What Is TyCO, After All ?
Final Seminar

Maxime Gamboni

EPFL

Asynchronous π-Calculus with Nested Variants for Dummies

Values.

- *Names* : a, b, x $\ldots$ *Labels* : l, a, $\ldots$
- *Values* : A name or a labeled value (a, $\mathrm{l}\langle a \rangle$, $\mathrm{l}_1\langle \mathrm{l}_2\langle x \rangle\rangle$)

Processes.

- Sending $\mathrm{l}\langle b \rangle$ on x: $x!\mathrm{l}\langle b \rangle$
- Receiving x on a (and then processing it in P) : $a?(x).P$
- Parallel composition : $P_1 \mid P_2$
- Restriction : $(\boldsymbol{\nu} a)\,(P)$

Asynchronous π-Calculus with Nested Variants (continued)

- Analysing a value : $\mathsf{case}\ v\ \mathsf{of}\ \{\mathrm{l}_j(x_j){=}P_j \mid j{\in}J\}$

Example :

$$((\boldsymbol{\nu} x)\, b!\mathrm{l}_3\langle x\rangle) \mid b?(y).\mathsf{case}\ y\ \mathsf{of}\ \{\mathrm{l}_j(z_j){=}P_j \mid j{\in}J\}$$

$$\rightarrow$$

$$(\boldsymbol{\nu} x)\,(\mathbf{0} \mid \mathsf{case}\ \mathrm{l}_3\langle x\rangle\ \mathsf{of}\ \{\mathrm{l}_j(z_j){=}P_j \mid j{\in}J\}$$

$$\rightarrowtail$$

$$(\boldsymbol{\nu} x)\,(\mathbf{0} \mid P_3\{{}^{x}/_{z_3}\})$$

π_{a}^{V} vs TyCO

TyCO can be seen as a simplified version of π_{a}^{V}.

- All values ("*messages*") are one name ("*argument*") with a single label ("*method name*")
- *Receiving* a value and *Analysing* it are done *atomically*

TyCO Receptiveness Theory

Receptive names are a special category of names for which any valid process must ensure the following properties :

1. A message can be sent to a name only if there is a receiver
2. There can be no more than one receiver on a name

Receptive names will be created by the encoding to provide “control” names with which we don’t want the user to mess up.

Our Goal

Are TyCO and π_{a}^{V} equivalent ?
We need *encodings* between TyCO and π_{a}^{V} that are :
Good :

- Preserve process interface and structure
- Divergence- and deadlock-free

Fully Abstract :

- The encodings of two equivalent processes yield two equivalent processes
- The encodings of two non-equivalent processes yield two non-equivalent processes

What I had

A draft of a paper with :

- Full description of π_{a}^{V} and TyCO, syntax, semantics, receptiveness, equivalences etc
- Almost working version of encodings in both directions.

What I was supposed to do

Proofs !

- Fix what needs to be fixed in the theory and in the encoding
- Once everything works, write proofs for it

π_{a}^{V} *to TyCO encoding briefly*

We need to encode separate case reduction in TyCO, as well as transmitting values that have no or more than one label.
This is done with an encoding of the concept of values :

$$[\![\, b \,]\!]_u \stackrel{\text{def}}{=} u \gg b$$

$$[\![\, \mathrm{l}\langle v \rangle \,]\!]_u \stackrel{\text{def}}{=} u?^*\{\mathrm{d}\,(r){=}(\boldsymbol{\nu} u')\,(r!\mathrm{l}\langle u' \rangle \mid [\![\, v \,]\!]_{u'})\}$$

- To encode sending data we send this “accessor” (u) instead, with label c
- Case analysis is done sending a d-labeled message to an encoding of a value

What I did

★ TyCO being a sub-calculus of π_{a}^{V}, an encoding of TyCO into π_{a}^{V} is straightforward.

- It was not :-)

This is what I did, with the help of my guides :

- We had to fix the handling of case reduction
- Weaken the receptiveness type system
- Write the definitions of TyCO-equivalences
- Simplify the π_{a}^{V}-TyCO encoding
- Prove almost everything

So, does it work?

- One day before my deadline I found a counter-example!
- But there is serious hope we get a working encoding soon.
- Which means there is serious hope that having nested variants and separating input and case analysis does *not* bring additional expressive power.

What's next ?

For studying this at home,
You can find our report, the previous presentation as well as this one one at `http://gamboni.org/tyco`

- Thank you for following me and
- Enjoy the cookies :-)