

Separation of synchronous and asynchronous communication via testing

Diletta Cacciagrano

Flavio Corradini

Catuscia Palamidessi

Plan of the talk

- Classical encodings of the output prefix of π
- Must semantics
- Impossibility of a must-preserving encoding
- Discussion

Classical encodings of the output prefix of π

- At the beginning of the 90's various researchers (Boudol and Honda-Tokoro) proposed independently elegant encodings of the output-prefix of the π -calculus into the asynchronous π -calculus
 - The idea was not new, based on rendezvous protocol, but the presence of the new operator allowed for a solution particularly elegant and modular (which would not be possible, for instance, in CCS)
 - This encodings contributed to show the expressiveness of the asynchronous π -calculus and helped significantly its popularity
- Despite the importance of this result, the formal properties of the encodings were not explored much. The aim of this paper is to show that these encodings (and in fact any encoding from π into asynchronous π) does not preserve certain semantics properties
- This result has nothing to do with [Palamidessi 97]: here we consider a language without choice

Classical encodings of the output prefix of π

- The source language: The synchronous π -calculus w/o choice

$$P ::= 0 \mid x(y).P \mid \tau.P \mid \bar{x}y.P \mid P \mid P \mid (\nu x)P \mid !P$$


- The target language: The asynchronous π -calculus

$$P ::= 0 \mid x(y).P \mid \tau.P \mid \bar{x}y \mid P \mid P \mid (\nu x)P \mid !P$$


Classical encodings of the output prefix of π

- The encoding of Boudol [1992]

- $\llbracket \bar{x}y.P \rrbracket = (\nu z)(\bar{x}z \mid (z(w).\bar{w}y \mid \llbracket P \rrbracket))$
- $\llbracket x(y).Q \rrbracket = x(z).(\nu w)(\bar{z}w \mid w(y).\llbracket Q \rrbracket)$

$\llbracket \cdot \rrbracket$ is homomorphic for all the other operators. Namely:

- $\llbracket 0 \rrbracket = 0$
- $\llbracket P \mid Q \rrbracket = \llbracket P \rrbracket \mid \llbracket Q \rrbracket$
- $\llbracket (\nu x)P \rrbracket = (\nu x)\llbracket P \rrbracket$
- $\llbracket ! P \rrbracket = ! \llbracket P \rrbracket$

Classical encodings of the output prefix of π

- The encoding of Honda-Tokoro [1991]

- $\llbracket \bar{x}y.P \rrbracket = x(z).(\bar{z}y \mid \llbracket P \rrbracket)$
- $\llbracket x(y).Q \rrbracket = (\nu z)(\bar{x}z \mid z(y).\llbracket Q \rrbracket)$

$\llbracket \cdot \rrbracket$ is homomorphic for all the other operators. Namely:


- $\llbracket 0 \rrbracket = 0$
- $\llbracket P \mid Q \rrbracket = \llbracket P \rrbracket \mid \llbracket Q \rrbracket$
- $\llbracket (\nu x)P \rrbracket = (\nu x)\llbracket P \rrbracket$
- $\llbracket ! P \rrbracket = ! \llbracket P \rrbracket$

Classical encodings of the output prefix of π

- Both these encodings have been proved correct with respect certain weak semantics (Morris preorder for the Boudol's encoding, and a sort of weak bisimulation for the Honda-Tokoro encoding)
- They are also may-preserving, in the sense that
$$P \text{ may } o \text{ iff } [[P]] \text{ may } [[o]] \quad [\text{Cacciagrano \& Corradini, 2001}]$$
- Both these encodings have the feature that one of the encoded partners reaches before the other the continuation
- We will see that this fact causes problems with respect to preservation of the must semantics

The must semantics

- Given a process P and a test o containing the special action ω , we say that P **must** o if and only if for every maximal computation of $P \mid o$ we have that we reach a state where ω is enabled

$$P \mid o = \langle P_0 \mid o_0 \rangle \xrightarrow{\tau} \langle P_1 \mid o_1 \rangle \xrightarrow{\tau} \langle P_2 \mid o_2 \rangle \xrightarrow{\tau} \dots$$


The main result

- Given an encoding $\llbracket \cdot \rrbracket : \pi \rightarrow \text{asynchronous } \pi$ such that
 - $\llbracket \cdot \rrbracket$ is compositional with respect to the prefixes
 - There exists P such that $\llbracket P \rrbracket \uparrow$ (i.e. $\llbracket P \rrbracket$ has a computation consisting of an infinite sequence of τ 's)
- Then $\llbracket \cdot \rrbracket$ cannot be must-preserving, namely there exists a process P and a test O such that

$$P \text{ must } o \quad \text{while} \quad \llbracket P \rrbracket \not\text{must } \llbracket o \rrbracket$$

Proof of the main result (outline)

Let P be a process such that $\llbracket P \rrbracket \uparrow$. Then consider

- 1) Process $x(y).P$ with test $\bar{x}z.\omega$
- 2) Process $\bar{x}z.P$ with test $x(y).\omega$

Note that in both cases we have

By contr: If $\llbracket \cdot \rrbracket$ were must-preserving we should have

1) $x(y).P \text{ must } \bar{x}z.\omega$

1) $\llbracket x(y).P \rrbracket \text{ must } \llbracket \bar{x}z.\omega \rrbracket$

2) $\bar{x}z.P \text{ must } x(y).\omega$

2) $\llbracket \bar{x}z.P \rrbracket \text{ must } \llbracket x(y).\omega \rrbracket$

Proof of the main result (outline)

Because of compositionality we have $\llbracket x(y).P \rrbracket = \mathcal{C}_{x(y)}[\llbracket P \rrbracket]$
and analogously for the other prefixes. Hence

$$1) \llbracket x(y).P \rrbracket \mid \llbracket \bar{x}z.\omega \rrbracket = \mathcal{C}_{x(y)}[\llbracket P \rrbracket] \mid \mathcal{C}_{\bar{x}z}[\llbracket \omega \rrbracket] \rightarrow \dots \rightarrow \llbracket P \rrbracket \mid Q \mid a.\llbracket \omega \rrbracket \rightarrow^*$$

or

$$1) \llbracket x(y).P \rrbracket \mid \llbracket \bar{x}z.\omega \rrbracket = \mathcal{C}_{x(y)}[\llbracket P \rrbracket] \mid \mathcal{C}_{\bar{x}z}[\llbracket \omega \rrbracket] \rightarrow \dots \rightarrow b.\llbracket P \rrbracket \mid Q \mid \llbracket \omega \rrbracket \dots \xrightarrow{3}$$

In the second case, we just have to consider the other pair:

$$2) \llbracket \bar{x}z.P \rrbracket \mid \llbracket x(y).\omega \rrbracket = \mathcal{C}_{\bar{x}z}[\llbracket P \rrbracket] \mid \mathcal{C}_{x(y)}[\llbracket \omega \rrbracket] \rightarrow \dots \rightarrow \llbracket P \rrbracket \mid Q \mid b.\llbracket \omega \rrbracket \rightarrow^*$$

Discussion

- What does the result really mean?
 - The π -calculus (even with no choice) is more expressive than the asynchronous π -calculus w.r.t. testing semantics
 - However the result heavily depends on some features of the testing semantics that may be considered not too essential
 - The result does not hold in the following variants of the testing semantics
 1. Success is declared only if ω is actually performed
 2. Success is declared if at some point ω is available or there are infinitely many points from which one can reach a point where ω is available (fair testing, [Natarajan and Cleaveland, 1995] and [Brinksma, Rensink and Vogler, 1995])

Future work

- Study a notion of testing semantics with a limited form of fairness (essentially, a fairness which guarantees only the delivery of the outputs)
 - fair testing is “too coarse” for our purposes: we do not want to identify processes like P and $P \mid !\tau$
- Prove positive results (encodings preserving the testing semantics) w.r.t. this limited fairness variant

Thank you !