Marco Carbone, David Castro-Perez, Francisco Ferreira,
Lorenzo Gheri, Frederik Krogsdal Jacobsen, Alberto Momigliano,
Luca Padovani, Alceste Scalas, Dawit Tirore, Martin Vassor,
Nobuko Yoshida, Daniel Zackon

The Concurrent Calculi Formalisation Benchmark

at Logic & AI @ AlgoLoG

Introduction

We want (everyone) to mechanise concurrent systems!

Introduction

We want (everyone) to mechanise concurrent systems!

Proof assistants are (fun and) useful:

- certified code generation
- no mistakes in overlooked cases
- new insights

Introduction

We want (everyone) to mechanise concurrent systems!

Proof assistants are (fun and) useful:

- certified code generation
- no mistakes in overlooked cases
- new insights

Realisation: mechanising concurrent systems is a big effort.

The Benchmark Approach

Mixing concurrent calculi with the POPLMark spirit!

We want to encourage:

- comparison of different approaches
- the development of guidelines, tutorials, techniques, libraries...
- reusability

https://concurrentbenchmark.github.io/

The Benchmark Approach

Mixing concurrent calculi with the POPLMark spirit!

We want to encourage:

- comparison of different approaches
- the development of guidelines, tutorials, techniques, libraries...
- reusability

Three fundamental challenges on concurrency and session types:

1. linearity and behavioural type systems
2. name passing and scope extrusion
3. coinduction and infinite processes

`https://concurrentbenchmark.github.io/`

Linearity and behavioural type systems

Linearity and behavioural type systems

Processes:

$$
\begin{array}{rclcl}
v, w & ::= & a & \mid & l \\
P, Q & ::= & \mathbf{0} & \mid & x!v.P \quad \mid \quad x?(l).P \quad \mid \quad (P \mid Q) \quad \mid \quad (\nu xy)\, P
\end{array}
$$

Linearity and behavioural type systems

Processes:

$$
\begin{array}{lcllllllllll}
v, w & ::= & a & \mid & l \\
P, Q & ::= & \mathbf{0} & \mid & x!v.P & \mid & x?(l).P & \mid & (P \mid Q) & \mid & (\nu xy)\, P
\end{array}
$$

Semantics:

$$\textsc{R-Com}\quad \frac{}{(\nu xy)\,(x!a.P \mid y?(l).Q \mid R) \rightarrow (\nu xy)\,(P \mid Q\{a/l\} \mid R)}$$

$$\textsc{R-Res}\quad \frac{P \rightarrow Q}{(\nu xy)\, P \rightarrow (\nu xy)\, Q}$$

$$\textsc{R-Par}\quad \frac{P \rightarrow Q}{P \mid R \rightarrow Q \mid R}$$

$$\textsc{R-Struct}\quad \frac{P \equiv P' \qquad P' \rightarrow Q' \qquad Q \equiv Q'}{P \rightarrow Q}$$

Linearity and behavioural type systems

1. No endpoint is used simultaneously by parallel processes.
2. The two endpoints of the same session have dual types.

Linearity and behavioural type systems

1. No endpoint is used simultaneously by parallel processes.
2. The two endpoints of the same session have dual types.

Types:

$$
\begin{array}{rcl}
S, T & ::= & \textbf{end} \mid \textbf{base} \mid ?.S \mid !.S \\
\Gamma & ::= & \cdot \mid \Gamma, l \qquad \Delta ::= \cdot \mid \Delta, x : S
\end{array}
$$

Typing rules:

$$
\textsc{T-Inact}\ \frac{\textbf{end}(\Delta)}{\Gamma; \Delta \vdash \mathbf{0}}
\qquad
\textsc{T-Par}\ \frac{\Gamma; \Delta_1 \vdash P \qquad \Gamma; \Delta_2 \vdash Q}{\Gamma; \Delta_1, \Delta_2 \vdash P \mid Q}
\qquad
\textsc{T-Res}\ \frac{\Gamma; (\Delta, x : T, y : \overline{T} \vdash P)}{\Gamma \vdash (\nu xy)\ P}
$$

$$
\textsc{T-Out}\ \frac{\Gamma \vdash_v v : \textbf{base} \qquad \Gamma; \Delta, x : T \vdash P}{\Gamma; (\Delta, x : !.T) \vdash x!v.P}
\qquad
\textsc{T-In}\ \frac{(\Gamma, l); (\Delta, x : T) \vdash P}{\Gamma; (\Delta, x : ?.T) \vdash x?(l).P}
$$

Linearity and behavioural type systems

Challenge:

Theorem (Subject reduction)

If $\Gamma; \Delta \vdash P$ *and* $P \to Q$ *then* $\Gamma; \Delta \vdash Q$.

Linearity and behavioural type systems

Challenge:

Theorem (Subject reduction)

If $\Gamma; \Delta \vdash P$ and $P \to Q$ then $\Gamma; \Delta \vdash Q$.

Theorem (Type safety)

If $\Gamma; \cdot \vdash P$, then P is well formed.

In particular, no $(\nu x x')\ (x!v.P \mid x'!v'.P')$.

Name passing and scope extrusion

Name passing and scope extrusion

Processes:

$$P, Q := \mathbf{0} \quad | \quad (P \mid Q) \quad | \quad x!y.P \quad | \quad x?(y).P \quad | \quad (\nu x)\ P$$

Name passing and scope extrusion

Processes:

$$P, Q := \mathbf{0} \quad | \quad (P \mid Q) \quad | \quad x!y.P \quad | \quad x?(y).P \quad | \quad (\nu x)\ P$$

One relevant example:

$$((\nu y)\ x!y.P) \mid (x?(z).Q)$$

Name passing and scope extrusion

First approach: structural congruence and reduction.

$$((\nu y)\ x!y.P) \mid (x?(z).Q)$$

Name passing and scope extrusion

First approach: structural congruence and reduction.

$$((\nu y)\ x!y.P) \mid (x?(z).Q) \quad \equiv$$
$$(\nu y)\ (x!y.P \mid x?(z).Q)$$

$$\textsc{Sc-Res-Par}\quad \frac{x \notin \mathrm{fn}(Q)}{(\nu x)\ P \mid Q \equiv (\nu x)\ (P \mid Q)}$$

Name passing and scope extrusion

First approach: structural congruence and reduction.

$$\begin{array}{ll} ((\nu y)\ x!y.P) \mid (x?(z).Q) & \equiv \\ (\nu y)\ (x!y.P \mid x?(z).Q) & \to \\ (\nu y)\ (P \mid Q\{y/z\}) & \end{array}$$

$$\dfrac{\text{R-COM}}{x!y.P \mid x?(z).Q \to P \mid Q\{y/z\}} \quad \text{and} \quad \dfrac{\overset{\text{R-RES}}{P \to Q}}{(\nu x)\ P \to (\nu x)\ Q}$$

Name passing and scope extrusion

Second approach: labelled transition system.

$$((\nu y)\ x!y.P) \mid (x?(z).Q)$$

Name passing and scope extrusion

Second approach: labelled transition system.

$$((\nu y)\ x!y.P) \mid (x?(z).Q) \quad \xrightarrow{\tau} \quad (\nu y)\ (P \mid Q\{y/z\})$$

$$\dfrac{\dfrac{x!y.P \xrightarrow{x!y} P \qquad x \neq y}{(\nu y)\ x!y.P \xrightarrow{x!(y)} P} \qquad x?(z).Q \xrightarrow{x?y} Q\{y/z\} \qquad z \notin \mathrm{fn}(Q)}{((\nu y)\ x!y.P) \mid (x?(z).Q) \xrightarrow{\tau} (\nu y)\ (P \mid Q\{y/z\})}$$

Name passing and scope extrusion

Second approach: labelled transition system.

$$((\nu y)\ x!y.P) \mid (x?(z).Q) \quad \xrightarrow{\tau} \quad (\nu y)\ (P \mid Q\{y/z\})$$

$$\dfrac{\dfrac{x!y.P \xrightarrow{x!y} P \qquad x \neq y}{(\nu y)\ x!y.P \xrightarrow{x!(y)} P} \qquad x?(z).Q \xrightarrow{x?y} Q\{y/z\} \qquad z \notin \mathrm{fn}(Q)}{((\nu y)\ x!y.P) \mid (x?(z).Q) \xrightarrow{\tau} (\nu y)\ (P \mid Q\{y/z\})}$$

Close-L

$$\dfrac{P \xrightarrow{x!(z)} P' \qquad Q \xrightarrow{x?z} Q' \qquad z \notin \mathrm{fn}(Q)}{P \mid Q \xrightarrow{\tau} (\nu z)\ P' \mid Q'}$$

Name passing and scope extrusion

Second approach: labelled transition system.

$$((\nu y)\ x!y.P) \mid (x?(z).Q) \quad \xrightarrow{\tau} \quad (\nu y)\ (P \mid Q\{y/z\})$$

$$\dfrac{\dfrac{x!y.P \xrightarrow{x!y} P \qquad x \neq y}{(\nu y)\ x!y.P \xrightarrow{x!(y)} P} \qquad x?(z).Q \xrightarrow{x?y} Q\{y/z\} \qquad z \notin \mathrm{fn}(Q)}{((\nu y)\ x!y.P) \mid (x?(z).Q) \xrightarrow{\tau} (\nu y)\ (P \mid Q\{y/z\})}$$

OPEN

$$\dfrac{P \xrightarrow{x!z} P' \qquad z \neq x}{(\nu z)\ P \xrightarrow{x!(z)} P'}$$

Name passing and scope extrusion

Second approach: labelled transition system.

$$((\nu y)\; x!y.P) \mid (x?(z).Q) \quad \xrightarrow{\tau} \quad (\nu y)\; (P \mid Q\{y/z\})$$

$$\dfrac{\dfrac{\boxed{x!y.P \xrightarrow{x!y} P} \qquad x \neq y}{\boxed{(\nu y)\; x!y.P \xrightarrow{x!(y)} P}} \qquad \boxed{x?(z).Q \xrightarrow{x?y} Q\{y/z\}} \qquad z \notin \mathrm{fn}(Q)}{\boxed{((\nu y)\; x!y.P) \mid (x?(z).Q) \xrightarrow{\tau} (\nu y)\; (P \mid Q\{y/z\})}}$$

$$\textsc{Out}\;\dfrac{}{x!y.P \xrightarrow{x!y} P} \quad \text{and} \quad \textsc{In}\;\dfrac{}{x?(z).P \xrightarrow{x?y} P\{y/z\}}$$

Name passing and scope extrusion

Challenge:

Theorem

$P \xrightarrow{\tau} Q$ implies $P \rightarrow Q$.

Theorem

$P \rightarrow Q$ implies the existence of a Q' such that $P \xrightarrow{\tau} Q'$ and $Q \equiv Q'$.

Coinduction and infinite processes

Coinduction and infinite processes

Describing the behaviour of recursive loops in programs.

$$
\begin{array}{lcl}
v, w & ::= & a \mid l \\
P, Q & ::= & \mathbf{0} \mid x!v.P \mid x?(l).P \mid (P \mid Q) \mid (\nu x)\ P \mid !P
\end{array}
$$

Coinduction and infinite processes

Describing the behaviour of recursive loops in programs.

$$\begin{array}{lcl} v, w & ::= & a \mid l \\ P, Q & ::= & \mathbf{0} \mid x!v.P \mid x?(l).P \mid (P \mid Q) \mid (\nu x)\, P \mid \;!P \end{array}$$

$$\textsc{Rep}\ \frac{P \xrightarrow{\alpha} P'}{!P \xrightarrow{\alpha} P' \mid !P}$$

$$\alpha \ ::= \ x!a \mid x?a \mid \tau$$

Coinduction and infinite processes

Observability predicate:

$$P \downarrow_{x?} \quad \text{if } P \text{ can perform an input action via } x.$$
$$P \downarrow_{x!} \quad \text{if } P \text{ can perform an output action via } x.$$

Strong barbed bisimilarity:
the *largest* symmetric relation such that, whenever $P \stackrel{\bullet}{\sim} Q$:

$$P \downarrow_{\mu} \text{ implies } Q \downarrow_{\mu} \tag{1}$$
$$P \xrightarrow{\tau} P' \text{ implies } Q \xrightarrow{\tau} \stackrel{\bullet}{\sim} P' \tag{2}$$

Coinduction and infinite processes

Observability predicate:

$$P \downarrow_{x?} \quad \text{if } P \text{ can perform an input action via } x.$$
$$P \downarrow_{x!} \quad \text{if } P \text{ can perform an output action via } x.$$

Strong barbed bisimilarity:
the *largest* symmetric relation such that, whenever $P \stackrel{\bullet}{\sim} Q$:

$$P \downarrow_\mu \text{ implies } Q \downarrow_\mu \tag{1}$$
$$P \xrightarrow{\tau} P' \text{ implies } Q \xrightarrow{\tau} \stackrel{\bullet}{\sim} P' \tag{2}$$

Equivalence, but NOT A CONGRUENCE: $x!a.y!b.\mathbf{0} \stackrel{\bullet}{\sim} x!a.\mathbf{0}$, but in the context $C = [\cdot] \mid x?(l).\mathbf{0}$, $x!a.y!b.\mathbf{0} \mid x?(l).\mathbf{0} \stackrel{\bullet}{\not\sim} x!a.\mathbf{0} \mid x?(l).\mathbf{0}$.

Coinduction and infinite processes

Strong barbed congruence:
$P \simeq^c Q$, if $C[P] \stackrel{\bullet}{\sim} C[Q]$ for every context C.

Lemma

$\simeq^c$ is the largest congruence included in $\stackrel{\bullet}{\sim}$.

Coinduction and infinite processes

Strong barbed congruence:
$P \simeq^c Q$, if $C[P] \stackrel{\bullet}{\sim} C[Q]$ for every context C.

Lemma

$\simeq^c$ is the largest congruence included in $\stackrel{\bullet}{\sim}$.

Challenge:

Theorem

$P \simeq^c Q$ if, for any process R and substitution σ, $P\sigma \mid R \stackrel{\bullet}{\sim} Q\sigma \mid R$.

Was this tedious? :)

Was this tedious? :)

Repeating the mechanisation effort for the basics definitely is.

Was this tedious? :)

Repeating the mechanisation effort for the basics definitely is.

A community effort towards:

- tutorial formalisations for different approaches
- comparing different approaches
- establishing “best practices”
- investigating strengths and weaknesses of proof assistants
- suggesting and developing new features of proof assistants

Why contribute and how to get involved

WHY:

- relevance and interest (solving your problems and other people's)
- connecting different parts of the community
- conducting your own mechanisation
- publication, both experience reports/tutorials and novelties
- learn a new proof assistant with cool features

HOW:

`https://concurrentbenchmark.github.io/`

`https://groups.google.com/g/concurrentbenchmark`

The long and winding road

"How close are we to a world where every paper on programming languages is accompanied by an electronic appendix with machine-checked proofs?"

The long and winding road

"How close are we to a world where every paper on **concurrency** is accompanied by an electronic appendix with machine-checked proofs?"

The long and winding road

"How close are we to a world where every paper on **concurrency** is accompanied by an electronic appendix with machine-checked proofs?"

Thank you very much!