A Generic Type System for Higher-Order Ψ-calculi

Alex R. Bendixen, Bjarke B. Bojesen, Hans Hüttel, Stian Lybech

Aalborg University, University of Copenhagen, Reykjavík University

- There is an abundance of variants of the π-calculus

- There is an abundance of variants of the π-calculus
- There is also an abundance of *type systems* for these variants for ensuring correct name usage

- There is an abundance of variants of the π-calculus
- There is also an abundance of *type systems* for these variants for ensuring correct name usage
- But they have something in common:

- There is an abundance of variants of the π-calculus
- There is also an abundance of *type systems* for these variants for ensuring correct name usage
- But they have something in common:
  - Judgments for processes P have the form $\Gamma \vdash P$

- There is an abundance of variants of the π-calculus
- There is also an abundance of *type systems* for these variants for ensuring correct name usage
- But they have something in common:
  - Judgments for processes P have the form $\Gamma \vdash P$
  - Judgements for terms M have the form $\Gamma \vdash M : T$

- Bengtson et al. (2009, 2011) created (first order) Ψ-calculi

- Bengtson et al. (2009, 2011) created (first order) Ψ-calculi
- Huttel (2011) created a generic type system for Ψ-calculi

- Bengtson et al. (2009, 2011) created (first order) Ψ-calculi
- Huttel (2011) created a generic type system for Ψ-calculi
- Parrow et al. (2014) created Higher-Order Ψ-calculi (HOΨ) ...

Motivation

- A type system for the ρ-calculus?

Motivation

- A type system for the ρ-calculus?
  Maybe encode $\rho \to \pi$ and type the result (like HOπ)?

Motivation

- A type system for the ρ-calculus?
  Maybe encode $\rho \rightarrow \pi$ and type the result (like HOπ)?
- But the π-calculus cannot encode the ρ-calculus!

Motivation

- A type system for the ρ-calculus?
  Maybe encode $\rho \to \pi$ and type the result (like HOπ)?
- But the π-calculus cannot encode the ρ-calculus!
- But maybe HOΨ can? (Yes!)

Solution:

Extend the generic type system to the higher-order setting!

The HOΨ-calculus

Parameters

3 nominal sets:

$M \in \mathbb{T}$ terms
$\varphi \in \mathbb{C}$ conditions
$\Psi \in \mathbb{A}$ assertions

Parameters

3 nominal sets:

$M \in \mathbb{T}$ terms
$\varphi \in \mathbb{C}$ conditions
$\Psi \in \mathbb{A}$ assertions

4 equivariant operations:

$\dot{\leftrightarrow} : \mathbb{T} \times \mathbb{T} \to \mathbb{C}$ channel equivalence
$\otimes : \mathbb{A} \times \mathbb{A} \to \mathbb{A}$ assertion composition

$\mathbf{1} \in \mathbb{A}$ assertion unit
$\Vdash \subseteq \mathbb{A} \times \mathbb{C}$ entailment relation

Syntax

$P ::=$	$\mathbf{0}$	Nil
$\mid$	$P_1 \mid P_2$	Parallel
$\mid$	$\overline{M} N.P$	Output
$\mid$	$\underline{M}(\lambda\tilde{x} : \widetilde{T})N.P$	Input
$\mid$	$\mathbf{run}\ M$	Invocation
$\mid$	$\mathbf{case}\ \varphi_1 : P_1 \, [] \, \ldots \, [] \, \varphi_n : P_n$	Selection
$\mid$	$(\boldsymbol{\nu} x : T)P$	Restriction
$\mid$	$!P$	Replication
$\mid$	$(\!\mid \Psi \mid\!)$	Assertion

Syntax

$P ::= \mathbf{0}$	Nil
$\mid\ P_1 \mid P_2$	Parallel
$\mid\ \overline{M}N.P$	Output
$\mid\ \underline{M}(\lambda\tilde{x} : \widetilde{T})N.P$	Input
$\mid\ \mathbf{run}\ M$	Invocation
$\mid\ \mathbf{case}\ \varphi_1 : P_1\ []\ \ldots\ []\ \varphi_n : P_n$	Selection
$\mid\ (\nu x : T)P$	Restriction
$\mid\ !P$	Replication
$\mid\ (\!\mid \Psi \mid\!)$	Assertion

Semantics

Semantics

… are complicated

$$\Psi \triangleright P \rightarrow P'$$

Example (generic)

Assume $\mathbb{A} \triangleq \ldots \cup \{\, M \Leftarrow P \mid M \in \mathbb{T} \wedge P \in \mathscr{P} \,\}$

Example (generic)

Assume $\mathbb{A} \triangleq \ldots \cup \{ M \Leftarrow P \mid M \in \mathbb{T} \wedge P \in \mathscr{P} \}$

$$\Psi \rhd \underline{M}(\lambda x)x.\mathbf{run}\ x \mid \overline{M}N.(\!\{ N \Leftarrow P \}\!)$$

Example (generic)

Assume $\mathbb{A} \triangleq \ldots \cup \{\, M \Leftarrow P \mid M \in \mathbb{T} \wedge P \in \mathscr{P} \,\}$

$$\begin{aligned} &\Psi \rhd \underline{M}(\lambda x)x.\mathbf{run}\ x \mid \overline{M}N.(\!\{\, N \Leftarrow P \,\}\!) \\ \rightarrow &\Psi \rhd \mathbf{run}\ {\color{red}N} \mid (\!\{\, {\color{red}N} \Leftarrow P \,\}\!) \end{aligned}$$

Example (generic)

Assume $\mathbb{A} \triangleq \ldots \cup \{\, M \Leftarrow P \mid M \in \mathbb{T} \wedge P \in \mathscr{P} \,\}$

$$
\begin{aligned}
&\Psi \rhd \underline{M}(\lambda x)x.\mathbf{run}\ x \mid \overline{M}N.(\!\{\, N \Leftarrow P \,\}\!) \\
\rightarrow &\Psi \rhd \mathbf{run}\ N \mid (\!\{\, N \Leftarrow P \,\}\!) \\
\rightarrow &\Psi \otimes \{\, N \Leftarrow P \,\} \rhd \color{red}{P} \mid (\!\{\, N \Leftarrow P \,\}\!)
\end{aligned}
$$

Example (π-calculus)

Let

$$\mathbb{T} \triangleq \mathcal{N}$$
$$\mathbb{C} \triangleq \{x \dot{\leftrightarrow} y \mid x, y \in \mathcal{N}\} \cup \{\top\}$$
$$\Vdash \triangleq \{(\mathbf{1}, x \dot{\leftrightarrow} x) \mid x \in \mathcal{N}\} \cup \{(\mathbf{1}, \top)\}$$
$$\mathbb{A} \triangleq \{\emptyset\}$$
$$\otimes \triangleq \cup$$
$$\mathbf{1} \triangleq \emptyset$$

$$\llbracket \mathbf{0} \rrbracket = \mathbf{0}$$
$$\llbracket P_1 \mid P_2 \rrbracket = \llbracket P_1 \rrbracket \mid \llbracket P_2 \rrbracket$$
$$\llbracket x(y).P \rrbracket = \underline{x}(\lambda y)y.\llbracket P \rrbracket$$
$$\llbracket P_1 + P_2 \rrbracket = \mathbf{case}\ \top : P_1 \ [\!]\ \top : P_2$$
$$\vdots$$

... then you have the π-calculus!

Example (HOπ)

Tweak the parameters a little:

$$\mathbb{T} \triangleq \mathscr{N} \cup \mathscr{P}$$
$$\mathbb{C} \triangleq \{ x \dot{\leftrightarrow} y \mid x, y \in \mathscr{N} \} \cup \{ P \Leftarrow Q \mid P, Q \in \mathscr{P} \} \cup \{ \top \}$$
$$\Vdash \triangleq \{ (1, x \dot{\leftrightarrow} x) \mid x \in \mathscr{N} \} \cup \{ (1, P \Leftarrow P) \mid P \in \mathscr{P} \} \cup \{ (1, \top) \}$$

and with $[\![X]\!] = \textbf{run } x$

... then you get HOπ!

The generic type system

Challenges

- What language of types **Types**?
- What safety-predicate?
- How do we type the parameters $\mathbb{T}$, $\mathbb{C}$ and $\mathbb{A}$?
- What form of type judgments?
- What Γ should higher-order processes be typed relative to?

Challenges

- What language of types **Types**?
  Assume **Types** *is a nominal datatype!*
- What safety-predicate?
- How do we type the parameters $\mathbb{T}$, $\mathbb{C}$ and $\mathbb{A}$?
- What form of type judgments?
- What Γ should higher-order processes be typed relative to?

Challenges

- What language of types **Types**?
  Assume **Types** *is a nominal datatype!*
- What safety-predicate?
  We only show subject reduction!
- How do we type the parameters $\mathbb{T}$, $\mathbb{C}$ and $\mathbb{A}$?
- What form of type judgments?
- What Γ should higher-order processes be typed relative to?

Challenges

- What language of types **Types**?
  Assume **Types** *is a nominal datatype!*
- What safety-predicate?
  We only show subject reduction!
- How do we type the parameters $\mathbb{T}$, $\mathbb{C}$ and $\mathbb{A}$?
  Impose some restrictions and assume type rules are given as parameters!
- What form of type judgments?
- What Γ should higher-order processes be typed relative to?

Challenges

- What language of types **Types**?
  Assume **Types** *is a nominal datatype!*
- What safety-predicate?
  We only show subject reduction!
- How do we type the parameters $\mathbb{T}$, $\mathbb{C}$ and $\mathbb{A}$?
  Impose some restrictions and assume type rules are given as parameters!
- What form of type judgments?
  $\Psi, \Gamma \vdash P$ and $\Psi, \Gamma \vdash \mathscr{J}$ where $\mathscr{J} ::= M : T \mid \varphi \mid \Psi$
- What Γ should higher-order processes be typed relative to?

Challenges

- What language of types **Types**?
  Assume **Types** *is a nominal datatype!*
- What safety-predicate?
  We only show subject reduction!
- How do we type the parameters $\mathbb{T}$, $\mathbb{C}$ and $\mathbb{A}$?
  Impose some restrictions and assume type rules are given as parameters!
- What form of type judgments?
  $\Psi, \Gamma \vdash P$ and $\Psi, \Gamma \vdash \mathscr{J}$ where $\mathscr{J} ::= M : T \mid \varphi \mid \Psi$
- What Γ should higher-order processes be typed relative to?
  It must be derivable from the handle!

The two relations

The two relations

- If a term of type T_1 can carry a term of type T_2 then $T_1 \hookleftarrow T_2$

The two relations

- If a term of type T_1 can carry a term of type T_2 then $T_1 \hookleftarrow T_2$
  Example: $\mathrm{ch}(T) \hookleftarrow T$

The two relations

- If a term of type T_1 can carry a term of type T_2 then $T_1 \looparrowleft T_2$
  Example: $\mathrm{ch}(T) \looparrowleft T$
- If $M \Leftarrow P$ and $M : T$ then $T \curvearrowleft \Gamma$ (such that $\Gamma \vdash P$)

The two relations

- If a term of type T_1 can carry a term of type T_2 then $T_1 \looparrowleft T_2$
  Example: $\mathrm{ch}(T) \looparrowleft T$
- If $M \Leftarrow P$ and $M : T$ then $T \curvearrowleft \Gamma$ (such that $\Gamma \vdash P$)
  Example: $\langle T, \Gamma \rangle \curvearrowleft \Gamma$

Instance assumptions for $\mathbb{T}$, $\mathbb{C}$, $\mathbb{A}$ and **Types**

Instance assumptions for $\mathbb{T}$, $\mathbb{C}$, $\mathbb{A}$ and **Types**

$$[\textsc{t-env-weak}]\ \Gamma, \Psi \vdash \mathcal{J} \implies \Gamma, x : T, \Psi \vdash \mathcal{J}$$

$$[\textsc{t-env-strength}]\ \Gamma, x : T, \Psi \vdash \mathcal{J} \wedge x \notin \mathrm{n}(\mathcal{J}) \implies \Gamma, \Psi \vdash \mathcal{J}$$

$$[\textsc{t-comp-term}]\ \Gamma, \Psi \vdash M[\tilde{x} := \tilde{L}] : F(\widetilde{T}) \implies \Gamma, \Psi \vdash \tilde{L} : \widetilde{T}$$

$$[\textsc{t-ass-weak}]\ \Gamma, \Psi \vdash \mathcal{J} \wedge \Psi \leq \Psi' \wedge \mathrm{n}(\Psi') \subseteq \mathrm{dom}(\Gamma) \implies \Gamma, \Psi' \vdash \mathcal{J}$$

$$[\textsc{t-weak-chaneq}]\ \Psi \Vdash M_1 \dot{\leftrightarrow} M_2 \implies \Psi \otimes \Psi' \Vdash M_1 \dot{\leftrightarrow} M_2$$

$$[\textsc{t-subs}]\ \Gamma, \Psi \vdash \tilde{L} : \widetilde{T} \wedge \Gamma, \tilde{x} : \widetilde{T}, \Psi \vdash \mathcal{J} \implies \Gamma, \Psi \vdash \mathcal{J}[\tilde{x} := \tilde{L}]$$

$$[\textsc{t-equal}]\ \Gamma, \Psi \vdash M : T \wedge \Psi \Vdash M \dot{\leftrightarrow} N \implies \Gamma, \Psi \vdash N : T$$

$$\vdots$$

Main result

Theorem (Subject reduction)

If $\Gamma, \Psi \vdash P \wedge \Psi \rhd P \rightarrow P'$ *then* $\Gamma, \Psi \vdash P'$

... and all the assumptions are satisfied ...

So how does it actually work?

How to get a type system for calculus χ

How to get a type system for calculus χ

1. Instantiate χ as a HOΨ-calculus (define $\mathbb{T}$, $\mathbb{C}$, $\mathbb{A}$ etc.)

How to get a type system for calculus χ

1. Instantiate χ as a HOΨ-calculus (define $\mathbb{T}$, $\mathbb{C}$, $\mathbb{A}$ etc.)
2. Instantiate the type system (define **Types**, $\mathscr{J}$, $\hookleftarrow$, $\curvearrowleft$)

How to get a type system for calculus χ

1. Instantiate χ as a HOΨ-calculus (define $\mathbb{T}$, $\mathbb{C}$, $\mathbb{A}$ etc.)
2. Instantiate the type system (define **Types**, $\mathscr{J}$, $\hookleftarrow$, $\curvearrowleft$)
3. Prove safety and get subject reduction for free!

Example: The ρ-calculus (parameters)

Let

$$\mathbb{T} \triangleq \mathcal{N} \cup \{\ulcorner P \urcorner \mid P \in \mathscr{P}\} \cup \{\langle\!|\ulcorner P \urcorner|\!\rangle \mid P \in \mathscr{P}\}$$

$$\mathbb{C} \triangleq \{M \dot{\leftrightarrow} N \mid M, N \in \mathbb{T}\} \cup \{P_1 \equiv P_2 \mid P_1, P_2 \in \mathscr{P}\}$$
$$\cup \{M \Leftarrow P \mid M \in \mathbb{T} \wedge P \in \mathscr{P}\}$$

$$\mathbb{A} \triangleq \{\emptyset\}$$

$$\otimes \triangleq \cup$$

$$\mathbf{1} \triangleq \emptyset$$

... and postpone $\Vdash$ and $\dot{\leftrightarrow}$ a little.

Example: The ρ-calculus (translation)

Assume *bound names* are implemented as atomic names x, and then define

$$
\begin{aligned}
[\![0]\!] &= \mathbf{0} \\
[\![P_1 \mid P_2]\!] &= [\![P_1]\!] \mid [\![P_2]\!] \\
[\![n(x).P]\!] &= [\![n]\!](\lambda x)\langle\!|x|\!\rangle.[\![P]\!] \\
[\![n\langle\!|P|\!\rangle]\!] &= \overline{[\![n]\!]}\langle\!|\ulcorner[\![P]\!]\urcorner|\!\rangle.\mathbf{0}
\end{aligned}
\qquad
\begin{aligned}
[\![\urcorner x \ulcorner]\!] &= \mathbf{run}\ x \\
[\![\urcorner\ulcorner P \urcorner\ulcorner]\!] &= \mathbf{0} \\
[\![\ulcorner P \urcorner]\!] &= \ulcorner \mathcal{N}[\![P]\!] \urcorner \\
[\![x]\!] &= x
\end{aligned}
$$

where $n ::= x \mid \ulcorner P \urcorner$
and $\mathcal{N}[\![P]\!]$ is similar to $[\![P]\!]$ *except* that $\mathcal{N}[\![\urcorner\ulcorner P \urcorner\ulcorner]\!] = \mathbf{run}\ \ulcorner\mathcal{N}[\![P]\!]\urcorner$.

Example: The ρ-calculus (entailment of $\dot{\leftrightarrow}$)

$$[\textsc{ChanEq}_1]\ \frac{\Psi \Vdash M_1 \dot{\leftrightarrow} M_2}{\Psi \Vdash \ulcorner \textbf{run}\ M_1 \urcorner \dot{\leftrightarrow} M_2} \qquad [\textsc{ChanEq}_2]\ \frac{\Psi \Vdash P_1 \equiv P_2}{\Psi \Vdash \ulcorner P_1 \urcorner \dot{\leftrightarrow} \ulcorner P_2 \urcorner}$$

and reflexive and transitive closure of $\dot{\leftrightarrow}$

Example: The ρ-calculus (entailment of $\equiv$)

$$[\textsc{Par}]\frac{\Psi \Vdash P_1 \equiv P_2}{\Psi \Vdash P_1 \mid R \equiv P_2 \mid R} \qquad [\textsc{In}]\frac{\Psi \Vdash M_1 \dot{\leftrightarrow} M_2 \qquad \Psi \Vdash P_1 \equiv P_2}{\Psi \Vdash \underline{M_1}(\lambda x_1)\langle\!| x_1 |\!\rangle.P_1 \equiv \underline{M_2}(\lambda x_2)\langle\!| x_2 |\!\rangle.P_2}$$

$$[\textsc{Run}]\frac{\Psi \Vdash M_1 \dot{\leftrightarrow} M_2}{\Psi \Vdash \textbf{run}\ M_1 \equiv \textbf{run}\ M_2} \qquad [\textsc{Out}]\frac{\Psi \Vdash M_1 \dot{\leftrightarrow} M_2 \qquad \Psi \Vdash P_1 \equiv P_2}{\Psi \Vdash \overline{M_1}\langle\!| \ulcorner P_1 \urcorner |\!\rangle \equiv \overline{M_2}\langle\!| \ulcorner P_2 \urcorner |\!\rangle}$$

and $\equiv_\alpha \subseteq \equiv$ and $(\mathscr{P}_{/\equiv}, \mid, \mathbf{0})$ an abelian monoid

Challenges for a type system for the ρ-calculus

1. All names are global, so we cannot get a type from $(\boldsymbol{\nu}x : T)P$
2. New names can be constructed at runtime, so what type should they get?

Example: The type system (types)

Let

$$\begin{aligned} T \in \textbf{Types} &::= \langle \alpha, \beta \rangle \\ \alpha &::= \mathrm{ch}(T) \;\Big|\; \mathrm{nil} \\ \beta &::= \Gamma \;\Big|\; \mathrm{nil} \end{aligned}$$

and redefine

$$\mathbb{A} \triangleq \wp(\{ \ulcorner P \urcorner : T \mid P \in \mathscr{P} \wedge T \in \textbf{Types} \} \cup \{ \langle\!|\ulcorner P \urcorner|\!\rangle : T \mid P \in \mathscr{P} \wedge T \in \textbf{Types} \})$$

to give us a place to record the types of the names-to-be.

Example: The type system (type environment)

Append assertions to input and output:

$$\llbracket \ulcorner R \urcorner \langle P \rangle \rrbracket \triangleq \overline{\ulcorner \llbracket R \rrbracket \urcorner} \langle \ulcorner \llbracket P \rrbracket \urcorner \rangle . \mathbf{0} \mid (\!| \{ \ulcorner \llbracket R \rrbracket \urcorner : T, \langle \ulcorner \llbracket P \rrbracket \urcorner \rangle : T' \} |\!)$$

$$\llbracket \ulcorner R \urcorner (x) . P \rrbracket \triangleq \underline{\ulcorner \llbracket R \rrbracket \urcorner} (\lambda x) \langle x \rangle . \llbracket P \rrbracket \mid (\!| \{ \ulcorner \llbracket R \rrbracket \urcorner : T \} |\!)$$

to use Ψ as a 'type environment' for *processes*.

Example: The type system (parameters)

$$[\text{T-COMP}]\ \langle \mathrm{ch}(T), \beta\rangle \looparrowleft T \qquad [\text{T-ENV}]\ \langle \alpha, \Gamma\rangle \curvearrowleft \Gamma$$

$$[\text{TERM-1}]\frac{\ulcorner P \urcorner : \langle \alpha, \Gamma'\rangle \in \Psi \qquad \Gamma', \Psi \vdash P}{\Gamma, \Psi \vdash \ulcorner P \urcorner : \langle \alpha, \Gamma'\rangle} \qquad [\text{TERM-2}]\frac{(\!|\ulcorner P \urcorner|\!) : \langle \alpha, \Gamma'\rangle \in \Psi \qquad \Gamma', \Psi \vdash P}{\Gamma, \Psi \vdash (\!|\ulcorner P \urcorner|\!) : \langle \alpha, \Gamma'\rangle}$$

$$[\text{T-ASS}]\frac{P : T \in \Psi' \implies T \curvearrowleft \Gamma}{\Gamma, \Psi \vdash (\!|\Psi'|\!)} \qquad [\text{TERM-3}]\frac{\Gamma(x) = T}{\Gamma, \Psi \vdash x : T}$$

An unavoidable limitation

We must also redefine

$$[\textsc{ChanEq}_2]\ \frac{\Gamma, \Psi \Vdash P_1 \equiv P_2 \qquad \Gamma, \Psi \vdash \ulcorner P_1 \urcorner : T \iff \Gamma, \Psi \vdash \ulcorner P_2 \urcorner : T}{\Gamma, \Psi \Vdash \ulcorner P_1 \urcorner \dot{\leftrightarrow} \ulcorner P_2 \urcorner}$$

Channel equivalent terms must have the same type!

Conclusions

Conclusions

- HOΨ is useful to experiment with a *family* of languages.

Conclusions

- HOΨ is useful to experiment with a *family* of languages.
- The generic type system is useful to experiment with a *family* of type systems.

Conclusions

- HOΨ is useful to experiment with a *family* of languages.
- The generic type system is useful to experiment with a *family* of type systems.
- The instance assumptions give insights into *minimal requirements* for a type system for HOΨ-instances.

Conclusions

- HOΨ is useful to experiment with a *family* of languages.
- The generic type system is useful to experiment with a *family* of type systems.
- The instance assumptions give insights into *minimal requirements* for a type system for HOΨ-instances.
- The ρ-calculus *is* such an instance.
  (But typing reflection is still a mess)