Automatising Proofs for Multiparty Session Types

Kirstin Peters

TU Darmstadt – Theory of Parallel Systems

2020, June 05

Toy Example – Processes

$$P ::= s[r_1, r_2]!\langle y \rangle.P \mid s[r_2, r_1]?(x).P \mid P \mid_{\mathfrak{P}} P \mid \mathbf{0}$$

I did not use any binders.

$$\text{Com}\ \frac{}{s[r_1, r_2]!\langle y \rangle.P_1 \mid_{\mathfrak{P}} s[r_2, r_1]?(x).P_2 \longmapsto P_1 \mid_{\mathfrak{P}} P_2\{\!|x \triangleright y|\!\}}$$

$$\text{Par}\ \frac{P_1 \longmapsto P_1'}{P_1 \mid_{\mathfrak{P}} P_2 \longmapsto P_1' \mid_{\mathfrak{P}} P_2}$$

Toy Example – Types

$$G ::= r_1 \to r_2{:}S.G \mid \mathsf{end}_{\mathfrak{G}}$$

$$L ::= [r]!\langle S \rangle.L \mid [r]?\langle S \rangle.L \mid \mathsf{end}_{\mathfrak{L}}$$

- rolesGT$(\cdot)$
- projection: $G \upharpoonright r$

Toy Example – Typing Rules

$$\Gamma ::= n{:_{\mathfrak{E}}}S \qquad\qquad \Delta ::= s[r]{:}L$$

- free_in_GE $n\ \Gamma$, linearGE Γ
- free_in_SE $s\ r\ \Delta$, linearSE Δ

$$\text{Send}\ \frac{y{:_{\mathfrak{E}}}S \in \Gamma \quad \Gamma \vdash P \triangleright \Delta \cup \{s[r_1]{:}L\} \quad \ldots}{\Gamma \vdash s[r_1, r_2]!\langle y \rangle.P \triangleright \Delta \cup \{s[r_1]{:}[r_2]!\langle S \rangle.L\}}$$

$$\text{Get}\ \frac{\Gamma \cup \{x{:_{\mathfrak{E}}}S\} \vdash P \triangleright \Delta \cup \{s[r_1]{:}L\} \quad \ldots}{\Gamma \vdash s[r_1, r_2]?\langle y \rangle.P \triangleright \Delta \cup \{s[r_1]{:}[r_2]?\langle S \rangle.L\}}$$

$$\text{Par}\ \frac{\Delta = \Delta_1 \cup \Delta_2 \quad \Gamma \vdash P_1 \triangleright \Delta_1 \quad \Gamma \vdash P_2 \triangleright \Delta_2 \quad \ldots}{\Gamma \vdash P_1 |_{\mathfrak{P}} P_2 \triangleright \Delta}$$

$$\text{End}\ \frac{\ldots}{\Gamma \vdash \mathbf{0} \triangleright \{\}}$$

Theorem (Inversion Lemma)

If $\Gamma \vdash s[r_1, r_2]!\langle y\rangle.P \triangleright \Delta$, then there are some S, L such that
$s[r_1]{:}[r_2]!\langle S\rangle.L \in \Delta, \quad y{:}_{\mathfrak{e}}S \in \Gamma,$
$\Gamma \vdash P \triangleright (\Delta - \{s[r_1]{:}[r_2]!\langle S\rangle.L\}) \cup \{s[r_1]{:}L\}$, and ...
...

Theorem (Substitution Lemma)

If $\Gamma \cup \{x{:}_{\mathfrak{e}}S\} \vdash P \triangleright \Delta$, ..., and $y{:}_{\mathfrak{e}}S \in \Gamma$, then $\Gamma \vdash P\{\!|x \triangleright y|\!\} \triangleright \Delta$.

$$\begin{aligned}&\text{weakly\_coherent } \Delta\\ &\equiv \exists G\ s.\ \Delta \subseteq \{X \mid \exists r.\ r \in \text{rolesGT}(G) \wedge X = s[r]{:}(G \restriction r)\}\end{aligned}$$

Theorem (Subject Reduction)

If $\Gamma \vdash P \triangleright \Delta$, Δ is weakly coherent, and $P \longmapsto P'$,
then there is some Δ' such that $\Gamma \vdash P \triangleright \Delta'$.

Theorem (Subject Reduction)

If $\Gamma \vdash P \triangleright \Delta$, Δ *is weakly coherent, and* $P \longmapsto P'$*,*
then there is some Δ' *such that* $\Gamma \vdash P \triangleright \Delta'$*.*

Proof Strategy:

- the proof is by induction on the $\longmapsto$-rules
- for $\longmapsto$-axioms:
  - use the *inversion* lemma to obtain type information from $\Gamma \vdash P \triangleright \Delta$ in the concrete $\longmapsto$-case
  - if necessary, use coherence to combine information on parallel parts
  - if necessary, use the *substitution* lemma to get rid of additional names
  - use the typing rules to construct the proof tree for the derivative
  - if necessary, exploit linearity (deconstruction using inversion) and re-establish linearity (construction using the typing rules)
- for non-axioms:
  - similar to before (though usually easier), but additionally you have the induction hypothesis

Small Case Study

in total: > 40 kB and > 850 lines of code

	lines of code	%
model	≈ 115	$\approx 13\%$
subject reduction	≈ 70	$\approx 8\%$
inversion lemma	≈ 170	$\approx 19\%$
substitution lemma	≈ 85	$\approx 10\%$
linear environments	≈ 445	$\approx 50\%$

paper proof

without binders!

Larger Case Study

larger (and more useful) model with binders (nominal sets)
but no proofs (except for the inversion lemmata), no subject reduction

in total: $> 1,3$ MB and > 19.500 lines of code

	lines of code	%
model	≈ 4764	$\approx 24\%$
inversion lemma	≈ 2042	$\approx 10\%$
linear environments	≈ 7642	$\approx 39\%$
binders	≈ 5320	$\approx 27\%$

paper proof

Challenges:

- **implementation of binders**
- **linearity of type environments**

Ideas:

- binders using nominal sets
  - works fine (at least for me)
  - but requires a lot of boring auxiliary results
  - needs more automation
- linearity of type environments
  - lists or sets are not a good idea
  - maybe use a structure that ensures linearity by design
  - still you might need algorithmic support
- algorithmic support for different parts of the proofs
  - deconstructing typing proofs and
    constructing typing proofs from partial proof trees
  - structural congruence