A Gentle Adventure Mechanising Message Passing Concurrency Systems

Formalising the Metatheory for smol-Zooid

David Castro-Perez, Francisco Ferreira, Lorenzo Gheri, and Nobuko Yoshida
The MPST World, as We Know It

K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. POPL '08
Introducing the Metatheory of smol-Zooid Types

Simple, but significant multiparty session type metatheory!

[Diagram]

Embark on our Gentle Adventure!!! https://github.com/emtst/GentleAdventure
Formalisation of Global and Local Types

Inductively Defined Datatypes

\[ G ::= \text{end} \mid X \mid \mu X.G \mid p \rightarrow q : (S).G \]

Coinductively Defined Datatypes

\[ G^c ::= \text{end}^c \mid p \rightarrow q : (S).G^c \mid p \rightsquigarrow q : (S).G^c \]

Local Types

\[ L ::= \text{end} \mid X \mid \mu X.L \mid ![q]; (S).L \mid ?[p]; (S).L \]

Coinductive Local Types

\[ L^c ::= \text{end}^c \mid ![^c][p]; (S).L^c \mid ?[^c][q]; (S).L^c \]
Formalisation of Global and Local Types

\[ G = \mu X. p \rightarrow q : (S).X \]

\[ \uparrow \]

\[ G \mid_p = \mu X. ![q]; (S).X \]

\[ G \mid_q = \mu X. ?[p]; (S).X \]

\[ \Downarrow \]

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

\[ \Downarrow \]

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

\[ L^c_q = ?^c[p]; (S).L^c_q \]

with \( G^c \mid_p L^c_p \)

and \( G^c \mid_q L^c_q \)
Theorem (Unravelling preserves projections)

Given \( G, L, G^c \) and \( L^c \), such that

(a) \( G \upharpoonright r = L \)
(b) \( G \mathcal{R} G^c \)
(c) \( L \mathcal{R} L^c \)

then \( G^c \upharpoonright^c r L^c \).

Proof.

By coinduction. :)

The Paco Library for Coq: https://plv.mpi-sws.org/paco/
Type Semantics for Zooid

$G^c \xrightarrow{\text{LTS}} \text{global trace} \downarrow \downarrow \uparrow$

$|^c$

$L^c \xrightarrow{\text{LTS}} \text{local trace}$
With Love, from $p$ to $q$

$p$ sends:

$$p \rightarrow q : (S).G^c \xrightarrow{!pqS} p \rightsquigarrow q : (S).G^c \xrightarrow{?qpS} G^c$$

$$!_p: G^c$$

$$!^{c}[q]; (S).L^c$$

$q$ receives:

$$p \rightarrow q : (S).G^c \xrightarrow{!pqS} p \rightsquigarrow q : (S).G^c \xrightarrow{?qpS} G^c$$

$$?_q: G^c$$

$$?^{c}[p]; (S).L^{c'}$$
Tools for our LTS

**Actions.** \(!pqS\) and \(?qpS\)

**(Local) Environments.** \(E\) such that, \(E(p) = L^{c_p}\) where \(G^c \uparrow^c p L^{c_p}\)

**Queues and Queue Environments.** \(Q\), buffers for asynchronous communication.

\[
\begin{align*}
!^c[q];(S) . L^c & \xrightarrow{\text{step}} L^c \\
Q(p, q) &= [] & Q(p, q) &= [S] & Q(p, q) &= [] \\
?^c[p];(S) . L^{c'} & \xrightarrow{\text{step}} L^{c'}
\end{align*}
\]
Tools for our LTS

**Actions.** \(!pqS\) and \(?qpS\)

**Local Environments.** \(E\) such that, \(E(p) = L^c_p\) where \(G^c \uparrow^c p L^c_p\)

**Queues and Queue Environments.** \(Q\), buffers for asynchronous communication.

\[
\begin{align*}
!^c[q](S).L^c & \xrightarrow{\text{step}} L^c \\
Q(p, q) = [] & \xrightarrow{\text{enqueue}} Q(p, q) = [S] & \xrightarrow{\text{dequeue}} Q(p, q) = [] \\
?^c[p](S).L^{c'} & \xrightarrow{\text{step}} L^{c'}
\end{align*}
\]
Theorem (Step Soundness)
If $G^c \xrightarrow{a} G'^c$ and $G^c \models (E, Q)$, there exist $E'$ and $Q'$ such that $G'^c \models (E', Q')$ and $(E, Q) \xrightarrow{a} (E', Q')$.

Theorem (Step Completeness)
If $(E, Q) \xrightarrow{a} (E', Q')$ and $G^c \models (E, Q)$, there exist $G'^c$ such that $G'^c \models (E', Q')$ and $G^c \rightarrow G'^c$.

Theorem (Trace equivalence)
If $G^c \models (E, Q)$, then $\text{tr}^g t G^c$ if and only if $\text{tr}^t t(E, Q)$.
Lemma, to give the flavour

$p \rightarrow q : (S).G^c$ \implies $p \rightsquigarrow q : (S).G^c$

$!pq_s$ \quad $\vdash_p$ \quad $\vdash_p$

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

$p \rightarrow q : (S).G^c$ \implies $p \rightsquigarrow q : (S).G^c$

$\vdash_q$ \quad $\vdash_q$

$?^c[p]; (S).L^{c'}$
Our Adventurer Rests and Meditates

- Formal proofs are not easy! (But useful and fun!)
- Proof design is the key.
- Proof techniques are to be taken seriously: (co)induction, functions VS relations...
Our Adventurer Rests and Meditates

• Formal proofs are not easy! (But useful and fun!)
• Proof design is the key.
• Proof techniques are to be taken seriously: (co)induction, functions VS relations...

“You need to stay focused. Otherwise you miss the subtleties!”

1 Barney Greenway (Napalm Death), after surprising the audience with a blitz performance of “You Suffer”.
Future

- Adding Features for Reasoning about Processes
- Certifying Existing Systems (e.g., integration with $\nu$Scr)
- Moving Further towards Coinduction (e.g., Interaction Trees)
- Hoping for New People and Collaborations :)

Check out our material!

website: http://mrg.doc.ic.ac.uk/publications/zooid-paper/

→ This tutorial is available at https://github.com/emtst/GentleAdventure

Thank You!