Typed Non-determinism in Concurrent Calculi: The Eager Way

Daniele Nantes
Imperial College London (UK)

Joint work with
Joseph Paulus (University of Oxford, UK)
Bas van den Heuvel (Karlsruhe Univ. of Applied Sciences, GER)
Jorge A. Pérez (Univ. of Groningen, NL)

Mathematical Foundations in Program Semantics (MFPS)

Our Work

We explore the delicate interplay of non-determinism, and resource management (linearity!), across functional and concurrent programming calculi and under session types and intersection types disciplines.

Our Work

We explore the delicate interplay of non-determinism, and resource management (linearity!), across functional and concurrent programming calculi and under session types and intersection types disciplines.

A **session type** describes a protocol used by communicating processes.

e.g. $x : A$ means "x should conform to the protocol specified by the session type A"

Our Work

We explore the delicate interplay of non-determinism, and resource management (linearity!), across functional and concurrent programming calculi and under session types and intersection types disciplines.

A **session type** describes a protocol used by communicating processes.

e.g. $x : A$ means "x should conform to the protocol specified by the session type A"

Goals:

- To improve over prior works that used confluent non-determinism (FSCD21, TYPES21) and non-confluent non-determinism (APLAS'23), among others.
- To design expressive session typed π-calculi with non-deterministic choice, and that use types to control resources.

Our Work

We present:

- A π-calculus with standard (non-confluent) nondeterministic choice and failure behaviour featuring an *eager semantics*.
- A (session) type system which ensures *type preservation* and *deadlock-freedom* (processes never get stuck).
- An intersection-typed resource λ-calculus with non-deterministic fetching of resources from bags.
- A translation between these typed calculi with *loose* correctness results (type preservation, operational correspondence).

Non-determinism

Non-determinism is when reductions may introduce multiple behaviors.

Non-determinism

Non-determinism is when reductions may introduce multiple behaviors. Reductions may be confluent:

$$P_1 \longrightarrow Q_1 \;,\; P_2 \longrightarrow Q_2$$

then

$$P_1 + P_2 \longrightarrow Q_1 + P_2 \text{ and } P_1 + P_2 \longrightarrow P_1 + Q_2$$

Non-determinism

Non-determinism is when reductions may introduce multiple behaviors.
Reductions may be confluent:

$$P_1 \longrightarrow Q_1 \;,\; P_2 \longrightarrow Q_2$$

then

$$P_1 + P_2 \longrightarrow Q_1 + P_2 \text{ and } P_1 + P_2 \longrightarrow P_1 + Q_2$$

But standard non-determinism is non-confluent:

$$P + Q \longrightarrow P \text{ or } P + Q \longrightarrow Q$$

Motivation for Non-Confluence

- Non-confluent non-determinism is of undiscussed convenience in formal modeling. For instance, in specifications of distributed protocols commitment is essential.

Motivation for Non-Confluence

- Non-confluent non-determinism is of undiscussed convenience in formal modeling. For instance, in specifications of distributed protocols commitment is essential.
- Non-confluent non-deterministic choice is commonplace in verification frameworks such as mCRL2.

Motivation for Non-Confluence

- Non-confluent non-determinism is of undiscussed convenience in formal modeling. For instance, in specifications of distributed protocols commitment is essential.
- Non-confluent non-deterministic choice is commonplace in verification frameworks such as mCRL2.
- It is also relevant in functional calculi; a well-known framework is de'Liguoro and Piperno's (untyped) non-deterministic λ-calculus.

Motivation for Non-Confluence

- Non-confluent non-determinism is of undiscussed convenience in formal modeling. For instance, in specifications of distributed protocols commitment is essential.
- Non-confluent non-deterministic choice is commonplace in verification frameworks such as mCRL2.
- It is also relevant in functional calculi; a well-known framework is de'Liguoro and Piperno's (untyped) non-deterministic λ-calculus.
- **Challenge**: Interplay between non-confluent non-determinism and resource management (linearity).

Our Contributions

We study new concurrent and functional calculi with usual (non-confluent) forms of non-determinism.

- The concurrent calculus $s\pi^!$:
  A π-calculus with non-deterministic choice, governed by session types.
- The functional calculus $\lambda_{\mathbb{C}}$:
  A resource λ-calculus, governed by intersection types, in which non-determinism concerns fetching of resources from bags.
- A correct translation of $\lambda_{\mathbb{C}}$ into $s\pi^!$:
  Formal connections for non-determinism across paradigms.

Non-Determinism in $s\pi^!$

$P \mathbin{+\!\!\!\!+} Q$ denotes the non-deterministic choice between P and Q:
if one branch can perform a synchronisation, the other branch may be discarded if it cannot.

Non-Determinism in s$\pi^!$

$P \mathbin{+\!\!\!\!+} Q$ denotes the non-deterministic choice between P and Q:
if one branch can perform a synchronisation, the other branch may be discarded if it cannot.

Consider the usual reduction axiom for the (untyped) π-calculus:

$$(\overline{x}[z]; P_1 + M_1) \mid (x(y); P_2 + M_2) \longrightarrow P_1 \mid P_2\{z/y\}$$

Example: A Server and a Non-Deterministic Client

$$\mathrm{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l}\mathrm{buy} : s.\mathit{case}\left\{\begin{array}{l}\mathrm{card} : s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[] \ , \\ \mathrm{cash} : \overline{s}[\mathtt{movie}]; \overline{s}[]\end{array}\right\} \\ \mathrm{peek} : \overline{s}[\mathtt{trailer}]; \overline{s}[]\end{array}\right\}$$

$$\mathrm{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathrm{buy}; \overline{s}.\mathrm{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0 \ , \\ \mathbin{+\!\!\!\!+} \overline{s}.\mathrm{buy}; \overline{s}.\mathrm{cash}; s(\mathit{movie}); s(); 0 \\ \mathbin{+\!\!\!\!+} \overline{s}.\mathrm{peek}; s(\mathit{link}); s(); 0 \ ,\end{array}\right)$$

Example: A Server and a Non-Deterministic Client

$$\mathrm{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l}\mathrm{buy} : s.\mathit{case}\left\{\begin{array}{l}\mathrm{card} : s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[]\ , \\ \mathrm{cash} : \overline{s}[\mathtt{movie}]; \overline{s}[]\end{array}\right\} \\ \mathrm{peek} : \overline{s}[\mathtt{trailer}]; \overline{s}[]\end{array}\right\}$$

$$\mathrm{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathrm{buy}; \overline{s}.\mathrm{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0\ , \\ \# \overline{s}.\mathrm{buy}; \overline{s}.\mathrm{cash}; s(\mathit{movie}); s(); 0 \\ \# \overline{s}.\mathrm{peek}; s(\mathit{link}); s(); 0\ ,\end{array}\right)$$

Example: A Server and a Non-Deterministic Client

$$\mathrm{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l}\mathsf{buy} : s.\mathit{case}\left\{\begin{array}{l}\mathsf{card} : s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[\,]\ , \\ \mathsf{cash} : \overline{s}[\mathtt{movie}]; \overline{s}[\,]\end{array}\right\} \\ \mathsf{peek} : \overline{s}[\mathtt{trailer}]; \overline{s}[\,]\end{array}\right\}$$

$$\mathrm{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathsf{buy}; \overline{s}.\mathsf{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0\ , \\ \nparallel \overline{s}.\mathsf{buy}; \overline{s}.\mathsf{cash}; s(\mathit{movie}); s(); 0 \\ \nparallel \overline{s}.\mathsf{peek}; s(\mathit{link}); s(); 0\ ,\end{array}\right)$$

Example: A Server and a Non-Deterministic Client

$$\mathsf{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l}\mathsf{buy} : s.\mathit{case}\left\{\begin{array}{l}\mathsf{card} : s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[]\ , \\ \mathsf{cash} : \overline{s}[\mathtt{movie}]; \overline{s}[]\end{array}\right\} \\ \mathsf{peek} : \overline{s}[\mathtt{trailer}]; \overline{s}[]\end{array}\right\}$$

$$\mathsf{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathsf{buy}; \overline{s}.\mathsf{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0\ , \\ \mathbin{+\!\!\!\!+} \overline{s}.\mathsf{buy}; \overline{s}.\mathsf{cash}; s(\mathit{movie}); s(); 0 \\ \mathbin{+\!\!\!\!+} \overline{s}.\mathsf{peek}; s(\mathit{link}); s(); 0\ ,\end{array}\right)$$

$$(\boldsymbol{\nu} s)(\mathsf{MovieS}_s \mid \mathsf{MovieC}_s) \longrightarrow^*$$

Example: A Server and a Non-Deterministic Client

$$\mathsf{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l}\mathsf{buy} : s.\mathit{case}\left\{\begin{array}{l}\mathsf{card} : s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[]\ , \\ \mathsf{cash} : \overline{s}[\mathtt{movie}]; \overline{s}[]\end{array}\right\} \\ \mathsf{peek} : \overline{s}[\mathtt{trailer}]; \overline{s}[]\end{array}\right\}$$

$$\mathsf{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathsf{buy}; \overline{s}.\mathsf{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0\ , \\ \mathbin{+\!\!\!\!+}\ \overline{s}.\mathsf{buy}; \overline{s}.\mathsf{cash}; s(\mathit{movie}); s(); 0 \\ \mathbin{+\!\!\!\!+}\ \overline{s}.\mathsf{peek}; s(\mathit{link}); s(); 0\ ,\end{array}\right)$$

$$(\boldsymbol{\nu} s)(\mathsf{MovieS}_s \mid \mathsf{MovieC}_s) \longrightarrow^* \begin{array}{l}(\boldsymbol{\nu} s)(\overline{s}[\mathtt{trailer}]; \overline{s}[] \\ \qquad \mid s(\mathit{link}); s(); 0)\end{array}$$

Example: A Server and a Non-Deterministic Client

$$\mathrm{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l}\mathrm{buy}: s.\mathit{case}\left\{\begin{array}{l}\mathrm{card}: s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[\,]\ ,\\ \mathrm{cash}: \overline{s}[\mathtt{movie}]; \overline{s}[\,]\end{array}\right\}\\ \mathrm{peek}: \overline{s}[\mathtt{trailer}]; \overline{s}[\,]\end{array}\right\}$$

$$\mathrm{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathrm{buy}; \overline{s}.\mathrm{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0\ ,\\ \mathbin{+\!\!\!\!+} \overline{s}.\mathrm{buy}; \overline{s}.\mathrm{cash}; s(\mathit{movie}); s(); 0\\ \mathbin{+\!\!\!\!+} \overline{s}.\mathrm{peek}; s(\mathit{link}); s(); 0\ ,\end{array}\right)$$

$$(\boldsymbol{\nu} s)(\mathrm{MovieS}_s \mid \mathrm{MovieC}_s) \longrightarrow^* \begin{array}{l}(\boldsymbol{\nu} s)(s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[\,]\\ \qquad\qquad \mid \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0)\end{array}$$

Example: A Server and a Non-Deterministic Client

$$\mathsf{MovieS}_s := s(\mathit{title}); s.\mathit{case}\left\{\begin{array}{l} \mathsf{buy} : s.\mathit{case}\left\{\begin{array}{l}\mathsf{card} : s(\mathit{info}); \overline{s}[\mathtt{movie}]; \overline{s}[]\ , \\ \mathsf{cash} : \overline{s}[\mathtt{movie}]; \overline{s}[]\end{array}\right\} \\ \mathsf{peek} : \overline{s}[\mathtt{trailer}]; \overline{s}[] \end{array}\right\}$$

$$\mathsf{MovieC}_s := \overline{s}[\mathtt{Barbie}]; \left(\begin{array}{l}\overline{s}.\mathsf{buy}; \overline{s}.\mathsf{card}; \overline{s}[\mathtt{visa}]; s(\mathit{movie}); s(); 0\ , \\ \mathbin{+\!\!\!\!+}\ \overline{s}.\mathsf{buy}; \overline{s}.\mathsf{cash}; s(\mathit{movie}); s(); 0 \\ \mathbin{+\!\!\!\!+}\ \overline{s}.\mathsf{peek}; s(\mathit{link}); s(); 0\ , \end{array}\right)$$

$$(\boldsymbol{\nu} s)(\mathsf{MovieS}_s \mid \mathsf{MovieC}_s) \longrightarrow^* \begin{array}{l}(\boldsymbol{\nu} s)(\overline{s}[\mathtt{movie}]; \overline{s}[] \\ \qquad\qquad \mid s(\mathit{movie}); s(); 0)\end{array}$$

Our New Calculus $s\pi^!$ (Excerpt)

$$
\begin{array}{rlcl}
P, Q ::= 0 & & \mid & [x \leftrightarrow y] \\
\mid & (\boldsymbol{\nu} x)(P \mid Q) & \mid & P \mathbin{\#} Q \\
\mid & \overline{x}[y]; (P \mid Q) & \mid & x(y); P \\
\mid & \overline{x}.\ell; P & \mid & x.\mathit{case}\{i : P\}_{i \in I} \\
\mid & \overline{x}[] & \mid & x(); P \\
\mid & x.\mathtt{some}_{w_1,\ldots,w_n}; P & \mid & \overline{x}.\mathtt{some}; P \\
\mid & P \mid Q & \mid & \overline{x}.\mathtt{none}
\end{array}
$$

Contexts

- ND-contexts ($\mathtt{N}, \mathtt{M}$):

$$\mathtt{N}, \mathtt{M} ::= [\cdot] \;\mid\; \mathtt{N} \mid P \;\mid\; (\boldsymbol{\nu} x)(\mathtt{N} \mid P) \;\mid\; \mathtt{N} \mathbin{+\!\!\!\!+} P$$

- The *commitment* of an ND-context $\mathtt{N}$:

$$(\![[\cdot]]\!) := [\cdot] \quad (\![\mathtt{N} \mid P]\!) := (\![\mathtt{N}]\!) \mid P \quad (\![(\boldsymbol{\nu} x)(\mathtt{N} \mid P)]\!) := (\boldsymbol{\nu} x)((\![\mathtt{N}]\!) \mid P)$$
$$(\![\mathtt{N} \mathbin{+\!\!\!\!+} P]\!) := (\![\mathtt{N}]\!)$$

Key Reduction Rules in $s\pi^!$

$$[\rightarrow_{\mathsf{Id}}] \quad (\boldsymbol{\nu} x)(\mathtt{N}[[x \leftrightarrow y]] \mid Q) \longrightarrow (\!|\mathtt{N}|\!)[Q\{y/x\}]$$

$$[\rightarrow_{\otimes⅋}] \quad \begin{array}{c} (\boldsymbol{\nu} x)(\mathtt{N}[\overline{x}[y];(P \mid Q)] \mid \mathtt{N}'[x(z);R]) \longrightarrow \\ (\!|\mathtt{N}|\!)[(\boldsymbol{\nu} x)(Q \mid (\boldsymbol{\nu} y)(P \mid (\!|\mathtt{N}'|\!)[R\{y/z\}]))] \end{array}$$

$$[\rightarrow_{\oplus\&}] \quad \begin{array}{c} \forall k' \in K.\ (\boldsymbol{\nu} x)(\mathtt{N}[\overline{x}.k';P] \mid \mathtt{N}'[x.case\{k : Q^k\}_{k \in K}]) \longrightarrow \\ (\boldsymbol{\nu} x)((\!|\mathtt{N}|\!)[P] \mid (\!|\mathtt{N}'|\!)[Q^{k'}]) \end{array}$$

$$[\rightarrow_{\nu}] \quad \frac{P \longrightarrow P'}{(\boldsymbol{\nu} x)(P \mid Q) \longrightarrow (\boldsymbol{\nu} x)(P' \mid Q)} \qquad [\rightarrow_{|}] \quad \frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q}$$

$$[\rightarrow_{\#}] \quad \frac{P \longrightarrow P'}{P \# Q \longrightarrow P' \# Q}$$

Session Types for s$\pi^!$

Session types in linear logic form ('propositions-as-sessions'):

$$
\begin{aligned}
A, B ::= 1 \mid \bot \mid A \otimes B \mid A \mathbin{⅋} B \mid ?A \mid !A \\
\mid \oplus\{i : A\}_{i \in I} \mid \&\{i : A\}_{i \in I} \mid \&A \mid \oplus A
\end{aligned}
$$

Judgments are of the form:

$$P \vdash \Gamma$$

Typing rules for non-determinism and failure:

$$[\mathrm{T}\#]\ \frac{P \vdash \Gamma \qquad Q \vdash \Gamma}{P \mathbin{\#} Q \vdash \Gamma} \qquad [\mathrm{T}\&\mathtt{some}]\ \frac{P \vdash \Gamma, x{:}A}{\overline{x}.\mathtt{some}; P \vdash \Gamma, x{:}\&A}$$

$$[\mathrm{T}\&\mathtt{none}]\ \frac{}{\overline{x}.\mathtt{none} \vdash x{:}\&A} \qquad [\mathrm{T}\oplus\mathtt{some}]\ \frac{P \vdash \&\Gamma, x{:}A}{x.\mathtt{some}_{dom(\Gamma)}; P \vdash \&\Gamma, x{:}\oplus A}$$

Non-deterministic Resource λ-calculus: $\lambda_{\mathbb{C}}$

$$
\begin{aligned}
M, N, L ::= {} & x[*] \mid (M\ B) \mid \lambda x.M \mid M[\widetilde{x} \leftarrow x] \mid M\langle\!\langle B/x \rangle\!\rangle \mid M\langle\!| C/\widetilde{x} |\!\rangle \mid M\llbracket U/x \rrbracket \mid \mathtt{fail}^{\widetilde{x}} \\
[*] ::= {} & [l] \mid [i] \quad i \in \mathbb{N} \\
A, B ::= {} & C \star U \\
U, V ::= {} & 1^{!} \mid \wr M \int^{!} \mid U \diamond V \\
C, D ::= {} & 1 \mid \wr M \int \cdot C \\
\mathcal{C} ::= {} & [\cdot] \mid (\mathcal{C}\ B) \mid \mathcal{C}\langle\!| C/\widetilde{x} |\!\rangle \mid \mathcal{C}\llbracket U/x \rrbracket \mid \mathcal{C}[\widetilde{x} \leftarrow x]
\end{aligned}
$$

Reduction in λ_{C}, by Example

Consider the following reductions, where $I = \lambda x.(x_1[x_1 \leftarrow x])$.

$$(\lambda x.x_1 \wr x_2 \wr x_3\ 1 \int \int[\widetilde{x} \leftarrow x]) \wr \mathtt{fail}^{\emptyset}, y, I \int$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3\ 1 \int \int[\widetilde{x} \leftarrow x])\langle\!\langle \wr \mathtt{fail}^{\emptyset}, y, I \int / x \rangle\!\rangle$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, y, I \int / x_1, x_2, x_3 |\!\rangle = M$$

$$\begin{array}{ll} & \nearrow (\mathtt{fail}^{\emptyset} \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr y, I \int / x_2, x_3 |\!\rangle = N_1 \\ M & \longrightarrow (y \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, I \int / x_2, x_3 |\!\rangle = N_2 \\ & \searrow (I \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, y \int / x_2, x_3 |\!\rangle = N_3 \end{array}$$

Reduction in λ_C, by Example

Consider the following reductions, where $I = \lambda x.(x_1[x_1 \leftarrow x])$.

$$(\lambda x.x_1 \wr x_2 \wr x_3\ 1 \int \int [\widetilde{x} \leftarrow x])\ \wr \mathtt{fail}^{\emptyset}, y, I \int$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3\ 1 \int \int [\widetilde{x} \leftarrow x])\langle\langle \wr \mathtt{fail}^{\emptyset}, y, I \int / x \rangle\rangle$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, y, I \int / x_1, x_2, x_3 |\!\rangle = M$$

$$M \begin{array}{l} \nearrow (\mathtt{fail}^{\emptyset} \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr y, I \int / x_2, x_3 |\!\rangle = N_1 \\ \longrightarrow (y \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, I \int / x_2, x_3 |\!\rangle = N_2 \\ \searrow (I \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, y \int / x_2, x_3 |\!\rangle = N_3 \end{array}$$

Reduction in λ_C, by Example

Consider the following reductions, where $I = \lambda x.(x_1[x_1 \leftarrow x])$.

$$(\lambda x. x_1 \wr x_2 \wr x_3 \; 1 \int \int [\widetilde{x} \leftarrow x]) \wr \mathtt{fail}^{\emptyset}, y, I \int$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3 \; 1 \int \int [\widetilde{x} \leftarrow x]) \langle\langle \wr \mathtt{fail}^{\emptyset}, y, I \int / x \rangle\rangle$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3 \; 1 \int \int) \langle\!| \wr \mathtt{fail}^{\emptyset}, y, I \int / x_1, x_2, x_3 |\!\rangle = M$$

$$\begin{array}{rl} & \nearrow (\mathtt{fail}^{\emptyset} \wr x_2 \wr x_3 \; 1 \int \int) \langle\!| \wr y, I \int / x_2, x_3 |\!\rangle = N_1 \\ M & \longrightarrow (y \wr x_2 \wr x_3 \; 1 \int \int) \langle\!| \wr \mathtt{fail}^{\emptyset}, I \int / x_2, x_3 |\!\rangle = N_2 \\ & \searrow (I \wr x_2 \wr x_3 \; 1 \int \int) \langle\!| \wr \mathtt{fail}^{\emptyset}, y \int / x_2, x_3 |\!\rangle = N_3 \end{array}$$

Reduction in λ_{C}, by Example

Consider the following reductions, where $I = \lambda x.(x_1[x_1 \leftarrow x])$.

$$(\lambda x.x_1 \wr x_2 \wr x_3\ 1 \int \int [\widetilde{x} \leftarrow x]) \wr \mathtt{fail}^{\emptyset}, y, I \int$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3\ 1 \int \int [\widetilde{x} \leftarrow x])\langle\!\langle \wr \mathtt{fail}^{\emptyset}, y, I \int / x \rangle\!\rangle$$

$$\downarrow$$

$$(x_1 \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, y, I \int / x_1, x_2, x_3 |\!\rangle = M$$

$$M \begin{array}{l} \nearrow\ (\mathtt{fail}^{\emptyset} \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr y, I \int / x_2, x_3 |\!\rangle = N_1 \\ \longrightarrow\ (y \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, I \int / x_2, x_3 |\!\rangle = N_2 \\ \searrow\ (I \wr x_2 \wr x_3\ 1 \int \int)\langle\!| \wr \mathtt{fail}^{\emptyset}, y \int / x_2, x_3 |\!\rangle = N_3 \end{array}$$

Some Reduction Rules for λ_{C}

$$[\texttt{RS:Beta}]\quad \frac{}{(\lambda x.M)\ B \longrightarrow M\langle\!\langle B/x\rangle\!\rangle}$$

$$[\texttt{RS:Ex-Sub}]\quad \frac{\mathsf{size}(C) = |\widetilde{x}| \qquad M \neq \texttt{fail}^{\tilde{y}}}{(M[\widetilde{x} \leftarrow x])\langle\!\langle C \star U/x\rangle\!\rangle \longrightarrow M\langle\!| C/\widetilde{x} |\!\rangle \llbracket U/x \rrbracket}$$

$$[\texttt{RS:Fetch}^{\ell}]\quad \frac{\mathsf{head}(M) = x_j \qquad 0 < i \leq \mathsf{size}(C)}{M\langle\!| C/\widetilde{x}, x_j |\!\rangle \longrightarrow (M\{C_i/x_j\})\langle\!|(C \setminus C_i)/\widetilde{x}|\!\rangle}$$

$$[\texttt{RS:Fail}^{\ell}]\quad \frac{\mathsf{size}(C) \neq |\widetilde{x}| \qquad \widetilde{y} = (\mathsf{lfv}(M) \setminus \{\widetilde{x}\}) \cup \mathsf{lfv}(C)}{(M[\widetilde{x} \leftarrow x])\langle\!\langle C \star U/x\rangle\!\rangle \longrightarrow \texttt{fail}^{\widetilde{y}}}$$

$$[\texttt{RS:Fetch}^{!}]\quad \frac{\mathsf{head}(M) = x[i] \qquad U_i = \wr N \int^{!}}{M\llbracket U/x \rrbracket \longrightarrow M\{N/x[i]\}\llbracket U/x \rrbracket}$$

$$[\texttt{RS:Fail}^{!}]\quad \frac{\mathsf{head}(M) = x[i] \qquad U_i = 1^{!}}{M\llbracket U/x \rrbracket \longrightarrow M\{\texttt{fail}^{\emptyset}/x[i]\}\llbracket U/x \rrbracket}$$

Key Typing Rules

Strict types (σ, τ, δ) and multiset types (π, ζ) are defined as follows:

$$\sigma, \tau, \delta ::= \mathbf{unit} \mid \pi \to \sigma \qquad \pi, \zeta ::= \bigwedge_{i \in I} \sigma_i \mid \omega$$

$$\eta, \epsilon ::= \sigma \mid \epsilon \diamond \eta \qquad (\pi, \eta)$$

(list) (tuple)

Linear and Unrestricted Type contexts:

$$\Gamma, \Delta ::= \text{-} \mid \Gamma, x : \pi \mid \Gamma, x : \sigma$$

$$\Theta, \Upsilon ::= \text{-} \mid \Theta, x^! : \eta$$

Judgments:

$$\Gamma \vDash M : \tau \quad \Gamma \vDash B : \pi$$

Translation: Key Ideas

A translation of $\lambda_{\mathtt{C}}$ into $\mathsf{s}\pi^{!}$ is insightful as:

- It provides a formal connection of (fail-prone) programs to (fail-prone) interactive processes.
- Relates intersection types into session types.
- Shows how non-confluent non-deterministic functional behavior may be expressed as session-typed protocols in the π-calculus

Translation of Terms

$$
\begin{aligned}
[\![x]\!]_u &= \overline{x}.\mathtt{some}; [x \leftrightarrow u] \\
[\![\lambda x.M]\!]_u &= \overline{u}.\mathtt{some}; u(x); [\![M]\!]_u \\
[\![(M\ C)]\!]_u &= (\boldsymbol{\nu} v)([\![M]\!]_v \mid v.\mathtt{some}_{u,\mathsf{lfv}(C)}; \overline{v}[x]; ([\![C]\!]_x \mid [v \leftrightarrow u])) \\
[\![M \langle\!\langle C/x \rangle\!\rangle]\!]_u &= (\boldsymbol{\nu} x)([\![M]\!]_u \mid [\![C]\!]_x)
\end{aligned}
$$

Translation of Terms

Non-deterministic fetch (λ_C) codified as non-deterministic choice ($s\pi^!$):

$$\llbracket M \langle\!| \wr N_1, N_2 \int / x_1, x_2 |\!\rangle \rrbracket_u = (\boldsymbol{\nu} z_1)(z_1.\mathtt{some}_{\mathsf{fv}(N_1)}; \llbracket N_1 \rrbracket_{z_1} \mid (\boldsymbol{\nu} z_2)(z_2.\mathtt{some}_{\mathsf{fv}(N_2)}; \llbracket N_2 \rrbracket_{z_2} \mid \mathbin{+\!\!\!\!\parallel}_{x_i \in \{x_1, x_2\}} \mathbin{+\!\!\!\!\parallel}_{x_j \in \{x_1, x_2 \setminus x_i\}} \llbracket M \rrbracket_u \{z_1/x_i\}\{z_2/x_j\}))$$

$$\llbracket M[\widetilde{x} \leftarrow x] \rrbracket_u = \overline{x}.\mathtt{some}; \overline{x}[y_i]; \big(y_i.\mathtt{some}_{\emptyset}; y_i(); 0 \mid \overline{x}.\mathtt{some}; x.\mathtt{some}_{u, \mathsf{fv}(M) \setminus \widetilde{x}}; \mathbin{+\!\!\!\!\parallel}_{x_i \in \widetilde{x}} x(x_i); \llbracket M[(\widetilde{x} \setminus x_i) \leftarrow x] \rrbracket_u \big)$$

$$\llbracket \mathtt{fail}^{x_1, \ldots, x_k} \rrbracket_u = \overline{u}.\mathtt{none} \mid \overline{x_1}.\mathtt{none} \mid \ldots \mid \overline{x_k}.\mathtt{none}$$

Translation of Types

Session types give a precise, protocol-oriented abstraction of functional resources:

$$[\![\mathrm{unit}]\!] = \&1 \qquad [\![\sigma^k \to \tau]\!] = \&(\overline{[\![\sigma^k]\!]_{(\sigma,i)}} \mathbin{⅋} [\![\tau]\!])$$

$$[\![\sigma \wedge \pi]\!]_{(\tau,i)} = \oplus((\&1) \mathbin{⅋} (\oplus \& ((\oplus[\![\sigma]\!]) \otimes ([\![\pi]\!]_{(\tau,i)}))))$$

$$[\![\omega]\!]_{(\sigma,i)} = \begin{cases} \oplus((\&1) \mathbin{⅋} (\oplus \& 1)) & \text{if } i = 0 \\ \oplus((\&1) \mathbin{⅋} (\oplus \& ((\oplus[\![\sigma]\!]) \otimes ([\![\omega]\!]_{(\sigma,i-1)})))) & \text{if } i > 0 \end{cases}$$

Dynamic Correctness

$$\frac{}{P \succeq_{\parallel} P} \qquad \frac{P_i \succeq_{\parallel} P_i' \qquad {\scriptstyle i \in \{1,2\}}}{P_1 \mathbin{\#} P_2 \succeq_{\parallel} P_i'} \qquad \frac{P \succeq_{\parallel} P' \qquad Q \succeq_{\parallel} Q'}{P \mid Q \succeq_{\parallel} P' \mid Q'}$$

$$\frac{P \succeq_{\parallel} P'}{(\boldsymbol{\nu} x)P \succeq_{\parallel} (\boldsymbol{\nu} x)P'}$$

Intuitively, $P \succeq_{\parallel} Q$ says that P has at least as many branches as Q.

Dynamic Correctness

(Loose Completeness)

If $N \longrightarrow M$ for a well-formed closed λ_{C}-term N, then there exists Q such that $[\![N]\!]_u \longrightarrow^* Q$ and $[\![M]\!]_u \succeq_{\Vert} Q$.

Dynamic Correctness

(Loose Completeness)

If $N \longrightarrow M$ for a well-formed closed λ_{C}-term N, then there exists Q such that $[\![N]\!]_u \longrightarrow^* Q$ and $[\![M]\!]_u \succeq_{\Vert} Q$.

(Loose Weak Soundness)

If $[\![N]\!]_u \longrightarrow^* Q$ for a well-formed closed λ_{C}-term N, then there exist N' and Q' such that
(i) $N \longrightarrow^* N'$ and (ii) $Q \longrightarrow^* Q'$ with $[\![N']\!]_u \succeq_{\Vert} Q'$.

Dynamic Correctness

(Loose Completeness)

If $N \longrightarrow M$ for a well-formed closed $\lambda_{\mathtt{C}}$-term N, then there exists Q such that $[\![N]\!]_u \longrightarrow^* Q$ and $[\![M]\!]_u \succeq_{\Vert} Q$.

(Loose Weak Soundness)

If $[\![N]\!]_u \longrightarrow^* Q$ for a well-formed closed $\lambda_{\mathtt{C}}$-term N, then there exist N' and Q' such that
(i) $N \longrightarrow^* N'$ and (ii) $Q \longrightarrow^* Q'$ with $[\![N']\!]_u \succeq_{\Vert} Q'$.

(Success Sensitivity)

$M \Downarrow \checkmark_\lambda$ iff $[\![M]\!]_u \Downarrow \checkmark_\pi$ for well-formed closed terms M.

Summary of Technical Results

Results in $s\pi^!$

Theorem (Type Preservation)

Theorem (Deadlock-freedom)

Results in λ_C

Theorem (SR in λ_C)

Theorem (SE in λ_C)

Translation correctness from λ_C to $s\pi^!$

Theorem (Translation Preserves Types)

Theorem (Translation correctness under $\longrightarrow$)

Summary of Technical Results

Results in $s\pi^!$

Theorem (Type Preservation)

If $P \vdash \Gamma$, then both $P \equiv Q$ and $P \longrightarrow Q$ imply $Q \vdash \Gamma$.

Theorem (Deadlock-freedom)

If $P \vdash \emptyset$ and $P \not\equiv 0$, then there is R such that $P \longrightarrow R$.

Results in λ_{C}

Theorem (SR in λ_{C})

Theorem (SE in λ_{C})

Translation correctness from λ_{C} to $s\pi^!$

Theorem (Translation Preserves Types)

Theorem (Translation correctness under $\longrightarrow$)

Summary of Technical Results

Results in $s\pi^!$

Theorem (Type Preservation)

Theorem (Deadlock-freedom)

Results in λ_C

Theorem (SR in λ_C)

If $\Theta; \Gamma \models M : \tau$ and $M \longrightarrow M'$, then $\Theta; \Gamma \models M' : \tau$.

Theorem (SE in λ_C)

If $\Theta; \Gamma \vdash M' : \tau$ and $M \longrightarrow M'$, then $\Theta; \Gamma \vdash M : \tau$.

Translation correctness from λ_C to $s\pi^!$

Theorem (Translation Preserves Types)

Theorem (Translation correctness under $\longrightarrow$)

Summary of Technical Results

Results in $s\pi^!$

Theorem (Type Preservation)

Theorem (Deadlock-freedom)

Results in λ_C

Theorem (SR in λ_C)

Theorem (SE in λ_C)

Translation correctness from λ_C to $s\pi^!$

Theorem (Translation Preserves Types)

1. *If* $\Theta; \Gamma \models B : (\sigma^k, \eta)$ *then* $[\![B]\!] \vdash [\![\Gamma]\!], u : [\![(\sigma^k, \eta)]\!]_{(\sigma,i)}, [\![\Theta]\!]$.
2. *If* $\Theta; \Gamma \models M : \tau$, *then* $[\![M]\!]_u \vdash [\![\Gamma]\!], u : [\![\tau]\!], [\![\Theta]\!]$.

Theorem (Translation correctness under $\longrightarrow$)

The translation $[\![\cdot]\!]_{\_} : (\Lambda, \longrightarrow) \to (\Pi, \longrightarrow)$ *is correct using equivalence* $\succeq_{\Downarrow}$.

Discussion

- Under $\longrightarrow$, non-deterministic choice in $\mathsf{s}\pi^!$ is eager.
- Recall the $\lambda_{\mathtt{C}}$ example. In M, variables x_1, x_2, x_3 are substituted non-deterministically in three steps: first one of three substitutions is chosen for x_1, then one of two remaining substitutions is chosen for x_2, then one substitution remains for x_3.
- In $[\![M]\!]$, $\longrightarrow$ chooses eagerly: all three choices are made in one step.
- Hence, correctness of translation holds up to $\succeq_{\Vert}$.

Discussion

APLAS'23: s$\pi^!$ with lazy semantics that postpones non-deterministic choice as long as possible; translation is correct up to $\equiv$.

Comparison:

- Eager semantics: close to traditional non-determinism in π, straightforward definition, usual notions of bisimulation ("$\alpha; P + \alpha; Q \not\simeq \alpha; (P + Q)$").
- Lazy semantics: more fine-grained non-determinism, complex definition, unusual notions of bisimulation ("$\alpha; P + \alpha; Q \simeq \alpha; (P + Q)$").

Closing Remarks

We studied the interplay between resource control and non-determinism in typed calculi.

- We introduced two calculi with non-confluent non-determinism, both equipped with type systems for resource control.
- Inspired by the untyped calculus, non-determinism in $s\pi^!$ is gradual and explicit, with session types.
- In λ_C, non-determinism arises in the fetching of resources, and is regulated by intersection types.
- A correct translation of λ_C into $s\pi^!$ precisely connects their different forms of non-determinism.
- This work reinforces our discovered connection between intersection and session types.

Typed Non-determinism in Concurrent Calculi: The Eager Way

Daniele Nantes
Imperial College London (UK)

Joint work with
Joseph Paulus (University of Oxford, UK)
Bas van den Heuvel (Karlsruhe Univ. of Applied Sciences, GER)
Jorge A. Pérez (Univ. of Groningen, NL)

Mathematical Foundations in Program Semantics (MFPS)