \text{YES!} \]

\[ p \rightarrow q : \{ \ell_1(\text{oranges}).\text{end} \}, \ell_2(\text{bananas}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \]

\[ \text{NO!} \]

\[ p \rightarrow q : \{ \ell_1(\text{oranges}).p \rightarrow r : \ell_3(\text{pears}).\text{end} \}, \ell_2(\text{bananas}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \]

\[ \text{NO!} \]

\[ p \rightarrow q : \{ \ell_1(\text{oranges}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \}, \ell_2(\text{bananas}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \]

\[ \text{YES!} \]
A global type for a simple protocol:

\[ G = p \to q : \ell(S).q \to r : \ell'(S').\text{end} \]

and its projection on participant q:

\[ G|q = ?[p];\ell(S).![r];\ell'(S').\text{end} \]

Do these global types have well defined projections?

\[ p \to q : \{ \ell_1(\text{oranges}).\text{end}, \]
\[ \ell_2(\text{bananas}).q \to p : \ell_3(\text{pears}).\text{end} \} \]

\[ p \to q : \{ \ell_1(\text{oranges}).\text{end}, \]
\[ \ell_2(\text{bananas}).q \to r : \ell_3(\text{pears}).\text{end} \} \]

\[ p \to q : \{ \ell_1(\text{oranges}).p \to r : \ell_3(\text{pears}).\text{end}, \]
\[ \ell_2(\text{bananas}).q \to r : \ell_3(\text{pears}).\text{end} \} \]

\[ p \to q : \{ \ell_1(\text{oranges}).q \to r : \ell_3(\text{pears}).\text{end}, \]
\[ \ell_2(\text{bananas}).q \to r : \ell_3(\text{pears}).\text{end} \} \]
Warm-Up!

A global type for a simple protocol:

\[ G = p \rightarrow q : \ell(S).q \rightarrow r : \ell'(S').\text{end} \]

and its projection on participant q:

\[ G\upharpoonright q = ?[p];\ell(S).!r;\ell'(S').\text{end} \]

Do these global types have well defined projections?

- \[ p \rightarrow q : \{ \ell_1(\text{oranges}).\text{end}, \ell_2(\text{bananas}).q \rightarrow p : \ell_3(\text{pears}).\text{end} \} \quad \text{YES!} \]
- \[ p \rightarrow q : \{ \ell_1(\text{oranges}).\text{end}, \ell_2(\text{bananas}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \} \quad \text{NO!} \]
- \[ p \rightarrow q : \{ \ell_1(\text{oranges}).p \rightarrow r : \ell_3(\text{pears}).\text{end}, \ell_2(\text{bananas}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \} \quad \text{NO!} \]
- \[ p \rightarrow q : \{ \ell_1(\text{oranges}).q \rightarrow r : \ell_3(\text{pears}).\text{end}, \ell_2(\text{bananas}).q \rightarrow r : \ell_3(\text{pears}).\text{end} \} \quad \text{YES!} \]
Certifying the Semantics of Communication

G ↦ ρ ↦ G^c ↦ LTS ↦ Traces
L ↦ ρ_L ↦ L^c ↦ LTS ↦ Traces

- Multiparty Session Types ✓
- Coinductive Trees (equi-recursive point of view)
- Semantics by Traces
"We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.G$ and $G\{\mu X.G/X\}$."

Example: $\mu X.p \rightarrow q : \ell(S).X$ is "the same as" $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).X$), which is "the same as" $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : (\mu X.p \rightarrow q : \ell(S).X)$, ... with a coinductive unrolling process!"
An Equi-Recursive Viewpoint

“We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X.G$ and $G\{\mu X.G/X\}$.”

Example:
$\mu X.p \rightarrow q : \ell(S).X$ is “the same as” $p \rightarrow q : \ell(S). (\mu X.p \rightarrow q : \ell(S).X)$,
“We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X . G$ and $G\{\mu X . G/X\}$.”

Example:
$\mu X . p \rightarrow q : \ell(S).X$ is “the same as” $p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$, which is “the same as” $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
"We adopt the equi-recursive viewpoint, i.e., we identify $\mu X.G$ and $G{\mu X.G/X}$."

Example:
$\mu X.p \to q : \ell(S).X$ is “the same as” $p \to q : \ell(S). (\mu X.p \to q : \ell(S).X)$,
which is “the same as” $p \to q : \ell(S).p \to q : \ell(S). (\mu X.p \to q : \ell(S).X)$,
which is “the same as” $p \to q : \ell(S).p \to q : \ell(S).p \to q : \ell(S). (\mu X.p \to q : \ell(S).X)$,
An Equi-Recursive Viewpoint

“We adopt the equi-recursive viewpoint, i.e., we identify $\mu X . G$ and $G\{\mu X . G / X \}$.”

Example:
$\mu X . p \rightarrow q : \ell(S).X$ is “the same as” $p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
which is “the same as” $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
which is “the same as” $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
which is “the same as” $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : (\mu X . p \rightarrow q : \ell(S).X)$. 

David Castro-Perez, Francisco Ferreira, Lorenzo Gheri, Nobuko Yoshida Towards Formalising Trace Equivalence
An Equi-Recursive Viewpoint

“We adopt the *equi-recursive viewpoint*, i.e., we identify $\mu X . G$ and $G\{\mu X . G/X\}$.”

Example:
$\mu X . p \rightarrow q : \ell(S).X$ is “the same as” $p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
which is “the same as” $p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
which is “the same as”
$p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).(\mu X . p \rightarrow q : \ell(S).X)$,
which is “the same as”
$p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : \ell(S).p \rightarrow q : (\mu X . p \rightarrow q : \ell(S).X)$,

... with a coinductive unrolling process!
Coinductive Asynchronous Trees

Global Trees:

\[ G^c ::= \text{end}^c \quad \text{end type} \]

\[ \mid p \xrightarrow{\ell_i} q : \{\ell_i(S_i).G^c_i\}_{i \in I} \quad \text{message send} \]

\[ \mid p \xleftarrow{\ell_i} q : \{\ell_i(S_i).G^c_i\}_{i \in I} \quad \text{message receive} \]

Coinductive unrolling \( \rho \):

\[ \begin{align*}
\text{end} \rho \text{end}^c & \quad \mu X.G \rho G^c \\
G\{\mu X.G/X\} & \quad G^c
\end{align*} \]

\[ \forall i \in I. G_i \rho G^c_i \]

\[ p \rightarrow q : \{\ell_i(S_i).G_i\}_{i \in I} \rho p \rightarrow q : \{\ell_i(S_i).G^c_i\}_{i \in I} \]
Coinductive Asynchronous Trees

Global Trees:

\[
\begin{align*}
G^c & ::= \text{end}^c & \text{end type} \\
& | \quad p \to q : \{ \ell_i(S_i).G^c_i \}_{i \in I} & \text{message send} \\
& | \quad p \xrightarrow{\ell_i} q : \{ \ell_i(S_i).G^c_i \}_{i \in I} & \text{message receive}
\end{align*}
\]

Local Trees:

\[
\begin{align*}
L^c & ::= \text{end}^c & \text{end type} \\
& | \quad !^c[p]; \{ \ell_i(S_i).L^c_i \}_{i \in I} & \text{send type} \\
& | \quad ?^c[q]; \{ \ell_i(S_i).L^c_i \}_{i \in I} & \text{receive type}
\end{align*}
\]
Certifying the Semantics of Communication

- Multiparty Session Types ✓
- Coinductive Trees (equi-recursive point of view) ✓
- Semantics by Traces
p sends a message to q with label \( \ell \):

\[ p \rightarrow q : \ell(S).G^c \]

We keep track of this communication with a queue:

\[ \epsilon \]
Asynchronous Semantics

$p$ sends a message to $q$ with label $\ell$:

$$p \rightarrow q : \ell(S).G^c \xrightarrow{\text{step } \ell} p \leadsto q : \ell(S).G^c$$

We keep track of this communication with a queue:

$$\epsilon \xrightarrow{\text{enqueue}} [(\ell, S)]$$
p sends a message to q with label ℓ:

\[ p \rightarrow q : \ell(S).G^c \quad \text{step } \ell \quad p \rightsquigarrow q : \ell(S).G^c \quad \text{step } \ell \quad G^c \]

We keep track of this communication with a queue:

\[ \epsilon \xrightarrow{\text{enqueue}} [(\ell, S)] \xrightarrow{\text{dequeue}} \epsilon \]
Asynchronous Semantics

p sends a message to q with label \( \ell \):

\[
p \rightarrow q : \ell(S).G^c \xrightarrow{\text{step } \ell} p \leadsto q : \ell(S).G^c \xrightarrow{\text{step } \ell} G^c
\]

We keep track of this communication with a queue:

\[
\varepsilon \xrightarrow{\text{enqueue}} [(\ell, S)] \xrightarrow{\text{dequeue}} \varepsilon
\]

We have such a queue for each pair of participants.
p sends and q receive:

\[ !^c[q];\ell(S).L^c \quad \xrightarrow{\text{step}} \quad L^c \]

\[ Q(p, q) = \epsilon \quad \xrightarrow{\text{enqueue}} \quad Q(p, q) = [(\ell, S)] \quad \xrightarrow{\text{dequeue}} \quad Q(p, q) = \epsilon \]

\[ ?^c[p];\ell(S).L^{c'} \quad \xrightarrow{\text{step}} \quad L^{c'} \]

**Queue Environments**

A *queue environment* is a finitely supported function \( Q \) of type \( \text{role} \times \text{role} \to W \), where \( W \) is the set of finite words \( w \) (queues) on the alphabet \( \text{labels} \times \text{sorts} \).
Global and Local Steps

p sends:

\[ p \rightarrow q : \ell(S).G^c \xrightarrow{\text{step}_1} p \sim q : \ell(S).G^c \xrightarrow{\text{step}_2} G^c \]

\[ !^c[q]; \ell(S).L^c \xrightarrow{\text{step}_1} L^c \]
Global and Local Steps

p sends:

\[ p \to q : \ell(S).G \]

\[ \text{step}_1 \]

\[ \vdash_p \]

\[ !^c[q]; \ell(S).L \]

\[ \text{step}_1 \]

q receives:

\[ p \to q : \ell(S).G \]

\[ \text{step}_2 \]

\[ \vdash_q \]

\[ ?^c[p]; \ell(S).L' \]

\[ \text{step}_2 \]
We want to consider altogether the different local types involved in the communication.

**Environments for Local Types**

An *environment for local types* is a finitely supported function $E$ of type \( \text{role} \rightarrow \text{l\_ty}^c \).
We want to consider altogether the different local types involved in the communication.

**Environments for Local Types**

An *environment for local types* is a finitely supported function $E$ of type $\text{role} \rightarrow \text{l}_{\text{ty}}^c$.

Why?!
We want to consider altogether the different local types involved in the communication.

**Environments for Local Types**

An *environment for local types* is a finitely supported function $E$ of type $\text{role} \rightarrow \text{l}_\text{ty}^c$.

Why?!

It will be $E(p) = L^c_p$, where $G^c \models_L^c L^c_p$. 
We get the one-shot projection of a global type both on environments of local types and on queue environments.
Step Results

Theorem (Step Soundness)

If $G^c \parallel (E, Q)$ and $G^c \xrightarrow{\text{step}} G^c'$, then there exist $E'$ and $Q'$, such that $G^c' \parallel (E', Q')$ and $(E, Q) \xrightarrow{\text{step}} (E', Q')$.

Theorem (Step Completeness)

If $G^c \parallel (E, Q)$ and $(E, Q) \xrightarrow{\text{step}} (E', Q')$, then there exists $G^c'$, such that $G^c' \parallel (E', Q')$ and $G^c \xrightarrow{\text{step}} G^c'$.
Labelled Transition System

Keeping track of messages...

Actions
An *action* $a$ is an object of the shape:
- either $pq!(\ell, S)$ (send action),
- or $pq?\!(\ell, S)$ (receive action).

Traces
A *trace* $t$ is a coinductive, possibly infinite, stream of actions:
- $a_1a_2a_3\ldots$ is a trace.
p sends and q receives:

\[ p \rightarrow q : \ell(S).G^c \xrightarrow{\text{step}_1} p \sim \ell q : \ell(S).G^c \xrightarrow{\text{step}_2} G^c \]

Let \( t \) be a trace for \( G^c \),

\[ pq!(\ell, S)pq?(\ell, S)t \xleftarrow{\text{step}_1} pq?!(\ell, S)t \xleftarrow{\text{step}_2} t \]
Non-Determinism

Let us consider two different executions:

\[ p \rightarrow q : \{ \ell_i(S_i).G^c_i \}_{i \in I} \xrightarrow{\text{step } \ell_1} p \xrightarrow{\ell_1} q : \{ \ell_i(S_i).G^c_i \}_{i \in I} \xrightarrow{\text{step } \ell_1} G^c_1 \]

\[ p \rightarrow q : \{ \ell_i(S_i).G^c_i \}_{i \in I} \xrightarrow{\text{step } \ell_2} p \xrightarrow{\ell_2} q : \{ \ell_i(S_i).G^c_i \}_{i \in I} \xrightarrow{\text{step } \ell_2} G^c_2 \]

If \( t_1 \) is a trace admissible for \( G^c_1 \) and \( t_2 \) is admissible for \( G^c_2 \), both

\[ pq!(\ell_1, S_1)pq?(\ell_1, S_1)t_1 \]

and

\[ pq!(\ell_2, S_2)pq?(\ell_2, S_2)t_2 \]

are admissible for \( p \rightarrow q : \{ \ell_i(S_i).G^c_i \}_{i \in I} \).
And finally...

Theorem (Trace Equivalence)

If \( G_c \upharpoonright c \in E \), then the set of traces admissible for \( G_c \) is equal to the set of traces admissible for \( E \).
And finally...

**Theorem (Trace Equivalence)**

If $G^c \models^c E$ then the set of traces admissible for $G^c$ is equal to the set of traces admissible for $E$. 
Certifying the Semantics of Communication

- Multiparty Session Types ✓
- Coinductive Trees (equi-recursive point of view) ✓
- Semantics by Traces ✓ (almost)
Things We Used

Formalisation in the Coq\textsuperscript{3} Proof Assistant, in particular we have used:

- the SSReflect\textsuperscript{4} proof language;
- the Mathematical Components\textsuperscript{5} libraries;
- the PaCo library for parametrized coinduction\textsuperscript{6}.

\textsuperscript{3}https://coq.inria.fr/
\textsuperscript{4}https://coq.inria.fr/refman/proof-engine/ssreflect-proof-language.html
\textsuperscript{5}https://math-comp.github.io/
\textsuperscript{6}https://github.com/snu-sf/paco
## Conclusion

### Things we have got:
- a formalisation of the metatheory of multiparty session types in Coq
- two birds with a (coinductive) stone: equi-recursion and no bindings
- (non-deterministic) semantics through labelled transition systems
- types that are ready for typing!

### From here what?
- a more comprehensive version of MPST (e.g., with merge operator)
- communicating finite state automata
- ... the future is unwritten!
Conclusion

Things we have got:
- a formalisation of the metatheory of multiparty session types in Coq
- two birds with a (coinductive) stone: equi-recursion and no bindings
- (non-deterministic) semantics through labelled transition systems
- types that are ready for typing! Please attend Francisco’s talk! :)

From here what?
- a more comprehensive version of MPST (e.g., with merge operator)
- communicating finite state automata
- ... the future is unwritten!

Thank You!

David Castro-Perez, Francisco Ferreira, Lorenzo Gheri, Nobuko Y
Towards Formalising Trace Equivalence