Minimal Session Types for the π-calculus

PPDP 2021, Tallinn

Alen Arslanagić, Jorge A. Pérez, and Anda-Amelia Palamariuc

University of Groningen, The Netherlands

UNIFYING
CORRECTNESS FOR
COMMUNICATING
SOFTWARE

Our Work: Session Types from First Principles

- A study of **sequentiality** in **session types** for correct message-passing programs
- Sequential composition in types is key to protocol specification, but is not supported by most programming languages

Our Work: Session Types from First Principles

- A study of **sequentiality** in **session types** for correct message-passing programs
- Sequential composition in types is key to protocol specification, but is not supported by most programming languages

- Minimal session types (MSTs): Session types without sequential composition (';')
- Our prior work, a **minimality result**:
  every well-typed process can be decomposed into a process typable with MSTs.
- We focused on HO, a core higher-order process calculus (with abstraction passing).

Our Work: Session Types from First Principles

- A study of **sequentiality** in **session types** for correct message-passing programs
- Sequential composition in types is key to protocol specification, but is not supported by most programming languages

- Minimal session types (MSTs): Session types without sequential composition (';')
- Our prior work, a **minimality result**:
  every well-typed process can be decomposed into a process typable with MSTs.
- We focused on HO, a core higher-order process calculus (with abstraction passing).

- In the paper: MSTs for a **first-order** π-calculus (with name passing).
  - A new minimality result for π, based on the decomposition function $\mathcal{F}(\cdot)$
  - $\mathcal{F}^*(\cdot)$: an optimized decomposition function without redundant communications
  - Correctness proofs and examples for $\mathcal{F}(\cdot)$ and $\mathcal{F}^*(\cdot)$

Our Work: Session Types from First Principles

- A study of **sequentiality** in **session types** for correct message-passing programs
- Sequential composition in types is key to protocol specification, but is not supported by most programming languages

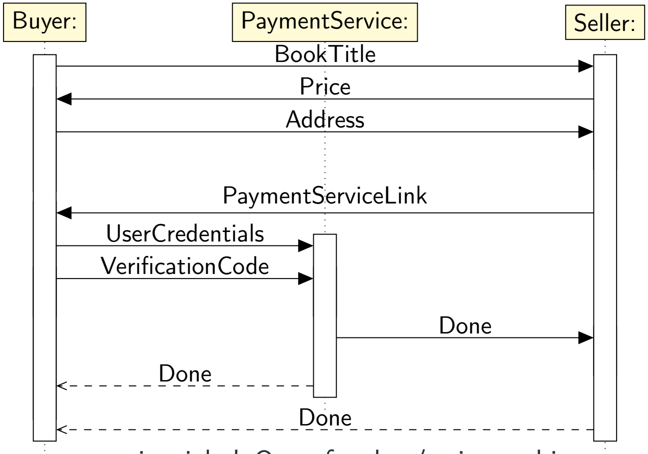
- Minimal session types (MSTs): Session types without sequential composition (';')
- Our prior work, a **minimality result**:
  every well-typed process can be decomposed into a process typable with MSTs.
- We focused on HO, a core higher-order process calculus (with abstraction passing).

- In the paper: MSTs for a **first-order** π-calculus (with name passing).
  - A new minimality result for π, based on the decomposition function $\mathcal{F}(\,\cdot\,)$
  - $\mathcal{F}^*(\,\cdot\,)$: an optimized decomposition function without redundant communications
  - Correctness proofs and examples for $\mathcal{F}(\,\cdot\,)$ and $\mathcal{F}^*(\,\cdot\,)$
- Minimality results based on MSTs do not depend on the kind of communicated objects

Context and Key Questions

Message-Passing Concurrency

- Key to most software systems today. Supported by Go, Erlang, Cloud Haskell, ...
- A typical e-commerce protocol:



- Communication correctness is tricky! Out-of-order / mismatching messages, deadlocks.

Session Types: The Good

- Type-based approach to communication correctness.
  Widely developed, multiple extensions and implementations.
- Session type: **what** and **when** should be sent through a channel.
  Correctness follows from type-level compatibility and linearity.

Session Types: The Good

- Type-based approach to communication correctness.
  Widely developed, multiple extensions and implementations.
- Session type: **what** and **when** should be sent through a channel.
  Correctness follows from type-level compatibility and linearity.
- A session type for the payment service

$$?(\mathsf{Str});?(\mathsf{Int});!\langle \mathsf{Bool} \rangle;\mathtt{end}$$

Sequential Composition in Session Types

- Distinctive feature. Very useful to specify / check intended protocol structures.

Session Types: The Good

- Type-based approach to communication correctness.
  Widely developed, multiple extensions and implementations.
- Session type: **what** and **when** should be sent through a channel.
  Correctness follows from type-level compatibility and linearity.
- A session type for the payment service on channel/endpoint u:

$$u : ?(\mathsf{Str});?(\mathsf{Int});!\langle \mathsf{Bool}\rangle;\mathtt{end}$$

Sequential Composition in Session Types

- Distinctive feature. Very useful to specify / check intended protocol structures.
- Goes hand-in-hand with sequential composition in processes (prefixes):

$$S_{\mathtt{Pay}} = u?(\mathit{UserCredentials}).u?(\mathit{Verification}).u!\langle \mathit{IsBalanceOK}\rangle.\mathbf{0}$$

Session Types: The Reality

- Sequential composition in types not typically supported by programming languages. Channel types only declare payload types and channel directions, not structure.
  - In Go:

    ```
    ch := make(chan int)
    ```
  - In CloudHaskell:

    ```
    (s,r) <- newChan::Process (SendPort Int, ReceivePort Int)
    ```

Session Types: The Reality

- Sequential composition in types not typically supported by programming languages. Channel types only declare payload types and channel directions, not structure.
  - In Go:
    ```
    ch := make(chan int)
    ```
  - In CloudHaskell:
    ```
    (s,r) <- newChan::Process (SendPort Int, ReceivePort Int)
    ```
- Programmers must enforce sequentiality themselves $\rightsquigarrow$ Error-prone
- A gap between theory and practice, still not fully understood.

Understanding the Gap

Can we dispense with sequential composition in session types?

Minimal Session Types (MSTs)

Session types without sequentiality — only ‘$\mathtt{end}$’ can appear after ‘;’.

Examples: ‘$?(\mathsf{Str});\mathtt{end}$’ and ‘$!\langle \mathsf{Int}, \mathsf{Bool}\rangle;\mathtt{end}$’.

Understanding the Gap

Can we dispense with sequential composition in session types?

Minimal Session Types (MSTs)

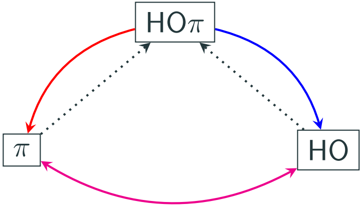
Session types without sequentiality — only ‘end’ can appear after ‘;’.
Examples: ‘$?(\mathsf{Str});\mathtt{end}$’ and ‘$!\langle \mathsf{Int}, \mathsf{Bool}\rangle;\mathtt{end}$’.

Different justifications for standard session types:

- **Formally**:
  Type-directed compilations to processes typable with MSTs (*minimality result*).
- **Conceptually**:
  Session types in terms of themselves (*absolute expressiveness*).
- **Pragmatically**:
  A potential new avenue for integrating session types in PLs.

A Language for MSTs?

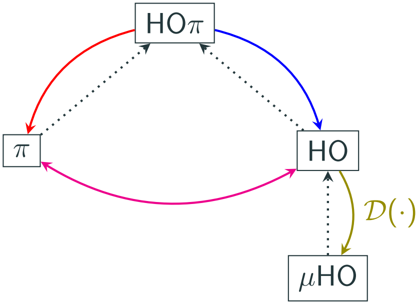
A Hierarchy of Session-Typed Process Languages (Kouzapas et al. - ESOP'16, I&C'19)



- HOπ: the higher-order π-calculus **with sessions**.
  Two relevant sub-calculi: π and HO.
- While π is strictly first-order (name passing only)...
- ... HO is a compact blend of λ- and π-calculi:
  - Passing of abstractions $\lambda x. P$, channels to processes
  - Recursive types, but no recursion in processes
  - Very expressive! Can encode name-passing, recursion
- HO and π are mutually encodable.

A Language for MSTs?

A Hierarchy of Session-Typed Process Languages (Kouzapas et al. - ESOP'16, I&C'19)



- HOπ: the higher-order π-calculus **with sessions**. Two relevant sub-calculi: π and HO.
- While π is strictly first-order (name passing only)...
- ... HO is a compact blend of λ- and π-calculi:
  - Passing of abstractions $\lambda x.\, P$, channels to processes
  - Recursive types, but no recursion in processes
  - Very expressive! Can encode name-passing, recursion
- HO and π are mutually encodable.

Our prior work (ECOOP'19) – HO with MSTs, denoted μHO

- Sequentiality in types can be codified by sequentiality in processes.
- Only sequential composition in processes is truly indispensable.

MSTs, In One Slide

A process P typed with standard session types $S_1, \ldots, S_n$:

MSTs, In One Slide

A process P typed with standard session types $S_1, \ldots, S_n$:

- Sequencing in $S_1, \ldots, S_n$ is codified by $\mathcal{D}(P)$, the **decomposition** of P.
- Each session type S_i is decomposed into $\mathcal{G}(S_i)$, a list of minimal session types.

MSTs, In One Slide

A process P typed with standard session types $S_1, \ldots, S_n$:

- Sequencing in $S_1, \ldots, S_n$ is codified by $\mathcal{D}(P)$, the **decomposition** of P.
- Each session type S_i is decomposed into $\mathcal{G}(S_i)$, a list of minimal session types.

$$\boxed{P}$$

$$u : ?(\mathsf{Str});?(\mathsf{Int});!\langle \mathsf{Bool} \rangle;\mathtt{end}$$

MSTs, In One Slide

A process P typed with standard session types $S_1, \ldots, S_n$:

- Sequencing in $S_1, \ldots, S_n$ is codified by $\mathcal{D}(P)$, the **decomposition** of P.
- Each session type S_i is decomposed into $\mathcal{G}(S_i)$, a list of minimal session types.

$$\boxed{P}$$

$$u : ?(\mathsf{Str});?(\mathsf{Int});!\langle \mathsf{Bool} \rangle;\mathtt{end}$$

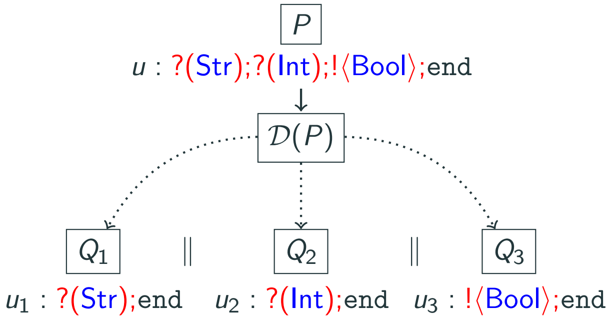
$$\downarrow$$

$$\boxed{\mathcal{D}(P)}$$

MSTs, In One Slide

A process P typed with standard session types $S_1, \ldots, S_n$:

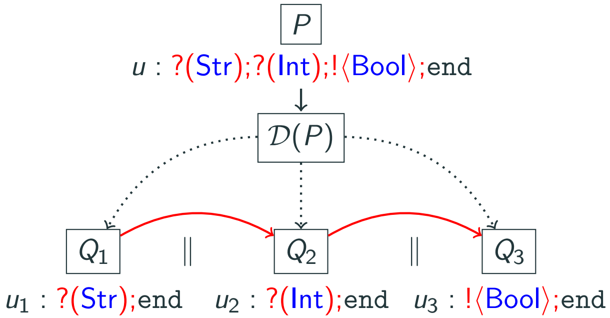
- Sequencing in $S_1, \ldots, S_n$ is codified by $\mathcal{D}(P)$, the **decomposition** of P.
- Each session type S_i is decomposed into $\mathcal{G}(S_i)$, a list of minimal session types.



MSTs, In One Slide

A process P typed with standard session types $S_1, \ldots, S_n$:

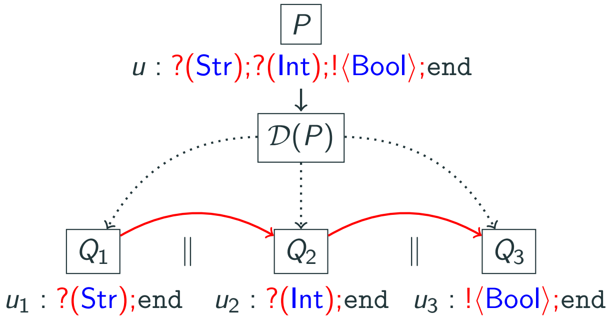
- Sequencing in $S_1, \ldots, S_n$ is codified by $\mathcal{D}(P)$, the **decomposition** of P.
- Each session type S_i is decomposed into $\mathcal{G}(S_i)$, a list of minimal session types.



MSTs, In One Slide

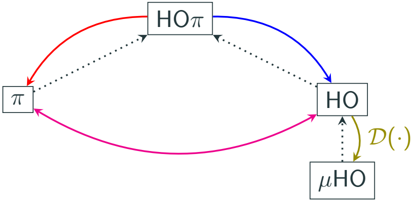
A process P typed with standard session types $S_1, \ldots, S_n$:

- Sequencing in $S_1, \ldots, S_n$ is codified by $\mathcal{D}(P)$, the **decomposition** of P.
- Each session type S_i is decomposed into $\mathcal{G}(S_i)$, a list of minimal session types.



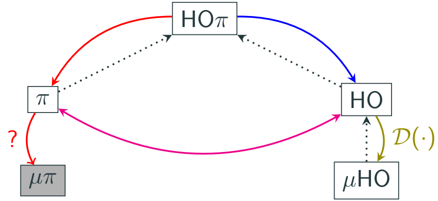
Sequencing in session types admits simpler explanations! If $\Gamma \vdash P$ then $\mathcal{G}(\Gamma) \vdash \mathcal{D}(P)$.

Open Question: MSTs for the π-calculus



- Our decomposition for HO heavily exploits abstraction passing to obtain MSTs.

Open Question: MSTs for the π-calculus



- Our decomposition for HO heavily exploits abstraction passing to obtain MSTs.

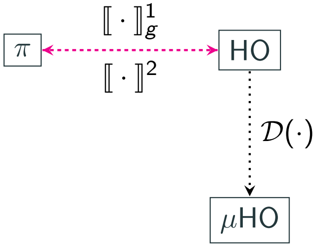
Open Question

Session types have been widely studied for first-order languages, with name passing. Does the minimality result hold also for π, the other sub-calculus of HOπ?

This Work

Decomposition by Composition

- We reuse typed encodings between π and HO



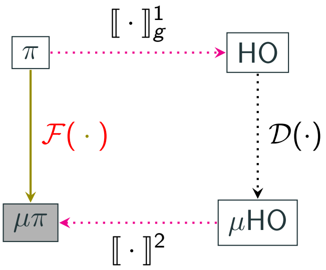
Decomposition by Composition

- We reuse typed encodings between π and HO
- Compose three known functions:
  - $[\![\cdot]\!]_g^1 : \pi \to \mathsf{HO}$ (typed encoding)
  - $\mathcal{D}(\cdot) : \mathsf{HO} \to \mu\mathsf{HO}$ (decomposition function)
  - $[\![\cdot]\!]^2 : \mathsf{HO} \to \pi$ (typed encoding)

  (Encodings on types are also composed.)
- The resulting function is $\mathcal{F}(\,\cdot\,) : \pi \to \mu\pi$

  Correctness follows by composing the three functions

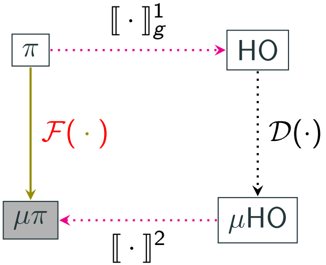
  (The decomposition on types is $\mathcal{H}(\,\cdot\,)$)



Decomposition by Composition

- We reuse typed encodings between π and HO
- Compose three known functions:
  - $[\![\cdot]\!]^1_g : \pi \to \text{HO}$ (typed encoding)
  - $\mathcal{D}(\cdot) : \text{HO} \to \mu\text{HO}$ (decomposition function)
  - $[\![\cdot]\!]^2 : \text{HO} \to \pi$ (typed encoding)

  (Encodings on types are also composed.)
- The resulting function is $\mathcal{F}(\cdot) : \pi \to \mu\pi$
  Correctness follows by composing the three functions
  (The decomposition on types is $\mathcal{H}(\cdot)$)
- **Outcome**: A positive, elegant answer to the open question — the minimality result holds for π, too



$$
\begin{aligned}
n &::= a, b \mid s, \overline{s} \\
u, w &::= n \mid x, y, z \\
V, W &::= u \mid \boxed{\lambda x.\, P} \mid \boxed{x, y, z} \\
P, Q &::= u!\langle V \rangle.P \mid u?(x).P \\
&\mid \boxed{V\ u} \mid P \mid Q \mid (\nu\, n)P \mid \mathbf{0} \mid X \mid \mu X.P
\end{aligned}
$$

- The sub-language π lacks boxed constructs
- The sub-language HO lacks shaded constructs

Session Types, Now Minimal

Session Types for π

$$C ::= S \mid \langle S \rangle$$

$$S ::= !\langle C \rangle; S \mid ?(C); S \mid \mu \mathtt{t}.S \mid \mathtt{t} \mid \mathtt{end}$$

MSTs for π

$$C ::= M \mid \langle M \rangle$$

$$M ::= \gamma \mid !\langle \widetilde{C} \rangle; \gamma \mid ?(\widetilde{C}); \gamma \mid \mu \mathtt{t}.M$$

$$\gamma ::= \mathtt{end} \mid \mathtt{t}$$

Session Types, Now Minimal

Session Types for π

$$C ::= S \mid \langle S \rangle$$
$$S ::= !\langle C \rangle; S \mid ?(C); S \mid \mu \mathtt{t}.S \mid \mathtt{t} \mid \mathtt{end}$$

MSTs for π

$$C ::= M \mid \langle M \rangle$$
$$M ::= \gamma \mid !\langle \widetilde{C} \rangle; \gamma \mid ?(\widetilde{C}); \gamma \mid \mu \mathtt{t}.M$$
$$\gamma ::= \mathtt{end} \mid \mathtt{t}$$

Note: We often omit ‘end’. Thus, ‘$!\langle \widetilde{C} \rangle$’ and ‘$?(\widetilde{C})$’ stand for ‘$!\langle \widetilde{C} \rangle; \mathtt{end}$’ and ‘$?(\widetilde{C}); \mathtt{end}$’.

MSTs for π: Step by Step

Output case $P = u_i!\langle w_j \rangle.Q$

- First step $\mathcal{A}'^{k}_{\tilde{x}}(\,\cdot\,)_g = \mathcal{D}(\llbracket \cdot \rrbracket^1_g) : \pi \to \mu\mathsf{HO}$

$$\mathcal{A}'^{k}_{\tilde{x}}(u_i!\langle w_j \rangle.Q)_g = c_k?(\tilde{x}).u_i!\langle W \rangle.\overline{c_{k+3}}!\langle \tilde{x} \rangle \mid \mathcal{A}'^{k+3}_{\tilde{x}}(Q\sigma)_g \;\; (\sigma = (u_i : S)\,?\,\{^{u_{i+1}}/u_i\} : \{\})$$

$$\text{where } W = \lambda z_1.\,(\overline{c_{k+1}}!\langle\rangle \mid c_{k+1}?().z_1?(x).\overline{c_{k+2}}!\langle x \rangle \mid c_{k+2}?(x).(x\,\widetilde{w}))$$

Output case $P = u_i!\langle w_j\rangle.Q$

- First step $\mathcal{A}'^{k}_{\tilde{x}}(\,\cdot\,)_g = \mathcal{D}([\![\,\cdot\,]\!]^1_g) : \pi \to \mu\mathrm{HO}$

$$\mathcal{A}'^{k}_{\tilde{x}}(u_i!\langle w_j\rangle.Q)_g = c_k?(\tilde{x}).u_i!\langle W\rangle.\overline{c_{k+3}}!\langle\tilde{x}\rangle \mid \mathcal{A}'^{k+3}_{\tilde{x}}(Q\sigma)_g \;\; (\sigma = (u_i : S)\,?\,\{^{u_{i+1}}/_{u_i}\} : \{\})$$

$$\text{where } W = \lambda z_1.\,(\overline{c_{k+1}}!\langle\rangle \mid c_{k+1}?().z_1?(x).\overline{c_{k+2}}!\langle x\rangle \mid c_{k+2}?(x).(x\,\widetilde{w}))$$

- Second step $\mathcal{A}^{k}_{\tilde{x}}(\,\cdot\,)_g = [\![\mathcal{A}'^{k}_{\tilde{x}}(\,\cdot\,)_g]\!]^2 : \pi \to \mu\pi$

$$\begin{aligned}\mathcal{A}^{k}_{\tilde{x}}(u_i!\langle w_j\rangle.Q)_g = c_k?(\widetilde{x}).(\nu\, a)(u_i!\langle a\rangle.(\overline{c_{k+3}}!\langle\widetilde{x}\rangle \mid \mathcal{A}^{k+3}_{\tilde{x}}(Q\sigma)_g \mid \\ a?(y).y?(z_1).\overline{c_{k+1}}!\langle z_1\rangle \mid c_{k+1}?(z_1).z_1?(x).\overline{c_{k+2}}!\langle x\rangle \mid \\ c_{k+2}?(x).(\nu\, s)(x!\langle s\rangle.\overline{s}!\langle\widetilde{w}\rangle)))\end{aligned}$$

Output case $P = u_i!\langle w_j\rangle.Q$

- First step $\mathcal{A}'^{k}_{\tilde{x}}(\cdot)_g = \mathcal{D}(\llbracket \cdot \rrbracket^1_g) : \pi \to \mu\mathrm{HO}$

$$\mathcal{A}'^{k}_{\tilde{x}}(u_i!\langle w_j\rangle.Q)_g = c_k?(\tilde{x}).u_i!\langle W\rangle.\overline{c_{k+3}}!\langle\tilde{x}\rangle \mid \mathcal{A}'^{k+3}_{\tilde{x}}(Q\sigma)_g \quad (\sigma = (u_i : S)\,?\,\{^{u_{i+1}}/u_i\} : \{\})$$

$$\text{where } W = \lambda z_1.\,(\overline{c_{k+1}}!\langle\rangle \mid c_{k+1}?().z_1?(x).\overline{c_{k+2}}!\langle x\rangle \mid c_{k+2}?(x).(x\,\widetilde{w}))$$

- Second step $\mathcal{A}^{k}_{\tilde{x}}(\cdot)_g = \llbracket \mathcal{A}'^{k}_{\tilde{x}}(\cdot)_g \rrbracket^2 : \pi \to \mu\pi$

$$\begin{aligned}\mathcal{A}^{k}_{\tilde{x}}(u_i!\langle w_j\rangle.Q)_g = c_k?(\widetilde{x}).(\nu\,a)(u_i!\langle a\rangle.(\overline{c_{k+3}}!\langle\widetilde{x}\rangle \mid \mathcal{A}^{k+3}_{\tilde{x}}(Q\sigma)_g \mid \\ a?(y).y?(z_1).\overline{c_{k+1}}!\langle z_1\rangle \mid c_{k+1}?(z_1).z_1?(x).\overline{c_{k+2}}!\langle x\rangle \mid \\ c_{k+2}?(x).(\nu\,s)(x!\langle s\rangle.\overline{s}!\langle\widetilde{w}\rangle)))\end{aligned}$$

$$\mathcal{F}(P) = (\nu\,\widetilde{c})(\overline{c_1}!\langle\rangle \mid \mathcal{A}^1_\epsilon(P))$$

MSTs for π: Example

P implements channel u of type $S = ?(\mathsf{Int});?(\mathsf{Int});!\langle\mathsf{Bool}\rangle;\mathtt{end}$:

$$P = (\nu\, u : S)(\underbrace{w!\langle\overline{u}\rangle.u?(a).u?(b).u!\langle a \geq b\rangle.\mathbf{0}}_{A} \mid \underbrace{\overline{w}?(x).x!\langle 5\rangle.x!\langle 4\rangle.x?(b).\mathbf{0}}_{B})$$

MSTs for π: Example

P implements channel u of type $S = ?(\mathsf{Int});?(\mathsf{Int});!\langle\mathsf{Bool}\rangle;\mathtt{end}$:

$$P = (\nu\, u : S)(\underbrace{w!\langle\overline{u}\rangle.u?(a).u?(b).u!\langle a \geq b\rangle.\mathbf{0}}_{A} \mid \underbrace{\overline{w}?(x).x!\langle 5\rangle.x!\langle 4\rangle.x?(b).\mathbf{0}}_{B})$$

The decomposition of P:

$$\mathcal{F}(P) = (\nu\, c_1, \ldots, c_{25})(\overline{c_1}!\langle\rangle.\mathbf{0} \mid (\nu\, u_1)c_1?().\overline{c_2}!\langle\rangle.\overline{c_{13}}!\langle\rangle \mid \mathcal{A}^2_\epsilon(A\sigma') \mid \mathcal{A}^{13}_\epsilon(B\sigma'))$$

$\mathcal{A}^2_\epsilon(A)$	$\mathcal{A}^{13}_\epsilon(B)$
$c_2?().(\nu\, a_1)(w_1!\langle a_1\rangle.($ $\overline{c_5}!\langle\rangle \mid \mathcal{A}^5_\epsilon(A') \mid$ $a_1?(y_1).y_1?(z_1).\overline{c_3}!\langle z_1\rangle \mid$ $c_3?(z_1).z_1?(x).\overline{c_4}!\langle x\rangle \mid$ $c_4?(x).(\nu\, s)(x!\langle s\rangle.\overline{s}!\langle\overline{u}_1, \overline{u}_2, \overline{u}_3\rangle)))$	$c_{13}?().\overline{w_1}?(y_4).\overline{c_{14}}!\langle y_4\rangle \mid$ $(\nu\, s_1)(c_{14}?(y).\overline{c_{15}}!\langle y\rangle.\overline{c_{16}}!\langle\rangle \mid$ $c_{15}?(y_4).(\nu\, s'')(y_4!\langle s''\rangle.\overline{s''}!\langle s_1\rangle.\mathbf{0}) \mid$ $c_{16}?().(\nu\, a_3)(s_1!\langle a_3\rangle.(\overline{c_{21}}!\langle\rangle \mid c_{21}?().\mathbf{0} \mid$ $a_3?(y_5).y_5?(x_1, x_2, x_3).(\overline{c_{17}}!\langle\rangle \mid \mathcal{A}^{17}_\epsilon(B')))))$

MSTs for π: Example

$\mathcal{A}^2_\epsilon(A)$

$$c_2?().(\nu\, a_1)(\, w_1!\langle a_1\rangle.\ (\\ \overline{c_5}!\langle\rangle \mid \mathcal{A}^5_\epsilon(A') \mid \\ a_1?(y_1).y_1?(z_1).\overline{c_3}!\langle z_1\rangle \mid \\ c_3?(z_1).z_1?(x).\overline{c_4}!\langle x\rangle \mid \\ c_4?(x).(\nu\, s)(x!\langle s\rangle.\ \overline{s}!\langle \overline{u}_1, \overline{u}_2, \overline{u}_3\rangle\ )))$$

$\mathcal{A}^{13}_\epsilon(B)$

$$c_{13}?().\ \overline{w_1}?(y_4).\ \overline{c_{14}}!\langle y_4\rangle \mid \\ (\nu\, s_1)(c_{14}?(y).\overline{c_{15}}!\langle y\rangle.\overline{c_{16}}!\langle\rangle \mid \\ c_{15}?(y_4).(\nu\, s'')(y_4!\langle s''\rangle.\overline{s''}!\langle s_1\rangle.\mathbf{0}) \mid \\ c_{16}?().(\nu\, a_3)(s_1!\langle a_3\rangle.(\overline{c_{21}}!\langle\rangle \mid c_{21}?().\mathbf{0} \mid \\ a_3?(y_5).\ y_5?(x_1, x_2, x_3).\ (\overline{c_{17}}!\langle\rangle \mid \mathcal{A}^{17}_\epsilon(B')))))$$

Minimal STs

$$w_1 : M = !\langle\langle ?(?(\langle ?(M_1, M_2, M_3)\rangle))\rangle\rangle$$

$$M_1 = ?(\langle ?(?(\langle ?(\mathsf{Int})\rangle))\rangle)$$

$$M_2 = ?(\langle ?(?(\langle ?(\mathsf{Int})\rangle))\rangle)$$

$$M_3 = !\langle\langle ?(?(\langle ?(\mathsf{Bool})\rangle))\rangle\rangle$$

An Optimized Decomposition

- Although conceptually simple, the function $\mathcal{F}(\cdot)$ obtained by "decompose by composition" induces redundancies
- Suboptimal features:
  1. channel redirections
  2. redundant synchronizations
  3. the structure of trio is lost
- Redundancies most prominent when treating recursive names and processes

An Optimized Decomposition

- Although conceptually simple, the function $\mathcal{F}(\,\cdot\,)$ obtained by "decompose by composition" induces redundancies
- Suboptimal features:
  1. channel redirections
  2. redundant synchronizations
  3. the structure of trio is lost
- Redundancies most prominent when treating recursive names and processes
- $\mathcal{F}^*(\,\cdot\,)$ is an optimized decomposition function:
  1. removes redundant synchronizations
  2. use native support for recursion in π
  3. recovers trio structure

  Optimized decomposition on types: $\mathcal{H}^*(\,\cdot\,)$

Optimized Decomposition: Example

P implements channel u of type $S = ?(\mathsf{Int});?(\mathsf{Int});!\langle\mathsf{Bool}\rangle;\mathtt{end}$:

$$P = (\nu\, u : S)(\underbrace{w!\langle\overline{u}\rangle.u?(a).u?(b).u!\langle a \geq b\rangle.\mathbf{0}}_{A} \mid \underbrace{\overline{w}?(x).x!\langle 5\rangle.x!\langle 4\rangle.x?(b).\mathbf{0}}_{B})$$

The optimized decomposition:

$$\mathcal{F}^*(P) = (\nu\, \widetilde{c})(\overline{c_1}!\langle\rangle \mid (\nu\, u_1, u_2, u_3)c_1?().\overline{c_2}!\langle\rangle.\overline{c_6}!\langle\rangle \mid \mathbb{A}^2_\epsilon(A\sigma') \mid \mathbb{A}^6_\epsilon(B\sigma'))$$

$\mathbb{A}^2_\epsilon(A\sigma')$

$$\begin{aligned} & c_2?().\, w_1!\langle\overline{u}_1, \overline{u}_2, \overline{u}_3\rangle.\, \overline{c_3}!\langle\rangle \mid \\ & \quad c_3?().\, u_1?(a).\, \overline{c_4}!\langle a\rangle \mid \\ & \quad\quad c_4?().\, u_2?(b).\, \overline{c_5}!\langle a, b\rangle \mid \\ & \quad\quad\quad c_5?().\, u_3!\langle a \geq b\rangle.\, \overline{c_6}!\langle\rangle \mid c_6?().\mathbf{0} \end{aligned}$$

$\mathbb{A}^6_\epsilon(B\sigma')$

$$\begin{aligned} & c_6?().\, \overline{w_1}?(x_1, x_2, x_3).\, \overline{c_7}!\langle x_1, x_2, x_3\rangle \mid \\ & \quad c_7?(x_1, x_2, x_3).\, x_1!\langle 5\rangle.\, \overline{c_8}!\langle x_2, x_3\rangle \mid \\ & \quad\quad c_8?(x_2, x_3).\, x_1!\langle 4\rangle.\, \overline{c_9}!\langle x_3\rangle \mid \\ & \quad\quad\quad c_9?(x_2).\, x_3?(b_1).\, \overline{c_{10}}!\langle\rangle \mid c_{10}?().\mathbf{0} \end{aligned}$$

Decomposing Session Types

$$A = u?(a).u?(b).u!\langle(a \geq b)\rangle.\mathbf{0}$$

$$u : ?(\mathsf{Int});?(\mathsf{Int});!\langle\mathsf{Bool}\rangle;\mathtt{end}$$

$$\downarrow$$

$$\mathcal{F}^*(A)$$

$$c_3?().u_1?(a).\overline{c_4}!\langle a\rangle \quad \| \quad c_4?(a).u_2?(b).\overline{c_5}!\langle a, b\rangle \quad \| \quad c_5?(a,b).u_3!\langle a \geq b\rangle.\overline{c_6}!\langle\rangle$$

$u_1 : ?(\mathsf{Int})$
$c_3 : ?()$

$u_2 : ?(\mathsf{Int})$
$c_4 : ?(\mathsf{Int})$

$u_3 : !\langle\mathsf{Bool}\rangle$
$c_5 : ?(\mathsf{Int}, \mathsf{Int})$

Improvements: Comparing Types Decompositions

$\mathcal{H}(\,\cdot\,)$

$$\mathcal{H}(!\langle C\rangle; S) = \begin{cases} M_C & \text{if } S = \text{end} \\ M_C, \mathcal{H}(S) & \text{otherwise} \end{cases}$$

where

$$M_C = !\langle\langle ?(?(\langle ?(\mathcal{H}(C))\rangle))\rangle\rangle$$

$$\mathcal{H}(?(C); S) = \begin{cases} M_C & \text{if } S = \text{end} \\ M_C, \mathcal{H}(S) & \text{otherwise} \end{cases}$$

where

$$M_C = ?(\langle ?(?(\langle ?(\mathcal{H}(C))\rangle))\rangle)$$

Improvements: Comparing Types Decompositions

$\mathcal{H}(\cdot)$

$$\mathcal{H}(!\langle C\rangle; S) = \begin{cases} M_C & \text{if } S = \text{end} \\ M_C, \mathcal{H}(S) & \text{otherwise} \end{cases}$$

where

$$M_C = !\langle\langle ?(?(\langle ?(\mathcal{H}(C))\rangle))\rangle\rangle$$

$$\mathcal{H}(?(C); S) = \begin{cases} M_C & \text{if } S = \text{end} \\ M_C, \mathcal{H}(S) & \text{otherwise} \end{cases}$$

where

$$M_C = ?(\langle ?(?(\langle ?(\mathcal{H}(C))\rangle))\rangle)$$

$\mathcal{H}^*(\cdot)$

$$\mathcal{H}^*(!\langle C\rangle; S) = \begin{cases} M_C & \text{if } S = \text{end} \\ M_C\,, \mathcal{H}^*(S) & \text{otherwise} \end{cases}$$

where

$$M_C = !\langle \mathcal{H}^*(C)\rangle$$

$$\mathcal{H}^*(?(C); S) = \begin{cases} M_C & \text{if } S = \text{end} \\ M_C\,, \mathcal{H}^*(S) & \text{otherwise} \end{cases}$$

where

$$M_C = ?(\mathcal{H}^*(C))$$

Handling Recursive Processes and Recursive Names

Consider process

$$R = \mu X.\, \underbrace{r?(z).}_{t_1}\, \underbrace{r!\langle -z\rangle.}_{t_2}\, \underbrace{r?(z).}_{t_3}\, \underbrace{r!\langle z\rangle.}_{t_4}\, X$$

where channel r implements the type

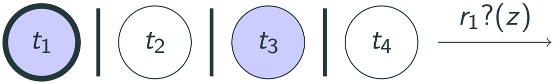
$$S = \mu \mathsf{t}.?(\mathsf{Int});!\langle \mathsf{Int}\rangle;\mathsf{t}$$

- Type S is decomposed into

$$S_1 = \mu \mathsf{t}.?(\mathsf{Int});\mathsf{t} \qquad S_2 = \mu \mathsf{t}.!\langle \mathsf{Int}\rangle;\mathsf{t}$$

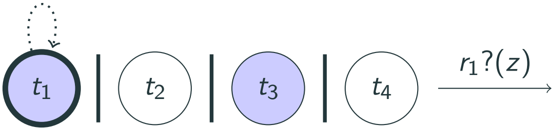
- Trios in $\mathcal{F}^*(R)$ must satisfy two properties:
  1. mimic recursive behaviour
  2. each instance should use the same decomposition of channel r, that is (r_1, r_2)

Handling Recursive Processes and Recursive Names, Intuitively

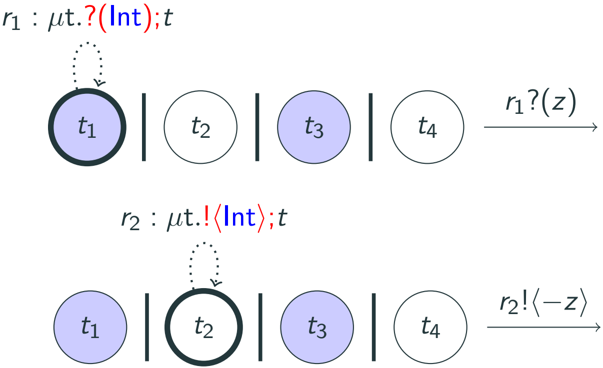


Handling Recursive Processes and Recursive Names, Intuitively

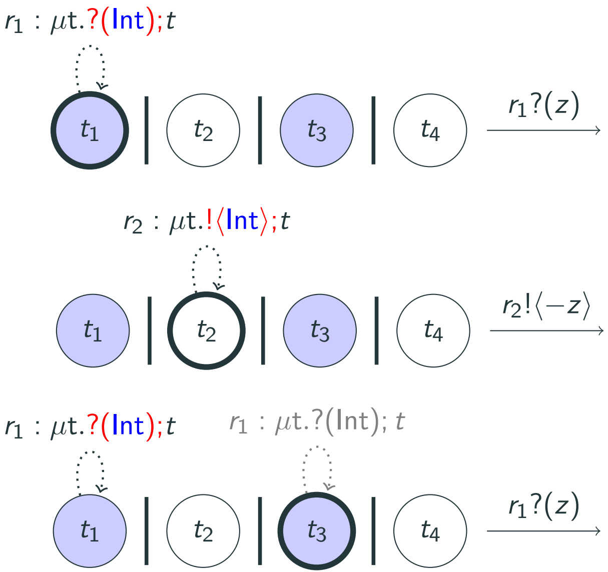
$r_1 : \mu \mathrm{t}.?(\mathrm{Int});t$



Handling Recursive Processes and Recursive Names, Intuitively

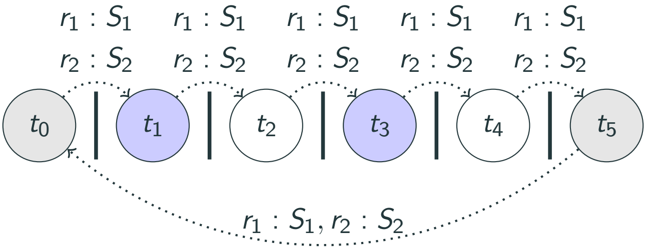


Handling Recursive Processes and Recursive Names, Intuitively



Handling Recursive Processes and Recursive Names, Intuitively

The trio structure for $R = \mu X.\, \underbrace{r?(z).}_{t_1}\, \underbrace{r!\langle -z\rangle.}_{t_2}\, \underbrace{r?(z).}_{t_3}\, \underbrace{r!\langle z\rangle.}_{t_4}\, X$ can be intuitively depicted as:



$$R = \mu X.\, \underbrace{r?(z).}_{t_1}\, \underbrace{r!\langle -z\rangle.}_{t_2}\, \underbrace{r?(z).}_{t_3}\, \underbrace{r!\langle z\rangle.}_{t_4}\, X$$

$\mathcal{F}^*(R)$ implements the circular structure of R using six recursive parallel processes:

$$\begin{array}{lr}
\overline{c_1^r}!\langle r_1, r_2\rangle.\mu X.\, c_X^r?(y_1, y_2).\, \overline{c_1^r}!\langle y_1, y_2\rangle.X \mid & t_0 \\
\mu X.c_1^r?(y_1, y_2).y_1?(z_1).\overline{c_2^r}!\langle y_1, y_2, z_1\rangle.X \mid & t_1 \\
\mu X.c_2^r?(y_1, y_2, z_1).y_2?(-z_1).\overline{c_3^r}!\langle y_1, y_2\rangle.X \mid & t_2 \\
\mu X.c_3^r?(y_1, y_2).y_1?(z_1).\overline{c_4^r}!\langle y_1, y_2, z_1\rangle.X \mid & t_3 \\
\mu X.c_4^r?(y_1, y_2, z_1).y_2?(z_1).\overline{c_5^r}!\langle y_1, y_2\rangle.X \mid & t_4 \\
\mu X.c_5^r?(y_1, y_2).\, \overline{c_X^r}!\langle y_1, y_2\rangle.\, X & t_5
\end{array}$$

Technical Results

- Quantifying improvements:

$$\text{number of prefixes in } \mathcal{F}(P) \geq \frac{5}{3} \cdot \text{ number of prefixes in } \mathcal{F}^*(P)$$

- Static correctness (Typability):

$$\Gamma \vdash P \text{ implies } \mathcal{H}^*(\Gamma) \vdash \mathcal{F}^*(P)$$

- Dynamic correctness:

$$P \approx^{\mathtt{M}} \mathcal{F}^*(P)$$

  where $\approx^{\mathtt{M}}$ is a form of weak bisimilarity, a mild modification of the **characteristic bisimilarity** by Kouzapas et al.

Conclusion

Related Work: Session Types into Linear Types (1/2)

Dardha, Giachino & Sangiorgi (PPDP'12) encode session-typed processes into processes with **linear types** (Kobayashi et al.):

- Sequentiality handled via a "detour" from session type theories
- Processes refactored to carry over sequentiality, in a continuation-passing style
- Implementations in Scala (Scalas et al. - ECOOP'16), OCaml (Padovani, JFP'17), Agda (Ciccone & Padovani, PPDP'20)

$\rightarrow$ **Differently**, our work clarifies the role of sequential composition in session types, both conceptually and formally, using session types themselves.

Related Work: A Comparison with Dardha et al. (2/2)

$$A = w!\langle \overline{u} \rangle.u?(a).u?(b).u!\langle a \geq b \rangle.\mathbf{0}$$

$\mathbb{A}^2_\epsilon(A\sigma')$

$$\begin{aligned}
&c_2?().\ w_1!\langle \overline{u}_1, \overline{u}_2, \overline{u}_3 \rangle.\ \overline{c_3}!\langle\rangle \mid \\
&\quad c_3?().\ u_1?(a).\ \overline{c_4}!\langle a \rangle \mid \\
&\qquad c_4?().\ u_2?(b).\ \overline{c_5}!\langle a, b \rangle \mid \\
&\qquad\quad c_5?().\ u_3!\langle a \geq b \rangle.\ \overline{c_6}!\langle\rangle \mid c_6?().\mathbf{0}
\end{aligned}$$

$[\![A]\!]_{w \mapsto z}$

$$\begin{aligned}
&(\nu\, c)z!\langle \overline{u}, c \rangle. \\
&\quad u?(a, c'). \\
&\qquad c'?(b, c''). \\
&\qquad\quad (\nu\, c''')c''!\langle a \geq b, c''' \rangle.\mathbf{0}
\end{aligned}$$

Minimal STs

$u_1 : ?(\mathsf{Int}),\ u_2 : ?(\mathsf{Int}),\ u_3 : !\langle \mathsf{Bool} \rangle$

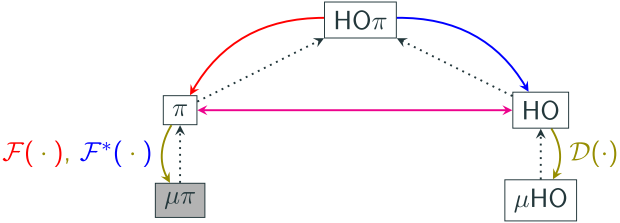
$w_1 : !\langle !\langle \mathsf{Int} \rangle, !\langle \mathsf{Int} \rangle, ?(\mathsf{Bool}) \rangle$

Linear Types

$u : l_i[\mathsf{Int}, l_i[\mathsf{Int}, l_o[\mathsf{Bool}, \mathsf{unit}]]]$

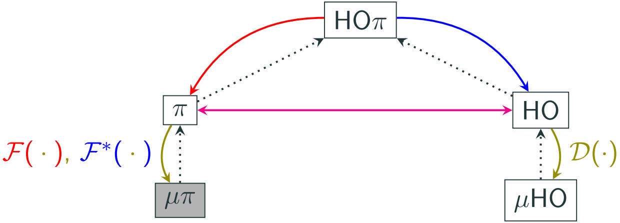
$w : l_o[l_o[\mathsf{Int}, l_o[\mathsf{Int}, l_i[\mathsf{Bool}, \mathsf{unit}]]], \mathsf{unit}]$

Conclusion: Minimal Session Types for π (1/2)



- A new minimality result for the session-typed π-calculus by two decompositions:
  1. $\mathcal{F}(\cdot)$: A composition of encodability results and minimality results for HO
  2. $\mathcal{F}^*(\cdot)$: An optimization without redundant synchronizations and with native recursion
- **Main takeaway**:
  The minimality result based on MSTs is independent from communicated objects:
  - abstractions in HO (ECOOP 2019)
  - names in π (This work)

Conclusion: Minimal Session Types for π (2/2)



- Potential for streamlining known session types frameworks, by removing redundancies.
- Bridging the gap between theories of session types and type systems in actual PLs.

In the Extended Version

- Full technical details
- Multiple examples of both decompositions
- `https://arxiv.org/abs/2107.10936`

Minimal Session Types for the π-calculus

PPDP 2021, Tallinn

Alen Arslanagić, Jorge A. Pérez, and Anda-Amelia Palamariuc

University of Groningen, The Netherlands

UNIFYING
CORRECTNESS FOR
COMMUNICATING
SOFTWARE

Extra Slides

Syntax

$$
\begin{aligned}
n &::= a, b \mid s, \bar{s} \\
u, w &::= n \mid x, y, z \\
V, W &::= \colorbox{lightgray}{u} \mid \boxed{\lambda x.\, P} \mid \boxed{x, y, z} \\
P, Q &::= u!\langle V \rangle.P \mid u?(x).P \\
&\quad \mid \boxed{V\ u} \mid P \mid Q \mid (\nu\, n)P \mid \mathbf{0} \mid \colorbox{lightgray}{$X \mid \mu X.P$}
\end{aligned}
$$

Figure 1: Syntax of HOπ. The sub-language HO lacks shaded constructs, while π lacks boxed constructs.

$$(\lambda x.\, P)\, u \longrightarrow P\{^{u}/_{x}\} \qquad \text{[App]}$$

$$n!\langle V\rangle.P \mid \overline{n}?(x).Q \longrightarrow P \mid Q\{^{V}/_{x}\} \qquad \text{[Pass]}$$

$$P \longrightarrow P' \Rightarrow (\nu\, n)P \longrightarrow (\nu\, n)P' \qquad \text{[Res]}$$

$$P \longrightarrow P' \Rightarrow P \mid Q \longrightarrow P' \mid Q \qquad \text{[Par]}$$

$$P \equiv Q \longrightarrow Q' \equiv P' \Rightarrow P \longrightarrow P' \qquad \text{[Cong]}$$

$$P_1 \mid P_2 \equiv P_2 \mid P_1 \quad P_1 \mid (P_2 \mid P_3) \equiv (P_1 \mid P_2) \mid P_3$$

$$P \mid \mathbf{0} \equiv P \quad P \mid (\nu\, n)Q \equiv (\nu\, n)(P \mid Q) \;\; (n \notin \mathtt{fn}(P))$$

$$(\nu\, n)\mathbf{0} \equiv \mathbf{0} \qquad \mu X.P \equiv P\{^{\mu X.P}/_{X}\} \qquad P \equiv Q \text{ if } P \equiv_\alpha Q$$

Figure 2: Operational Semantics of HOπ.

$$
\begin{aligned}
U &::= C \mid L & \quad C &::= S \mid \langle S \rangle \mid \langle L \rangle \\
L &::= U \rightarrow \diamond \mid U \multimap \diamond & \quad S &::= !\langle U \rangle; S \mid ?(U); S \\
& & & \mid \mu \mathtt{t}.S \mid \mathtt{t} \mid \mathtt{end}
\end{aligned}
$$

$$
\begin{aligned}
U &::= \widetilde{C} \rightarrow \diamond \mid \widetilde{C} \multimap \diamond & \quad C &::= M \mid \langle U \rangle \\
\gamma &::= \mathtt{end} \mid \mathtt{t} & \quad M &::= \gamma \mid !\langle \widetilde{U} \rangle; \gamma \mid ?(\widetilde{U}); \gamma \mid \mu \mathtt{t}.M
\end{aligned}
$$

Figure 3: STs for HOπ (top) and MSTs for HO (bottom).

Type encoding of π into HO

$$
\begin{aligned}
[\![u!\langle w\rangle.P]\!]^1_g &\stackrel{\text{def}}{=} u!\langle \lambda z.\ z?(x).(x\,w)\rangle.[\![P]\!]^1_g \\
[\![u?(x:C).Q]\!]^1_g &\stackrel{\text{def}}{=} u?(y).(\nu\, s)(y\,s \mid \overline{s}!\langle \lambda x.\ [\![Q]\!]^1_g\rangle.\mathbf{0}) \\
[\![P \mid Q]\!]^1_g &\stackrel{\text{def}}{=} [\![P]\!]^1_g \mid [\![Q]\!]^1_g \\
[\![(\nu\, n)P]\!]^1_g &\stackrel{\text{def}}{=} (\nu\, n)[\![P]\!]^1_g \\
[\![\mathbf{0}]\!]^1_g &\stackrel{\text{def}}{=} \mathbf{0} \\
[\![\mu X.P]\!]^1_g &\stackrel{\text{def}}{=} (\nu\, s)(\overline{s}!\langle V\rangle.\mathbf{0} \mid s?(z_X).[\![P]\!]^1_{g,\{X\to\tilde{n}\}}) \qquad \text{where } (\tilde{n} = \mathtt{fn}(P)) \\
& \qquad V = \lambda(\|\tilde{n}\|, y).\ y?(z_X).\llbracket [\![P]\!]^1_{g,\{X\to\tilde{n}\}} \rrbracket_\emptyset \\
[\![X]\!]^1_g &\stackrel{\text{def}}{=} (\nu\, s)(z_X\,(\tilde{n}, s) \mid \overline{s}!\langle z_X\rangle.\mathbf{0}) \quad (\tilde{n} = g(X))
\end{aligned}
$$

Figure 4: Typed encoding of π into HO, selection from [KPY19]. Above, $\mathtt{fn}(P)$ is a lexicographically ordered sequence of free names in P. Maps $\|\cdot\|$ and $\llbracket \cdot \rrbracket_\sigma$ are in Def. 1 and Fig. 5.

Definition (Auxiliary Mappings)

We define mappings $\| \cdot \|$ and $\llbracket \cdot \rrbracket_\sigma$ as follows:

- $\| \cdot \| : 2^{\mathcal{N}} \longrightarrow \mathcal{V}^\omega$ is a map of sequences of lexicographically ordered names to sequences of variables, defined inductively as:

$$\|\epsilon\| = \epsilon$$
$$\|n, \tilde{m}\| = x_n, \|\tilde{m}\| \quad (x \text{ fresh})$$

- Given a set of session names and variables σ, the map $\llbracket \cdot \rrbracket_\sigma : \mathsf{HO} \rightarrow \mathsf{HO}$ is as in Fig. 5.

Auxiliary Mapping

$$\llbracket w!\langle \lambda x.\, Q\rangle.P \rrbracket_{\sigma} \stackrel{\text{def}}{=} u!\langle \lambda x.\, \llbracket Q \rrbracket_{\sigma,x}\rangle.\llbracket P \rrbracket_{\sigma} \qquad \llbracket w \triangleright \{l_i : P_i\}_{i\in I} \rrbracket_{\sigma} \stackrel{\text{def}}{=} u \triangleright \{l_i : \llbracket P_i \rrbracket_{\sigma}\}_{i\in I}$$

$$\llbracket w?(x).P \rrbracket_{\sigma} \stackrel{\text{def}}{=} u?(x).\llbracket P \rrbracket_{\sigma} \qquad \llbracket w \triangleleft l.P \rrbracket_{\sigma} \stackrel{\text{def}}{=} u \triangleleft l.\llbracket P \rrbracket_{\sigma}$$

$$\llbracket (\nu\, n)P \rrbracket_{\sigma} \stackrel{\text{def}}{=} (\nu\, n)\llbracket P \rrbracket_{\sigma,n} \qquad \llbracket (\lambda x.Q)\, w \rrbracket_{\sigma} \stackrel{\text{def}}{=} (\lambda x.\llbracket Q \rrbracket_{\sigma,x})\, u$$

$$\llbracket P \mid Q \rrbracket_{\sigma} \stackrel{\text{def}}{=} \llbracket P \rrbracket_{\sigma} \mid \llbracket Q \rrbracket_{\sigma} \qquad \llbracket x\, w \rrbracket_{\sigma} \stackrel{\text{def}}{=} x\, u$$

$$\llbracket \mathbf{0} \rrbracket_{\sigma} \stackrel{\text{def}}{=} \mathbf{0}$$

$$\text{In all cases: } u = \begin{cases} x_n & \text{if } w \text{ is a name } n \text{ and } n \notin \sigma \ (x \text{ fresh}) \\ w & \text{otherwise: } w \text{ is a variable or a name } n \text{ and } n \in \sigma \end{cases}$$

Figure 5: Auxiliary mapping used to encode HOπ into HO.

Type encoding of π into HO

Types:

$$\lfloor S \rfloor^1 \stackrel{\text{def}}{=} (?(\langle\!\langle S \rangle\!\rangle^1 \multimap \diamond); \mathsf{end}) \multimap \diamond$$

$$\lfloor \langle S \rangle \rfloor^1 \stackrel{\text{def}}{=} (?(\langle \langle\!\langle S \rangle\!\rangle^1 \rangle \to \diamond); \mathsf{end}) \multimap \diamond$$

$$\langle\!\langle !\langle U \rangle; S \rangle\!\rangle^1 \stackrel{\text{def}}{=} !\langle \lfloor U \rfloor^1 \rangle; \langle\!\langle S \rangle\!\rangle^1$$

$$\langle\!\langle ?(U); S \rangle\!\rangle^1 \stackrel{\text{def}}{=} ?(\lfloor U \rfloor^1); \langle\!\langle S \rangle\!\rangle^1$$

$$\langle\!\langle \langle S \rangle \rangle\!\rangle^1 \stackrel{\text{def}}{=} \langle \langle\!\langle S \rangle\!\rangle^1 \rangle \qquad \langle\!\langle \mu \mathsf{t}.S \rangle\!\rangle^1 \stackrel{\text{def}}{=} \mu \mathsf{t}.\langle\!\langle S \rangle\!\rangle^1$$

$$\langle\!\langle \mathsf{end} \rangle\!\rangle^1 \stackrel{\text{def}}{=} \mathsf{end} \qquad \langle\!\langle \mathsf{t} \rangle\!\rangle^1 \stackrel{\text{def}}{=} \mathsf{t}$$

Typed encoding of HO into π

Terms:

$$[\![u!\langle \lambda x.\, Q\rangle.P]\!]^2 \stackrel{\text{def}}{=}$$

$$\begin{cases} (\nu\, a)(u!\langle a\rangle.([\![P]\!]^2 \mid *\; a?(y).y?(x).[\![Q]\!]^2)) & \text{if } \mathtt{fs}(Q) = \emptyset \\ (\nu\, a)(u!\langle a\rangle.([\![P]\!]^2 \mid a?(y).y?(x).[\![Q]\!]^2)) & \text{otherwise} \end{cases}$$

$$[\![u?(x).P]\!]^2 \stackrel{\text{def}}{=} u?(x).[\![P]\!]^2$$

$$[\![x\, u]\!]^2 \stackrel{\text{def}}{=} (\nu\, s)(x!\langle s\rangle.\overline{s}!\langle u\rangle.\mathbf{0})$$

$$[\![(\lambda x.\, P)\, u]\!]^2 \stackrel{\text{def}}{=} (\nu\, s)(s?(x).[\![P]\!]^2 \mid \overline{s}!\langle u\rangle.\mathbf{0})$$

Types:

$$(\!(!\langle S \multimap \diamond\rangle; S_1)\!)^2 \stackrel{\text{def}}{=} !\langle\langle ?((\!(S)\!)^2); \mathsf{end}\rangle\rangle; (\!(S_1)\!)^2$$

$$(\!(?(S \multimap \diamond); S_1)\!)^2 \stackrel{\text{def}}{=} ?(\langle ?((\!(S)\!)^2); \mathsf{end}\rangle); (\!(S_1)\!)^2$$

$$
\begin{aligned}
C &::= M \mid \langle M \rangle \\
\gamma &::= \mathtt{end} \mid \mathtt{t} \\
M &::= \gamma \mid !\langle \widetilde{C} \rangle;\gamma \mid ?(\widetilde{C});\gamma \mid \mu\mathtt{t}.M
\end{aligned}
$$

Figure 7: Minimal Session Types for π

Decomposition of types

$$\mathcal{H}(\langle S \rangle) = \langle \mathcal{H}(S) \rangle$$

$$\mathcal{H}(!\langle S \rangle; S') = \begin{cases} M & \text{if } S' = \text{end} \\ M, \mathcal{H}(S') & \text{otherwise} \end{cases}$$

$$\text{where } M = !\langle\langle ?(?(\langle ?(\mathcal{H}(S)); \text{end}\rangle); \text{end}); \text{end}\rangle\rangle; \text{end}$$

$$\mathcal{H}(?(S); S') = \begin{cases} M & \text{if } S' = \text{end} \\ M, \mathcal{H}(S') & \text{otherwise} \end{cases}$$

$$\text{where } M = ?(\langle ?(?(\langle ?(\mathcal{H}(S)); \text{end}\rangle); \text{end}); \text{end}\rangle); \text{end}$$

$$\mathcal{H}(\text{end}) = \text{end}$$

$$\mathcal{H}(S_1, \ldots, S_n) = \mathcal{H}(S_1), \ldots, \mathcal{H}(S_n)$$

Figure 8: Decomposition of types $\mathcal{H}(\,\cdot\,)$

Decomposition of types

$$\mathcal{H}(\mu t.S) = \begin{cases} \mathcal{R}'(S) & \text{if } \mu t.S \text{ is tail-recursive} \\ \mu t.\mathcal{H}(S) & \text{otherwise} \end{cases}$$

$$\mathcal{R}'(!\langle S\rangle; S') = \mu t.!\langle\langle ?(?(\langle ?(\mathcal{H}(S)); \text{end}\rangle); \text{end}); \text{end}\rangle\rangle; t, \mathcal{R}'(S')$$

$$\mathcal{R}'(?(S); S') = \mu t.?(\langle ?(?(\langle ?(\mathcal{H}(S)); \text{end}\rangle); \text{end}); \text{end}\rangle); t, \mathcal{R}'(S')$$

$$\mathcal{H}(t) = t \qquad \mathcal{R}'(t) = \epsilon$$

$$\mathcal{R}'^{\star}(?(S); S') = \mathcal{R}'^{\star}(S') \qquad \mathcal{R}'^{\star}(!\langle S\rangle; S') = \mathcal{R}'^{\star}(S')$$

$$\mathcal{R}'^{\star}(\mu t.S) = \mathcal{R}'^{\star}(S)$$

Figure 9: Decomposition of types $\mathcal{H}(\,\cdot\,)$

Decomposition of types Optimized

$$\mathcal{H}^*(\mathsf{end}) = \mathsf{end}$$

$$\mathcal{H}^*(\langle S \rangle) = \langle \mathcal{H}^*(S) \rangle$$

$$\mathcal{H}^*(S_1, \ldots, S_n) = \mathcal{H}^*(S_1), \ldots, \mathcal{H}^*(S_n)$$

$$\mathcal{H}^*(!\langle C \rangle; S) = \begin{cases} !\langle \mathcal{H}^*(C) \rangle; \mathsf{end} & \text{if } S = \mathsf{end} \\ !\langle \mathcal{H}^*(C) \rangle; \mathsf{end}\,, \mathcal{H}^*(S) & \text{otherwise} \end{cases}$$

$$\mathcal{H}^*(?(C); S) = \begin{cases} ?(\mathcal{H}^*(C)); \mathsf{end} & \text{if } S = \mathsf{end} \\ ?(\mathcal{H}^*(C)); \mathsf{end}\,, \mathcal{H}^*(S) & \text{otherwise} \end{cases}$$

Figure 10: Decomposition of types $\mathcal{H}^*(\,\cdot\,)$

Decomposition of types Optimized

$$
\begin{aligned}
\mathcal{H}^*(\mu \mathsf{t}.S') &= \mathcal{R}(S') \\
\mathcal{H}^*(S) &= \mathcal{R}^\star(S) \quad \text{where } S \neq \mu \mathsf{t}.S' \\
\mathcal{R}(\mathsf{t}) &= \epsilon \\
\mathcal{R}(!\langle C\rangle; S) &= \mu \mathsf{t}.!\langle \mathcal{H}^*(C)\rangle; \mathsf{t}, \mathcal{R}(S) \\
\mathcal{R}(?(C); S) &= \mu \mathsf{t}.?(\mathcal{H}^*(C)); \mathsf{t}, \mathcal{R}(S) \\
\mathcal{R}^\star(?(C); S) &= \mathcal{R}^\star(!\langle C\rangle; S) = \mathcal{R}^\star(S) \\
\mathcal{R}^\star(\mu \mathsf{t}.S) &= \mathcal{R}(S)
\end{aligned}
$$

Figure 11: Decomposition of types $\mathcal{H}^*(\,\cdot\,)$

Type System

(Sess)
$$\Gamma; \emptyset; \{u : S\} \vdash u \triangleright S$$

(Sh)
$$\Gamma, u : U; \emptyset; \emptyset \vdash u \triangleright U$$

(LVar)
$$\Gamma; \{x : C \multimap \diamond\}; \emptyset \vdash x \triangleright C \multimap \diamond$$

(RVar)
$$\Gamma, X : \Delta; \emptyset; \Delta \vdash X \triangleright \diamond$$

(Abs)
$$\frac{\Gamma; \Lambda; \Delta_1 \vdash P \triangleright \diamond \qquad \Gamma; \emptyset; \Delta_2 \vdash x \triangleright C}{\Gamma \backslash x; \Lambda; \Delta_1 \backslash \Delta_2 \vdash \lambda x. P \triangleright C \multimap \diamond}$$

(App)
$$\frac{\Gamma; \Lambda; \Delta_1 \vdash V \triangleright C \rightsquigarrow \diamond \qquad \rightsquigarrow \in \{\multimap, \rightarrow\} \qquad \Gamma; \emptyset; \Delta_2 \vdash u \triangleright C}{\Gamma; \Lambda; \Delta_1, \Delta_2 \vdash V\, u \triangleright \diamond}$$

(Prom)
$$\frac{\Gamma; \emptyset; \emptyset \vdash V \triangleright C \multimap \diamond}{\Gamma; \emptyset; \emptyset \vdash V \triangleright C \rightarrow \diamond}$$

(EProm)
$$\frac{\Gamma; \Lambda, x : C \multimap \diamond; \Delta \vdash P \triangleright \diamond}{\Gamma, x : C \rightarrow \diamond; \Lambda; \Delta \vdash P \triangleright \diamond}$$

(End)
$$\frac{\Gamma; \Lambda; \Delta \vdash P \triangleright T \qquad u \notin \mathtt{dom}(\Gamma, \Lambda, \Delta)}{\Gamma; \Lambda; \Delta, u : \mathtt{end} \vdash P \triangleright \diamond}$$

Type System

(Rec)

$$\frac{\Gamma, X : \Delta; \emptyset; \Delta \vdash P \triangleright \diamond}{\Gamma; \emptyset; \Delta \vdash \mu X.P \triangleright \diamond}$$

(Par)

$$\frac{\Gamma; \Lambda_i; \Delta_i \vdash P_i \triangleright \diamond \quad i = 1, 2}{\Gamma; \Lambda_1, \Lambda_2; \Delta_1, \Delta_2 \vdash P_1 \mid P_2 \triangleright \diamond}$$

(Nil)

$$\frac{}{\Gamma; \emptyset; \emptyset \vdash \mathbf{0} \triangleright \diamond}$$

(Send)

$$\frac{u : S \in \Delta_1, \Delta_2 \quad \Gamma; \Lambda_1; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \Lambda_2; \Delta_2 \vdash V \triangleright U}{\Gamma; \Lambda_1, \Lambda_2; ((\Delta_1, \Delta_2) \setminus u : S), u :!\langle U \rangle; S \vdash u!\langle V \rangle.P \triangleright \diamond}$$

(Rcv)

$$\frac{\Gamma; \Lambda_1; \Delta_1, u : S \vdash P \triangleright \diamond \quad \Gamma; \Lambda_2; \Delta_2 \vdash x \triangleright U}{\Gamma \setminus x; \Lambda_1 \setminus \Lambda_2; \Delta_1 \setminus \Delta_2, u :?(U); S \vdash u?(x).P \triangleright \diamond}$$

(Acc)

$$\frac{\begin{array}{c}\Gamma; \Lambda_1; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \emptyset; \emptyset \vdash u \triangleright \langle \mathcal{U} \rangle \\ \Gamma; \Lambda_2; \Delta_2 \vdash x \triangleright \mathcal{U} \quad \mathcal{U} \in \{S, L\}\end{array}}{\Gamma \setminus x; \Lambda_1 \setminus \Lambda_2; \Delta_1 \setminus \Delta_2 \vdash u?(x).P \triangleright \diamond}$$

(Acc)

$$\frac{\Gamma; \Lambda_1; \Delta_1 \vdash P \triangleright \diamond \quad \Gamma; \emptyset; \emptyset \vdash u \triangleright \langle \mathcal{U} \rangle \quad \Gamma; \Lambda_2; \Delta_2 \vdash x \triangleright \mathcal{U} \quad \mathcal{U} \in \{S, L\}}{\Gamma \backslash x; \Lambda_1 \backslash \Lambda_2; \Delta_1 \backslash \Delta_2 \vdash u?(x).P \triangleright \diamond}$$

(Bra)

$$\frac{\forall i \in I \quad \Gamma; \Lambda; \Delta, u : S_i \vdash P_i \triangleright \diamond}{\Gamma; \Lambda; \Delta, u : \&\{l_i : S_i\}_{i \in I} \vdash u \triangleright \{l_i : P_i\}_{i \in I} \triangleright \diamond}$$

(Sel)

$$\frac{\Gamma; \Lambda; \Delta, u : S_j \vdash P \triangleright \diamond \quad j \in I}{\Gamma; \Lambda; \Delta, u : \oplus\{l_i : S_i\}_{i \in I} \vdash u \triangleleft l_j.P \triangleright \diamond}$$

(ResS)

$$\frac{\Gamma; \Lambda; \Delta, s : S_1, \overline{s} : S_2 \vdash P \triangleright \diamond \quad S_1 \text{ dual } S_2}{\Gamma; \Lambda; \Delta \vdash (\nu\, s)P \triangleright \diamond}$$

(Res)

$$\frac{\Gamma, a : \langle S \rangle; \Lambda; \Delta \vdash P \triangleright \diamond}{\Gamma; \Lambda; \Delta \vdash (\nu\, a)P \triangleright \diamond}$$

Figure 14: Typing Rules for HOπ (including selection and branching constructs).

Minimal characteristic trigger process

Definition (Minimal characteristic processes)

$$\begin{aligned}
\langle ?(C); S\rangle_i^u &\stackrel{\text{def}}{=} u_i?(x).(t!\langle u_{i+1}, \ldots, u_{i+|\mathcal{G}(S)|}\rangle.\mathbf{0} \mid \langle C\rangle_i^x) \\
\langle !\langle C\rangle; S\rangle_i^u &\stackrel{\text{def}}{=} u_i!\langle\langle C\rangle_{\mathtt{c}}\rangle.t!\langle u_{i+1}, \ldots, u_{i+|\mathcal{G}(S)|}\rangle.\mathbf{0} \\
\langle \mathtt{end}\rangle_i^u &\stackrel{\text{def}}{=} \mathbf{0} \\
\langle\langle C\rangle\rangle_i^u &\stackrel{\text{def}}{=} u_1!\langle\langle C\rangle_{\mathtt{c}}\rangle.t!\langle u_1\rangle.\mathbf{0} \\
\langle \mu \mathtt{t}.S\rangle_i^u &\stackrel{\text{def}}{=} \langle S\{\mathtt{end}/\mathtt{t}\}\rangle_i^u \\
\langle S\rangle_{\mathtt{c}} &\stackrel{\text{def}}{=} \widetilde{s} \; (|\widetilde{s}| = |\mathcal{G}(S)|, \widetilde{s} \text{ fresh}) \\
\langle\langle C\rangle\rangle_{\mathtt{c}} &\stackrel{\text{def}}{=} a_1 \; (a_1 \text{ fresh})
\end{aligned}$$

Definition (Minimal characteristic trigger process)
Given a type C, the trigger process is

$$t \Leftarrow_{\mathtt{m}} v_i : C \stackrel{\text{def}}{=} t_1?(x).(\nu\, s_1)(s_1?(\widetilde{y}).\langle C\rangle_i^y \mid \overline{s_1}!\langle \tilde{v}\rangle.\mathbf{0})$$

MST-Bisimilarity

A typed relation $\Re$ is an *MST bisimulation* if for all $\Gamma_1; \Delta_1 \vdash P_1 \;\Re\; \Gamma_2; \Delta_2 \vdash Q_1$,

1. Whenever $\Gamma_1; \Delta_1 \vdash P_1 \xrightarrow{(\nu\, \widetilde{m_1})n!\langle v:C_1\rangle} \Delta_1'; \Lambda_1' \vdash P_2$ then there exist Q_2, Δ_2', and σ_v such that $\Gamma_2; \Delta_2 \vdash Q_1 \stackrel{(\nu\, \widetilde{m_2})\breve{n}!\langle \tilde{v}:\mathcal{H}^*(C)\rangle}{\Longrightarrow} \Delta_2' \vdash Q_2$ where $v\sigma_v \bowtie_{\mathsf{c}} \tilde{v}$ and, for a fresh t,
$$\Gamma; \Delta_1'' \vdash (\nu\, \widetilde{m_1})(P_2 \mid t \Leftarrow_{\mathtt{C}} v : C_1)\Re$$
$$\Delta_2'' \vdash (\nu\, \widetilde{m_2})(Q_2 \mid t \Leftarrow_{\mathtt{m}} v\sigma : C_1)$$
2. Whenever $\Gamma_1; \Delta_1 \vdash P_1 \xrightarrow{n?(v)} \Delta_1' \vdash P_2$ then there exist Q_2, Δ_2', and σ_v such that $\Gamma_2; \Delta_2 \vdash Q_1 \stackrel{\breve{n}?(\tilde{v})}{\Longrightarrow} \Delta_2' \vdash Q_2$ where $v\sigma_v \bowtie_{\mathsf{c}} \tilde{v}$ and $\Gamma_1; \Delta_1' \vdash P_2 \;\Re\; \Gamma_2; \Delta_2' \vdash Q_2$,
3. Whenever $\Gamma_1; \Delta_1 \vdash P_1 \xrightarrow{\ell} \Delta_1' \vdash P_2$, with ℓ not an output or input, then there exist Q_2 and Δ_2' such that $\Gamma_2; \Delta_2 \vdash Q_1 \stackrel{\hat{\ell}}{\Longrightarrow} \Delta_2' \vdash Q_2$ and $\Gamma_1; \Delta_1' \vdash P_2 \;\Re\; \Gamma_2; \Delta_2' \vdash Q_2$ and $\mathsf{sub}(\ell) = n$ implies $\mathsf{sub}(\hat{\ell}) = \breve{n}$.
4. The symmetric cases of 1, 2, and 3.

Results: Typability

Theorem (Typability of Breakdown)

Let P be an initialized π process. If $\Gamma; \Delta, \Delta_\mu \vdash P \triangleright \diamond$, then $\mathcal{H}(\Gamma'), \Phi'; \mathcal{H}(\Delta), \Theta' \vdash \mathcal{A}^k_\epsilon(P)_g \triangleright \diamond$, where $k > 0$; $\widetilde{r} = \mathtt{dom}(\Delta_\mu)$; $\Phi' = \prod_{r \in \widetilde{r}} c^r : \langle\langle ?(\mathcal{R}'^\star(\Delta_\mu(r))); \mathtt{end}\rangle\rangle$; and $\mathsf{balanced}(\Theta')$ *with*

$$\mathtt{dom}(\Theta') = \{c_k, c_{k+1}, \ldots, c_{k+\lfloor P \rceil - 1}\} \cup \{\overline{c_{k+1}}, \ldots, \overline{c_{k+\lfloor P \rceil - 1}}\}$$

such that $\Theta'(c_k) = ?(\cdot); \mathtt{end}$.

Theorem (Minimality Result for π)

Let P be a closed π process, with $\widetilde{u} = \mathtt{fn}(P)$ and $\widetilde{v} = \mathtt{rn}(P)$. If $\Gamma; \Delta, \Delta_\mu \vdash P \triangleright \diamond$, where Δ_μ only involves recursive session types, then
$\mathcal{H}(\Gamma\sigma); \mathcal{H}(\Delta\sigma), \mathcal{H}(\Delta_\mu\sigma) \vdash \mathcal{F}(P) \triangleright \diamond$, *where $\sigma = \{^{\mathsf{init}(\widetilde{u})}/_{\widetilde{u}}\}$.*

Optimized Results: Typability

Theorem (Typability of Breakdown)

Let P be an initialized process. If $\Gamma; \Delta \vdash P \triangleright \diamond$ then

$$\mathcal{H}^*(\Gamma \setminus \widetilde{x}); \mathcal{H}^*(\Delta \setminus \widetilde{x}), \Theta \vdash \mathbb{A}^k_{\widetilde{y}}(P) \triangleright \diamond \quad (k > 0)$$

where $\widetilde{x} \subseteq \texttt{fn}(P)$ and $\widetilde{y}$ such that $\texttt{indexed}_{\Gamma,\Delta}(\widetilde{y}, \widetilde{x})$. Also, balanced$(\Theta)$ *with*

$$dom(\Theta) = \{c_k, c_{k+1}, \ldots, c_{k+|P|-1}\} \cup \{\overline{c_{k+1}}, \ldots, \overline{c_{k+|P|-1}}\}$$

and $\Theta(c_k) = ?(\widetilde{M}); \mathtt{end}$, where $\widetilde{M} = (\mathcal{H}^*(\Gamma), \mathcal{H}^*(\Delta))(\widetilde{y})$.

Theorem (Minimality Result for π, Optimized)

Let P be a π process with $\widetilde{u} = \texttt{fn}(P)$. If $\Gamma; \Delta \vdash P \triangleright \diamond$ then $\mathcal{H}^*(\Gamma\sigma); \mathcal{H}^*(\Delta\sigma) \vdash \mathcal{F}^*(P) \triangleright \diamond$, where $\sigma = \{^{\mathsf{init}(\widetilde{u})}/_{\widetilde{u}}\}$.

Theorem (Operational Correspondence)

Let P be a π process such that $\Gamma_1; \Delta_1 \vdash P_1$. We have

$$\Gamma; \Delta \vdash P \;\approx^{\mathtt{M}}\; \mathcal{H}^*(\Gamma); \mathcal{H}^*(\Delta) \vdash \mathcal{F}^*(P)$$

Related Work: CPS Cont'd

P implements channel u of type $S = ?\mathsf{Int}; ?\mathsf{Int}; !\mathsf{Bool}; \mathtt{end}$:

$$P = (\nu\, u : S)(\underbrace{w!\langle \overline{u} \rangle.u?(a).u?(b).u!\langle a \geq b \rangle.\mathbf{0}}_{A} \mid \underbrace{\overline{w}?(x).x!\langle 5 \rangle.x!\langle 4 \rangle.x?(b).\mathbf{0}}_{B})$$

CPS encoding

$$[\![A]\!]_{w \mapsto z} = (\nu\, c)z!\langle u, c \rangle.\overline{u}?(a, c').c?(b, c'').(\nu\, c''')c''!\langle a \geq b, c''' \rangle.\mathbf{0}$$

$$[\![B]\!]_{w \mapsto z} = z?(x, c).(\nu\, c')x!\langle 5, c' \rangle.(\nu\, c'')c'!\langle 4, c'' \rangle.c''?(b, c''').\mathbf{0}$$

$$[\![S]\!] = l_i[\mathsf{Int}, l_i[\mathsf{Int}, l_o[\mathsf{Bool}, \mathsf{unit}]]]$$

$$[\![\overline{S}]\!] = l_o[\mathsf{Int}, l_i[\mathsf{Int}, l_o[\mathsf{Bool}, \mathsf{unit}]]]$$

References

Dimitrios Kouzapas, Jorge A. Pérez, and Nobuko Yoshida, *On the relative expressiveness of higher-order session processes*, Inf. Comput. **268** (2019).