What Is TyCO, After All ?
Final Seminar

Maxime Gamboni

EPFL

Plan

1. What I had to do
2. What I did
3. What is yet to be done

π_{a}^{V} ***vs.* TyCO**

- Asynchronous π-calculus with Nested Variants

What is TyCO?

- π_{a}^{V} with one-level variants only
- message input and destruction is atomic

Does π_{a}^{V} have more expressive power than TyCO?

Encodings, requirements

We want a good and fully abstract encoding from $\pi_{\mathbf{a}}^{V}$ to TyCO and the other way round.

- Fully Abstract Encoding

$P \sim Q$ if and only if $[\![\, P \,]\!] \sim [\![\, Q \,]\!]$

- Distributed Encoding

$[\![\, P|Q \,]\!] = [\![\, P \,]\!] | [\![\, Q \,]\!]$ and $[\![\, (\boldsymbol{\nu} a)\, P \,]\!] = (\boldsymbol{\nu} a)\, [\![\, P \,]\!]$

What I (we) Changed

From the original document, I did the following changes :

- Case Reduction Relation (doesn’t take a step)
- Linear Receptiveness
- Undecidability of D-Link
- Definition of Receptive Equivalences
- Made the Nested Encoding Syntax-Directed
- Minor Fixes (Substitution, Operational Correspondence, Full Abstraction . . .)

Case Reduction Semantics

We tried several semantics for handling of π_{a}^{V}'s case reduction :

1. Structural Congruence $\equiv$
   (Breaks Subject Congruence)
2. τ-transition $\rightarrow$
   (Full Abstraction on weak equivalences only)
3. Directional Congruence $\rightarrowtail$
   (works :-))

Linear Weakening (Receptiveness)

- The problem:

For a linear, the typability of $(\boldsymbol{\nu} a)\,P$ requires a to be read and written in P. But :

$$(\boldsymbol{\nu} a)\,(a!\mathrm{l_k}(\boldsymbol{\nu} x).Q \mid a?\{\mathrm{l}_j(y_j){=}P_j \;\mid\; j{\in}J\}) \xrightarrow{\tau}$$
$$(\boldsymbol{\nu} a)\,(\boldsymbol{\nu} x)\,(Q \mid P_k\{^x/_{y_k}\})$$

In that example type soundness is broken !

- The answer:

Linear Weakening

Undecidability of D-Link

- We had introduced the concept of *Dynamic Links* to avoid extrusion of plain names.
- Its definition is recursive using input and bound output: $a \gg b \overset{\mathrm{def}}{=} a?^*\{\mathrm{l}_j(x) = b!\mathrm{l}_{\mathrm{j}}(\boldsymbol{\nu} z).z \gg x \mid j \in J\}$ (uniform case)
- I spent a few weeks to prove its (receptive) typability before seeing that it is undecidable (so I made it an axiom)

Minor Changes

- Dynamic links have to work on branching inputs as well
- The first version of the $\pi_{\mathrm{a}}^{V} \rightarrow$ TyCO encoding was type-directed but it could be made syntax directed only.
- π_{a}^{V}-TyCO Full Abstraction could be simplified

So, does it work, finally ?

Short Answer : No.
Long Answer :
It works only on a subset of π_{a}^{V} processes.

1. The encoding doesn't work on processes that receive on received names because it breaks uniformity.
   $d?(x).\,x?(y)$
2. The operational correspondence is broken on processes that do input and free output on a name
   $a!x \mid a?(y).P$

Conclusion

There is still some work to be done in the area !

- Are TyCO and π_{a}^{V} equivalent ?

“Probably” ...

Thank You

- Thank You For Following Me (or attempting to) !
- Questions ?