Symmetries and dualities in the π-calculus

Davide Sangiorgi

INRIA Sophia Antipolis
2004 route des Lucioles, BP 93
06902 Sophia Antipolis CEDEX

Email: davide.sangiorgi@sophia.inria.fr
http://www.inria.fr/meije/personnel/Davide.Sangiorgi.html

July 1997 (revised November 1997)

OUTLINE

- The π-calculus 1
- Some complications of the theory of the π-calculus 2
- Towards symmetry and duality: internal mobility 12
- Internal mobility: expressiveness and programming 20
- Internal mobility for reasoning 27
- Future work 31

CONTENTS

☞ • The π-calculus [1]
  – The π-calculus
• Some complications of the theory of the π-calculus [2]
• Towards symmetry and duality: internal mobility [12]
• Internal mobility: expressiveness and programming [20]
• Internal mobility for reasoning [27]
• Future work [31]

$\boxed{\text{The } \pi\text{-calculus}}$

$a, b, \ldots, x, y, z, \ldots$ *Names*

Processes

$$
\begin{array}{lcll}
P & ::= & 0 & \text{nil process} \\
 & \mid & P \mid P & \text{parallel} \\
 & \mid & \boldsymbol{\nu} a\ P & \text{restriction} \\
 & \mid & a(b).\,P & \text{input} \\
 & \mid & \overline{a}\langle b\rangle.\,P & \text{output} \\
 & \mid & A\langle \widetilde{a}\rangle & \text{constants (with defs. } A(\widetilde{a}) \stackrel{\text{def}}{=} P) \\
 & \mid & P + P & \text{summation}
\end{array}
$$

- Abbreviation: $\overline{a}(b).\,P \stackrel{\text{def}}{=} \boldsymbol{\nu} b\ \overline{a}\langle b\rangle.\,P$
- Transitions: $P \xrightarrow{\tau} P'$ $\quad P \xrightarrow{a(b)} P'$
  $P \xrightarrow{\overline{a}\langle b\rangle} P'$ $\quad P \xrightarrow{\overline{a}(b)} P'$
- Basic communication rules:

$$
\begin{array}{rcll}
a(y).\,Q \mid \overline{a}\langle b\rangle.\,P & \xrightarrow{\tau} & Q\{^b\!/\!_y\} \mid P & \\
a(y).\,Q \mid \boldsymbol{\nu} b\ (\overline{a}\langle b\rangle.\,P) & \xrightarrow{\tau} & \boldsymbol{\nu} b\ (Q\{^b\!/\!_y\} \mid P) & b \text{ fresh.}
\end{array}
$$

Communications involve arbitrary substitutions among names.

CONTENTS

✓ • The π-calculus [1]

☞ • Some complications of the theory of the π-calculus [2]
  - Behavioural equivalences
  - Algebraic theory
  - Canonical normal forms
  - Types
  - Observables
  - Interaction

• Towards symmetry and duality: internal mobility [12]

• Internal mobility: expressiveness and programming [20]

• Internal mobility for reasoning [27]

• Future work [31]

Behavioural equivalences

Bisimulation in CCS

$P \sim Q$ implies

– if $P \xrightarrow{\mu} P'$ then there exists Q' s.t. $Q \xrightarrow{\mu} Q'$ and $P' \sim Q'$.

What is bisimulation in the π-calculus?

$P \sim Q$ implies

1. If $P \xrightarrow{a(x)} P'$ then

$$
?? \left\{ \begin{array}{lll}
- & \exists \; Q' \text{ s.t. } Q \xrightarrow{a(x)} Q' \text{ and } P' \sim Q'; & (\textit{ground}) \\
- & \exists \; Q' \text{ s.t. } Q \xrightarrow{a(x)} Q' \text{ and } \forall \; b,\; P'\{b/x\} \sim Q'\{b/x\}; & (\textit{late}) \\
- & \forall \; b,\, \exists \; Q' \text{ s.t. } Q \xrightarrow{a(x)} Q' \text{ and } P'\{b/x\} \sim Q'\{b/x\}. & (\textit{early})
\end{array} \right.
$$

2. if $P \xrightarrow{\mu} P'$ and μ is not an input, then $\exists \; Q'$ s.t. $Q \xrightarrow{\mu} Q'$ and $P' \sim Q'$.

(by alpha conversion we assume that the bound names of actions are always fresh)

- Neither ground, nor late, nor early bisimulation is a congruence relation:

$$\begin{array}{rcl} x \mid \overline{y} & \sim_{\texttt{Late/Early}} & x.\,\overline{y} + \overline{y}.\,x \\ a(x).\,(x \mid \overline{y}) & \not\sim_{\texttt{Late/Early}} & a(x).\,(x.\,\overline{y} + \overline{y}.\,x) \end{array}$$

- Therefore one has to introduce separately the induced congruence relations.

- *Open bisimulation*: The clauses of ground bisimulation plus: $P \sim Q$ implies $P\sigma \sim Q\sigma$, for all σ.

$$\begin{tikzcd}
 & & \sim_{\texttt{Ground}} & \\
 & & \sim_{\texttt{Early}} \arrow[u] & \\
\sim^{\texttt{c}}_{\texttt{Early}} = \sim^{\texttt{c}}_{\texttt{Ground}} \arrow[urr] & & & \\
 & & & \sim_{\texttt{Late}} \arrow[uul] \\
 & \sim^{\texttt{c}}_{\texttt{Late}} \arrow[uul] \arrow[urr] & & \\
 & \sim_{\texttt{Open}} = \sim^{\texttt{c}}_{\texttt{Open}} \arrow[u] & &
\end{tikzcd}$$

$\sim^{\texttt{c}}_{x} \equiv$ congruence induced by $\sim_x$

Algebraic theory

- Expansion law: *matching* is needed

$$
\begin{array}{rclcl}
\overline{x}\langle v\rangle . P \mid z(x) . Q & = & & \overline{x}\langle v\rangle . (P \mid z(x) . Q) & + \\
& & & z(x) . (Q \mid \overline{x}\langle v\rangle . P) & + \\
& & [x = z] & \tau . (P \mid Q\{v/x\}) &
\end{array}
$$

- For axiomatising the bisimulations, we also need *mismatching*. Axioms like

$$P \quad = \quad [x = y]P + [x \neq y]P$$

are used in the completeness proofs to make a case analysis on the possible equalities and inequalities among names.

Canonical normal forms

- With the exception of open bisimulation, for none of the bisimulations and their congruences we know what are the canonical normal forms

$\boxed{\text{Types}}$

- $a : T^\wedge \equiv$ "a carries values of type T."

- The untyped π-calculus can be seen as a typed calculus where all names have the type $\mu X.\,(X^\wedge)$

- The grammar of types can be extended to get richer languages:

tupling	(cf.: the polyadic π-calculus)
records	
capabilities	(also called I/O types)
polymorphism	
etc.	

- In the untyped calculus substitution gives problem. In typed calculi, the problems become more serious...
  Moreover, the notion of *observable* of the untyped calculus can be (very) unsatisfactory.

Observables

The ordinary bisimulations have the clauses

– if $P \xrightarrow{\overline{a}\langle x\rangle} P'$ then there exists Q' s.t. $Q \xrightarrow{\overline{a}\langle x\rangle} Q'$ and $P' \sim Q'$.
– if $P \xrightarrow{\overline{a}(x)} P'$ then there exists Q' s.t. $Q \xrightarrow{\overline{a}(x)} Q'$ and $P' \sim Q'$.

Is "privateness" observable?

Suppose only the output capability of names can be passed (this is VERY frequent; cf.: Pict or Join calculus).

Take

$$
\begin{array}{rcl}
P & \stackrel{\mathsf{def}}{=} & \boldsymbol{\nu} x\ (!x(y).\, R \mid \overline{a}\langle x\rangle \mid \overline{b}\langle x\rangle) \\
Q & \stackrel{\mathsf{def}}{=} & \boldsymbol{\nu} x\ (!x(y).\, R \mid \overline{a}\langle x\rangle \mid \overline{b}(z).\, z \rhd x)
\end{array}
$$

$$
\begin{array}{lrcl}
\text{where} & z \rhd x & \stackrel{\mathsf{def}}{=} & !z(u).\, \overline{x}\langle u\rangle \\
 & !P & \stackrel{\mathsf{def}}{=} & P \mid !P
\end{array}
$$

No well-typed context can distinguish between P and Q. How can we prove it?

NB: P and Q have these transitions:

$$
\begin{array}{l}
P \quad \xrightarrow{\overline{a}(x)} \xrightarrow{\overline{b}\langle x\rangle} \\
Q \quad \xrightarrow{\overline{a}(x)} \xrightarrow{\overline{b}(z)}
\end{array}
$$

Names that are passed with only the input or only the output capability should not be tested for equality by the recipient.

Then: **What is observable?**

- The previous formulations of bisimulation are too strong
- The algebraic theory of the untyped π-calculus is far too poor

$\boxed{\text{Interaction}}$

$$a(y).\,Q \mid \overline{a}\langle b\rangle.\,P \xrightarrow{\tau} Q\{{}^{b}\!/\!{}_{y}\} \mid P$$

The use of substitutions makes interaction a complex operation, as it shows up

– in implementations,
– in the theory.

Can interaction be simpler?

CONTENTS

✓ • The π-calculus [1]

✓ • Some complications of the theory of the π-calculus [2]

☞ • Towards symmetry and duality: internal mobility [12]
  - The calculus πI
  - Symmetries and dualities
  - Operational theory of πI
  - Bisimulation for πI
  - An axiomatisation of πI

• Internal mobility: expressiveness and programming [20]

• Internal mobility for reasoning [27]

• Future work [31]

The π-calculus has 2 communication rules:

1. $a(b).\, Q \mid \overline{a}(b).\, P \xrightarrow{\tau} \boldsymbol{\nu} b\, (Q \mid P)$ $\qquad$ b fresh.

   A private name is being communicated.

   For this, we only need α-conversion [call this *internal mobility*]

2. $a(y).\, Q \mid \overline{a}\langle b\rangle.\, P \xrightarrow{\tau} Q\{^{b}\!/\!_{y}\} \mid P$

   A global name is being communicated.

   For this, we need full substitutions [call this *external mobility*]

Do we need them both in a model of mobile processes?

$\boxed{\text{The calculus } \pi\text{I}}$

$a, b, \ldots, x, y, z, \ldots$ *Names*

			Processes
P	$::=$	0	nil process
	$\mid$	$P \mid P$	parallel
	$\mid$	$\boldsymbol{\nu} a\ P$	restriction
	$\mid$	$a(b).\,P$	input
	$\mid$	$\overline{a}(b).\,P$	output
	$\mid$	$A\langle\widetilde{a}\rangle$	recursion
	$\mid$	$P + P$	summation

- Basic communication rule:

$$a(x).\,Q \mid \overline{a}(x).\,P \xrightarrow{\tau} \boldsymbol{\nu} x\ (Q \mid P)$$

- In πI there *internal*, but not external, mobility.

Symmetries and dualities

– Every process $P \in \pi\mathsf{I}$ has a dual $\overline{P}$ s.t.

$$\begin{array}{ccc} P & \leftrightarrow & \overline{P} \\ \mu \downarrow & & \overline{\mu} \downarrow \\ P' & \leftrightarrow & \overline{P'} \end{array}$$

– The duality input/output can be strengthened, by requiring that

1. in an *input* $x(y).\,P$, the object y may only be used in *output* in P;
2. in an *output* $\overline{x}(y).\,P$, the object y may only be used in *input* in P.

Call this calculus $\pi\mathsf{I}^{-}$.

Example:

$$a(b).\,\overline{b}(c).\,c(d).\,0 \mid \overline{a}(b).\,b(c).\,\overline{c}(d).\,0 \in \pi\mathsf{I}^{-}$$

– A *trace* of a process P_1 is a finite or infinite sequence of actions

$$\mu_1, \ldots, \mu_n$$

s.t. there are processes P_i with $P_i \xrightarrow{\mu_i} P_{i+1}$, $1 \leq i < n$.

– From the trace of a process, we can extract *dependency chains*, that is sequences

$$\mu_1, \ldots, \mu_m$$

where for all $1 \leq j < m$, $\texttt{object}(\mu_{j+1}) = \texttt{subject}(\mu_j)$.

– Example: Some dependency chains of the trace $x(y_1), x(y_2), y_1(w), y_2(v), \overline{w}(z)$ are $x(y_1), y_1(w), \overline{w}(z)$ and $x(y_2), y_2(v)$.

– All dependency chains of a process in $\pi\mathsf{I}^-$ are of the form

$$x_1(x_2), \overline{x_2}(x_3), \overline{x_3}(x_4), \ldots,$$

or its dual.

- In π, or πI, we can write:

$$a(y).\,\overline{y}(z).\,C \mid \overline{a}(x).\,(x(c).\,A \mid \overline{x}(y).\,B)$$

  There can be interferences between $\overline{y}(z).\,C$ and $\overline{x}(v).\,B$.

- By contrast, in any πI$^-$ process $x(y).\,P \mid \overline{x}(y).\,Q$:

– all interferences on outputs at y are local to P;

– all interferences on inputs at x are local to Q.

cf.: Reynold's syntactic control of interferences

$\boxed{\text{Operational theory of } \pi\text{I}}$

Roughly, the operational theory of πI is that of CCS plus alpha conversion. This holds for:

– SOS semantics,
– bisimilarity,
– axiomatisation,
– canonical normal forms.

$\boxed{\text{Bisimulation for } \pi\text{I}}$

Only ground bisimulation makes sense:

$P \sim Q$ implies

– if $P \xrightarrow{\mu} P'$ then there exists Q' s.t. $Q \xrightarrow{\mu} Q'$ and $P' \sim Q'$.

Similarly for weak bisimulation $\approx$.

$\boxed{\text{An axiomatisation of } \pi\mathrm{I}}$

Inference rules for congruence, alpha-conversion			$\ldots$	
Summation		$P + 0$	$=$	P
		$P + Q$	$=$	$Q + P$
		$P + (Q + R)$	$=$	$(P + Q) + R$
Restriction		$\boldsymbol{\nu} x\ (P + Q)$	$=$	$\boldsymbol{\nu} x\ P + \boldsymbol{\nu} x\ Q$
	$\mathtt{subject}(\alpha) = x$	$\boldsymbol{\nu} x\ \alpha.\, P$	$=$	0
	$\mathtt{subject}(\alpha) \neq x$	$\boldsymbol{\nu} x\ \alpha.\, P$	$=$	$\alpha.\, \boldsymbol{\nu} x\ P$

Expansion Assume $P \equiv \sum_i \alpha_i.\, P_i$ and $Q \equiv \sum_j \beta_j.\, Q_j$. Then:

$$P \mid Q = \sum_i \alpha_i.\, (P_i \mid Q) + \sum_j \beta_j.\, (P \mid Q_j) + \sum_{\substack{\alpha_i = y(x)\ ,\, \beta_j = \overline{y}(x) \\ \text{or the dual}}} \tau.\, \boldsymbol{\nu} x\ R_{ij}$$

CONTENTS

✓ • The π-calculus [1]

✓ • Some complications of the theory of the π-calculus [2]

✓ • Towards symmetry and duality: internal mobility [12]

☞ • Internal mobility: expressiveness and programming [20]

  – Encoding (core) π-calculus into πI
  – Endoding πA into $\pi\mathrm{I}^-$
  – Programming in πI
  – The call-by-name λ-calculus
  – Encoding call-by-name λ-calculus

• Internal mobility for reasoning [27]

• Future work [31]

Encoding (core) π-calculus into πI

$$
\left.
\begin{array}{ll}
\pi & \\
\downarrow & \text{Honda and Tokoro [1991], Boudol [1992]} \\
\pi_a & \\
\downarrow & \text{Boreale [1996]}
\end{array}
\right\} \text{folklore?}
$$

$$
\begin{array}{ll}
\pi\mathsf{A} & \\
\downarrow & \text{Boreale [1996], Merro [1997]} \\
\pi\mathsf{I}^- &
\end{array}
$$

$\pi_a \stackrel{\text{def}}{=}$ asynchronous π-calculus (all outputs are of the form $\overline{a}\langle b \rangle$ — just messages)

$\pi\mathsf{A} \stackrel{\text{def}}{=}$ asynchronous π-calculus where only the output capability of names may be transmitted.

Endoding πA into πI$^-$

$$[\![\overline{a}\langle b\rangle]\!] \stackrel{\text{def}}{=} \overline{a}(c).\, c \succ b$$

$$\text{where} \quad c \succ b \stackrel{\text{def}}{=} !c(x).\,\overline{b}(y).\, y \succ x \qquad \text{(Link)}$$

Example:

$$\overline{a}\langle b\rangle \mid a(x).\,\overline{x}\langle v\rangle \quad \xrightarrow{\tau} \quad \overline{b}\langle v\rangle$$

$$\begin{array}{rcl}
[\![\overline{a}\langle b\rangle \mid a(x).\,\overline{x}\langle v\rangle]\!] & \stackrel{\text{def}}{=} & \\
\overline{a}(c).\, c \succ b \mid a(x).\,\overline{x}(w).\, w \succ v & \xrightarrow{\tau} & \\
\boldsymbol{\nu} c\,(c \succ b \mid \overline{c}(w).\, w \succ v) & \xrightarrow{\tau} & \\
(\boldsymbol{\nu} c, w\,)(\overline{b}(u).\, u \succ w \mid c \succ b \mid w \succ v) & \sim & \\
\overline{b}(u).\,(\boldsymbol{\nu} c, w\,)(u \succ w \mid w \succ v \mid c \succ b) & \approx & \\
\overline{b}(u).\,\boldsymbol{\nu} c\,(u \succ v \mid c \succ b) & \sim & \\
\overline{b}(u).\, u \succ v & = & [\![\overline{b}\langle v\rangle]\!]
\end{array}$$

A useful law: $\boldsymbol{\nu} b\,(a \succ b \mid b \succ c) \approx a \succ c$.

---

cf.: the encoding has strong similarities with the implementation of free outputs in the Pict compiler.

Theorem [Merro, 1997] Let $P, Q \in \pi$A.
P and Q are asynchronously barbed congruent in πA

iff

$[\![P]\!]$ and $[\![Q]\!]$ are asynchronously barbed congruent in the asynchronous πI$^-$.

NB: In CCS or πI barbed congruence coincides with weak bisimulation; we have similar characterisations for the asynchronous barbed congruence (on asynchronous calculi).

$\boxed{\text{Programming in } \pi\mathsf{I}}$

Programming in πI is similar to programming in π-calculus, provided that we use some garbage collection laws for links. As an example we look at the encoding of call-by-name λ-calculus

The call-by-name λ-calculus

$$\beta \frac{}{(\lambda x.\, M)N \longrightarrow M\{^{N}\!/\!_{x}\}} \qquad\qquad \mu \frac{M \longrightarrow M'}{MN \longrightarrow M'N}$$

Encoding call-by-name λ-calculus

Into the π-calculus

$$
\begin{array}{rcl}
[\![\lambda x.\, M]\!]_p & \stackrel{\text{def}}{=} & \boldsymbol{\nu} y\, \overline{p}\langle y\rangle.\, !y(x,q).\, [\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}\langle p\rangle \\
[\![M\,N]\!]_p & \stackrel{\text{def}}{=} & \boldsymbol{\nu} q\; (\;\; [\![M]\!]_q \mid \\
 & & \qquad q(v).\, \boldsymbol{\nu} x\; (\overline{v}\langle x, p\rangle.\, !x(r).\, [\![N]\!]_r))
\end{array}
$$

Into πI

$$
\begin{array}{rcl}
[\![\lambda x.\, M]\!]_p & \stackrel{\text{def}}{=} & \overline{p}(y).\, !y(x,q).\, [\![M]\!]_q \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}(q).\, q \succ p \\
[\![M\,N]\!]_p & \stackrel{\text{def}}{=} & \boldsymbol{\nu} q\; (\;\; [\![M]\!]_q \mid \\
 & & \qquad q(v).\, \overline{v}(x,q).\, (q \succ p \mid !x(r).\, [\![N]\!]_r))
\end{array}
$$

N.B.: This is also an encoding into (asynchronous) $\pi\mathrm{I}^-$.

The same theorems known for the encoding into the π-calculus hold for the encoding $[\![.]\!]$ into πI:

Theorem For all $M, N \in \Lambda$, if $\lambda\beta \vdash M = N$ then $[\![M]\!]_p \approx [\![N]\!]_p$.

Theorem For all $M, N \in \Lambda$, $[\![M]\!]_p \approx [\![N]\!]_p$ iff M and N have the same Lévy-Longo trees.

The proofs for πI are as for π, if we use these two properties of links:

Lemma

1. $\boldsymbol{\nu} b\, (a \succ b \mid b \succ c) \approx a \succ c$.
2. Suppose x is a free variable of M. Then $\boldsymbol{\nu} x\, (x \succ y \mid [\![M]\!]_p) \approx [\![M\{y/x\}]\!]_p$.

CONTENTS

✓ • The π-calculus [1]
✓ • Some complications of the theory of the π-calculus [2]
✓ • Towards symmetry and duality: internal mobility [12]
✓ • Internal mobility: expressiveness and programming [20]
☞ • Internal mobility for reasoning [27]
• Future work [31]

Using links, we can adapt the encoding of CBN λ-calculus to an encoding of *strong CBN* λ-calculus into πI. "Strong" means that rule ξ is allowed:

$$\xi\,\frac{M \longrightarrow M'}{\lambda x.\, M \longrightarrow \lambda x.\, M'}$$

$$\begin{array}{rcl}
[\![\lambda x.\, M]\!]_p & \stackrel{\text{def}}{=} & (\boldsymbol{\nu} q, x\,)(\overline{p}(w).\, w(y,r).\,(q \succ r \mid x \succ y) \mid [\![M]\!]_q) \\
[\![x]\!]_p & \stackrel{\text{def}}{=} & \overline{x}(q).\, q \succ p \\
[\![M N]\!]_p & \stackrel{\text{def}}{=} & \boldsymbol{\nu} q\;(\;\; [\![M]\!]_q \mid \\
 & & \qquad q(v).\, \overline{v}(x,q).\,(q \succ p \mid !x(r).\, [\![N]\!]_r))
\end{array}$$

Theorem For all $M, N \in \Lambda$, $[\![M]\!]_p =_{\texttt{must}} [\![N]\!]_p$ iff M and N have the same Böhm trees.

Having an encoding into πI we can simplify it and obtain an encoding into π-calculus:

$$
\begin{array}{rcl}
[\![\lambda x.\, M]\!]_p & \stackrel{\mathsf{def}}{=} & (\boldsymbol{\nu} q, x\,)(\overline{p}(w).\, w(y,r).\, (q \rhd r \mid x \rhd y) \mid [\![M]\!]_q) \\
[\![x]\!]_p & \stackrel{\mathsf{def}}{=} & \overline{x}\langle p\rangle \\
[\![MN]\!]_p & \stackrel{\mathsf{def}}{=} & \boldsymbol{\nu} q\ (\ \ [\![M]\!]_q \mid \\
 & & \qquad q(v).\, \boldsymbol{\nu} x\ (\overline{v}\langle x, p\rangle.\, !x(r).\, [\![N]\!]_r))
\end{array}
$$

where $a \rhd b \stackrel{\mathsf{def}}{=} !a(c).\, \overline{b}\langle c\rangle$.

But how can we prove this encoding correct?

- The encoding $[\![.]\!]$ of strong CBN into the π-calculus in slide 28 is *not* correct for the ordinary behavioural equivalences of the π-calculus (bisimulation, or even trace)
  For instance, there are M and N s.t. $\lambda\beta \vdash M = N$ but $[\![M]\!]_p$ and $[\![N]\!]_p$ are not bisimilar (early, late,...)

- However, the encoding *is* correct under the assumption that only the output capability of names may be passed. Under this assumption, for instance:

– $\lambda\beta \vdash M = N$ implies that $[\![M]\!]_p$ and $[\![N]\!]_p$ are behaviourally undistinguishable;
– $[\![M]\!]_p =_{\mathtt{must}} [\![N]\!]_p$ iff M and N have the same Böhm trees.

- **How can we prove these results?**

- A simple proof: compile π-calculus into πI and prove the theorems there.

- Similarly, we can prove the equality between the two processes P and Q in slide 9.

The simple theory of πI and the encodings of π-calculus into πI give rise to powerful proof techniques for proving properties of π-calculus processes.

We have seen these proof techniques in an example about functions as processes.

Another goal of the example was to show that links can be used to increase the parallelism in processes.

More on these proof techniques can be found in these papers:

- *The name discipline of uniform receptiveness*,
  D. Sangiorgi. Proc. ICALP'97.
- *Typed π-calculus at work: a correctness proof of Jones's parallelisation transformation on concurrent objects*,
  D. Sangiorgi, To appear in Theory and Practice of Object-Oriented Systems.

In πI we can carry out these proofs because πI is based on a more robust notion of *observable*.

Related work and future work

- Quaglia-Ferrari-Montanari's π-calculus with explicit substitutions
- Dale Miller's L_λ.
- $\pi\mathsf{I}^-$
- What is the logical meaning of the symmetries and dualities in $\pi\mathsf{I}$ and $\pi\mathsf{I}^-$?