Branching Input Strongest Typing Judgment

$P = a?\{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$

- Let $\emptyset; \Gamma^\star_{\mathtt{l}} \cup \{x_j\}_{\mathtt{l}}; \emptyset; \Gamma_{(j)_{\mathtt{u}}} \vdash_{\mathrm{R}} P_j$ be the strongest judgment for P_j. (By induction, $\exists! \Gamma^\star_{\mathtt{l}}$)
- Let $\Delta_{\mathtt{l}}; \Gamma_{\mathtt{l}}; \Delta_{\mathtt{u}}; \Gamma_{\mathtt{u}} \vdash_{\mathrm{R}} P$ be any judgment for process P. (R-INP) $\Rightarrow \Delta_{\mathtt{l}} = \{a\}_{\mathtt{l}}$, $\Gamma_{\mathtt{l}} = \Gamma^\star_{\mathtt{l}}$, $\Delta_{\mathtt{u}} = \emptyset$ and $\forall j \in J, \emptyset; \Gamma^\star_{\mathtt{l}} \cup \{x_j\}_{\mathtt{l}}; \emptyset; \Gamma_{\mathtt{u}} \cup \{x_j\}_{\mathtt{u}} \vdash_{\mathrm{R}} P_j$.
- By induction hypothesis : $\forall j \in J, \Gamma_{(j)_{\mathtt{u}}} \subseteq \Gamma_{\mathtt{u}} \cup \{x_j\}_{\mathtt{u}}$.
- Therefore : $\bigcup_{j\in J} \left(\Gamma_{(j)_{\mathtt{u}}} \setminus \{x_j\}_{\mathtt{u}}\right) \subseteq \Gamma_{\mathtt{u}}$.
- The strongest judgment for P is then : $\{a\}_{\mathtt{l}}; \Gamma^\star_{\mathtt{l}}; \emptyset; \bigcup_{j\in J} \left(\Gamma_{(j)_{\mathtt{u}}} \setminus \{x_j\}_{\mathtt{u}}\right) \vdash_{\mathrm{R}} a?\{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$

What Is TyCO, After All ?

Maxime Gamboni

EPFL

Asynchronous π-Calculus

Basic Components

- *Names* : a, b, c, x . . .
- *Processes* : P, Q, . . .

Processes use names as *channels* for sending or receiving data

- Sending x on a : $a!x$
- Receiving x on a (and then processing it in P) : $a?(x).P$

Asynchronous π-Calculus (continued)

Processes can be combined using *parallel composition*

- $P|Q$

Example :

- $(a!x)|(a?(y).y!b) \rightarrow \mathbf{0}|x!b$

Names can be *restricted* to a part of a process

- $((\boldsymbol{\nu} x)\, P)|Q$: x is visible in P but not (directly) in Q
- $((\boldsymbol{\nu} x)\, a!x)|(a?(y).y!b) \rightarrow (\boldsymbol{\nu} x)\,(\mathbf{0}|x!b)$: P sends x on a name visible to Q, which *extrudes* the scope of x.

π_{a}^{V}: *Asynchronous* π*-Calculus with Variants*

Names sent on a channel can be *labeled* :

- in $a!v$,
- $$\begin{array}{llll} v & ::= & a & \text{name} \\ & \mid & \mathrm{l}\langle v\rangle & \text{labeled value} \end{array}$$
- Labels are *not* names, they are just labels.

A *case destructor* can be used when receiving a labeled value :

- case v of $\{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$

What is TyCO?

With slight syntactic changes, TyCO is just a sub-calculus of π_{a}^{V}:

- Any output must be with a name having a single label (no label-nesting)
- At input time the case destruction is done immediately.
- Additionally, instead of writing $a?(v).\mathsf{case}\ v\ \mathsf{of}\ \{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$ like we would in π_{a}^{V}, we write $a?\{\mathrm{l}_j(x_j){=}P_j\}_{j\in J}$, which illustrates the atomicity of input and case-destruction.

Example : Church-Encoding of Natural Numbers

- $\boldsymbol{Zero}(x) \stackrel{\text{def}}{=} x?^*\{\mathrm{q}(a){=}a!\mathrm{z}\langle x\rangle\}$ (let x be the number *zero*)
- $\boldsymbol{Succ}(y,x) \stackrel{\text{def}}{=} x?^*\{\mathrm{q}(a){=}a!\mathrm{s}\langle y\rangle\}$ (let x be the *successor* of y)
- $\boldsymbol{Add}(x,y,z) \stackrel{\text{def}}{=}$

$$
\begin{aligned}
& x!\mathrm{q}(\boldsymbol{\nu} a).a?\{\mathrm{z}(b){=}z!\mathrm{a}\langle y\rangle, \\
& \qquad \mathrm{s}(b){=}(\boldsymbol{\nu} t)\ \boldsymbol{Add}(b,y,t) \mid \\
& \qquad\qquad t?\{\mathrm{a}(n){=}z!\mathrm{a}(\boldsymbol{\nu} r).\boldsymbol{Succ}(n,r)\} \\
& \qquad \}
\end{aligned}
$$

Is π_{a}^{V} More Expressive than TyCO?

- TyCO being a sub-calculus of π_{a}^{V}, an encoding of TyCO into π_{a}^{V} is straightforward.
- Is it possible however to make an encoding of π_{a}^{V} into TyCO?
- We need to encode nested variants as single-level variants and break the input / variant-destruction atomicity
- The encoding needs to respect the process equivalences, i.e. $P \mathcal{R} Q \Leftrightarrow [\![P]\!] \mathcal{R} [\![Q]\!]$

Description of my project

- To write a description of TyCO-π_{a}^{V} encoding and prove it is valid
- My guide @ EPFL provided me with a π_{a}^{V}-TyCO encoding
- The goal of my project is to prove that it is valid and maybe to make necessary changes to it
- If I have enough time, then study whether *Non-Uniform TyCO* can be encoded into π_{a}^{V}

That's All, Folks

. . .