Duality and i/o-types in the π-calculus

Picoq

Jean-Marie Madiot

2012-07-10

π-calculus

π is a *name-passing* process calculus. Some notations:

- $\overline{a}b$ sends the name b on the channel a;
- $a(x).P$ waits for some $\overline{a}b$ somewhere, then run $P[b/x]$;
- $\overline{a}b \mid a(x).P \to P[b/x]$.

Two λ-encodings that seem different

Milner's cbn encoding:

$$
\begin{aligned}
[\![x]\!]_p &= \overline{x}p \\
[\![\lambda x.M]\!]_p &= p(x,q).[\![M]\!]_q \\
[\![MN]\!]_p &= (\nu q)([\![M]\!]_q \mid (\nu x)(\overline{q}\langle x,p\rangle.!x(r).[\![N]\!]_r))
\end{aligned}
$$

van Bakel, Vigliotti's (strong) cbn encoding:

$$
\begin{aligned}
[\![x]\!]_p &= x(p').p' \multimap p \\
[\![\lambda x.M]\!]_p &= \overline{p}(x,q){:}[\![M]\!]_q \\
[\![MN]\!]_p &= (\nu q)([\![M]\!]_q \mid q(x,p').(p' \multimap p \mid !\overline{x}(r).[\![N]\!]_r))
\end{aligned}
$$

$a \multimap b \;=\; !a(x).\overline{b}x$ “link”

Duality in the π-calculus

Let's try switching inputs and outputs:

- $a(x).P \rightsquigarrow (\nu x)\overline{a}x.\overline{P}$: we can certainly send a new name.
- $\overline{a}b.P \rightsquigarrow (?)$: "freely" receiving a name?
  - solution 1: forbid free outputs (internal mobility)
  - solution 2: authorize free inputs and unify names (fusion calculi)

Internal mobility: πI [Sangiorgi, 1996]

Outputs cannot be free, only bound:

$$(\nu b)\overline{a}b.P = \overline{a}(b).P \text{ (notation)}$$

πI is a subcalculus of π, so we get this rule:

$$a(x).P \mid \overline{a}(x).Q \rightarrow (\nu x)(P \mid Q)$$

Consequences?

- simpler theory: $\sim$ is a congruence
- expressiveness: enough for the λ-calculus
- duality:

$$\overline{\overline{a}(x).x(y).(y \mid \overline{y})} \quad = \quad a(x).\overline{x}(y).(\overline{y} \mid y)$$

Fusion calculi [Parrow, Victor, Fu, Wischik, Gardner, Laneve, ...]

Authorizing both free outputs and free inputs. The objects *fuse*.

$$P ::= \bar{a}b.P \mid ab.P \mid (a = b) \mid (\nu a)P \mid \dots$$

$$\begin{array}{rcl} \bar{a}b.P \mid ac.Q & \rightarrow & P \mid Q \mid (b = c) \\ (a = b) \mid ac.P & \equiv & (a = b) \mid bc.P \\ (a = b) \mid \bar{a}c \mid bd & \rightarrow & (a = b) \mid (c = d) \end{array}$$

Consequences?

- nice theory: only one binder
- unique notion of bisimilarity (substitution-closed)
- duality:

$$\overline{\bar{a}b \mid ac \mid (d = e)} \;=\; ab \mid \bar{a}c \mid (d = e)$$

- links *vs* fusions

Types

Types provide safety, as always, but also a refined analysis of processes.

For example:

$$a : \sharp o T \quad \triangleright \quad (\nu b)\overline{a}b.\overline{b} \quad \simeq \quad (\nu b)\overline{a}b.0$$

Capability types (i/o-types) are a central type construct:

- $a : iT$ types processes **receiving** T-names on a;
- $a : oT$ types processes **sending** T-names on a;
- $a : \sharp T$ types processes doing both.

$$\frac{\Gamma \vdash a : iT \quad \Gamma, x : T \vdash P}{\Gamma \vdash a(x).P} \qquad \frac{\Gamma \vdash a : oT \quad \Gamma \vdash b : T}{\Gamma \vdash \overline{a}b}$$

Subtypes

Subtyping in name-passing:

$$T_1 \leq T_2 \text{ : any } T_1\text{-name is also a } T_2\text{-name.}$$

e.g. a $\sharp T$-name can be viewed as a iT-name.

Subtyping *in depth* is natural in i/o types; this rules follows for the operational semantics of π:

$$\frac{T_1 \leq T_2}{iT_1 \leq iT_2} \qquad \frac{T_1 \leq T_2}{oT_2 \leq oT_1}$$

Types and duality

Symmetric calculi and i/o-types don't go so well together:

- πI inherits i/o-types from π but they are not symmetric;
- fusion calculi are not compatible with *in-depth* subtyping.

$$c : i, \quad a : \sharp i, \quad b : o \quad \vdash \quad \overline{a}b \mid ac \mid \overline{b}$$

$$c : i \quad \vdash \quad (\nu ab)(\overline{a}b \mid ac \mid \overline{b}) \to \overline{c}$$

$\overline{\pi}$: π with typed duality

A calculus containing π and the syntactic dual of π:

$$P ::= \overline{a}b \mid ab \mid a(x).P \mid \overline{a}(x).P \mid (\nu a)P \mid \ldots$$

and behaving like π:

$$\begin{array}{rcll} \overline{a}b.P \mid a(x).Q & \to & P \mid Q[b/x] & \text{(as in } \pi\text{)} \\ ab.P \mid \overline{a}(x).Q & \to & P \mid Q[b/x] & \text{(dual of above)} \\ a(x).P \mid \overline{a}(x).Q & \to & (\nu x)(P \mid Q) & \text{(as in } \pi\text{I)} \\ \overline{a}b \mid ac & \not\to & & \text{(no fusion)} \end{array}$$

We create two sorts to forbid the last reduction: FO and FI.

- FO allows only free outputs: $\overline{a}b, a(x).P, \overline{a}(x).P$ (like in π)
- FI allows only free inputs: $ab, \overline{a}(x).P, a(x).P$

Properties of $\overline{\pi}$

We still have subtyping in depth:

- in the *FO* world, i is covariant and o contravariant
- in the *FI* world, i is contravariant and o covariant

Operational duality is straightforward:

$$P \to P' \iff \overline{P} \to \overline{P'} \qquad P \simeq Q \iff \overline{P} \simeq \overline{Q}$$

Typing is built towards duality, too:

$$a : i^{FO}T \vdash P \iff a : o^{FI}\overline{T} \vdash \overline{P} \qquad T_1 \leq T_2 \iff \overline{T_1} \leq \overline{T_2}$$

More importantly (and new), the typed barbed congruence:

$$\Gamma \vdash P \simeq Q \iff \overline{\Gamma} \vdash \overline{P} \simeq \overline{Q}$$

$\overline{\pi}$ compared to π

$\overline{\pi}$ contains π syntactically, but it is also very close to π:

Theorem ($\overline{\pi}$ is a conservative extension of π)

If Γ, P, Q are in π then:

$$\Gamma \vdash P \simeq_{\pi} Q \quad \Leftrightarrow \quad \Gamma \vdash P \simeq_{\overline{\pi}} Q \ .$$

Back to the two λ-encodings

Milner's cbn encoding:

$$\llbracket x \rrbracket_p = \overline{x}p$$
$$\llbracket \lambda x.M \rrbracket_p = p(x,q).\llbracket M \rrbracket_q$$
$$\llbracket MN \rrbracket_p = (\nu q)(\llbracket M \rrbracket_q \mid (\nu x)(\overline{q}\langle x,p\rangle.!x(r).\llbracket N \rrbracket_r))$$

van Bakel, Vigliotti's (strong) cbn encoding:

$$\llbracket x \rrbracket_p = x(p').p' \multimap p$$
$$\llbracket \lambda x.M \rrbracket_p = \overline{p}(x,q){:}\llbracket M \rrbracket_q$$
$$\llbracket MN \rrbracket_p = (\nu q)(\llbracket M \rrbracket_q \mid q(x,p').(p' \multimap p \mid !\overline{x}(r).\llbracket N \rrbracket_r))$$

Differences and similaritites

$$[\![x]\!]_p = \overline{x}p \qquad \text{Milner}$$
$$[\![\lambda x.M]\!]_p = p(x,q).[\![M]\!]_q$$

$$[\![x]\!]_p = x(p').p' \multimap p \qquad \text{van Bakel,}$$
$$[\![\lambda x.M]\!]_p = \overline{p}(x,q):[\![M]\!]_q \qquad \text{Vigliotti}$$

Differences:

- inputs and outputs switched (duality)
- strong cbn (not a problem)
- usage of links: $a \multimap b \;=\; !a(x).\overline{b}x$

We would like to relate them!

Available tools

We need a setting:

- big enough to contain both encodings $[\![\cdot]\!]$ and $[\![\cdot]\!]$;
- powerful to study link processes (types);
- closed by duality;
- close enough to π.

What do we have?

- π**I**:
  - hard to encode (not ALπ);
  - hard to relate to π;
- fusion calculi:
  - hard to relate to π (not a conservative extension);
  - not enough types
- $\overline{\pi}$: has both types and duality, close to π.

First step: duality

From $[\![\, \cdot \,]\!]$ (in the *FO* part of $\overline{\pi}$) we get $\overline{[\![\, \cdot \,]\!]}$.

$\overline{[\![\, \cdot \,]\!]}$ is in the *FI* part of $\overline{\pi}$. (e.g. $\overline{[\![x]\!]_p} = xp$)

Second step: link transformation

We define a **link transformation** $[\![\cdot]\!]_\ell$:

$$[\![ab.P]\!]_\ell = a(x).(x \multimap b \mid [\![P]\!]_\ell) \quad ; \quad [\![\overline{a}b]\!]_\ell = \overline{a}b \quad ; \quad \ldots$$

Types are fundamental:

- $[\![\cdot]\!]_\ell$ demands **asynchrony** and **output-capability** for the links to work
- $[\![\cdot]\!]_\ell$ transforms *FI*-types into *FO*-types

$\overline{[\![\, \cdot \,]\!]}$ is in the *FI*-part of $\overline{\pi}$.

$\left[\!\!\left[\overline{[\![\, \cdot \,]\!]}\right]\!\!\right]_\ell$ is in the *FO*-part of $\overline{\pi}$.

Composing

From $[\![\cdot]\!]$ we get $\overline{[\![\cdot]\!]}$ and then $\left[\!\!\left[\overline{[\![\cdot]\!]}\right]\!\!\right]_\ell$ which is exactly $[\![\cdot]\!]$.

Conclusion

Contributions:

- a calculus, $\overline{\pi}$,
  - (first) typed duality
  - operationally close to π (with types)
  - useful enough to handle λ-encodings
- a link transformation,
- related two different λ-encodings.

Future work:

- investigating the subtypes in symmetric calculi,
- a theory of links in a more general framework.

Typing rules of $\overline{\pi}$

$$\frac{\Gamma \vdash a : i^{FO}\,T \qquad \Gamma, x : T \vdash P}{\Gamma \vdash a(x).P} \qquad \frac{\Gamma \vdash a : i^{FI}\,T \qquad \Gamma, x : T^{\leftrightarrow} \vdash P}{\Gamma \vdash a(x).P}$$

$$\frac{\Gamma \vdash a : o^{FI}\,T \qquad \Gamma, x : T \vdash P}{\Gamma \vdash \overline{a}(x).P} \qquad \frac{\Gamma \vdash a : o^{FO}\,T \qquad \Gamma, x : T^{\leftrightarrow} \vdash P}{\Gamma \vdash \overline{a}(x).P}$$

$$\frac{\Gamma \vdash a : i^{FI}\,T \qquad \Gamma \vdash b : T \qquad \Gamma \vdash P}{\Gamma \vdash ab.P}$$

$$\frac{\Gamma \vdash a : o^{FO}\,T \qquad \Gamma \vdash b : T \qquad \Gamma \vdash P}{\Gamma \vdash \overline{a}b.P} \qquad \frac{\Gamma, a : T \vdash P}{\Gamma \vdash (\nu a)P}$$

$$\frac{\Gamma \vdash P \qquad \Gamma \vdash Q}{\Gamma \vdash P \mid Q} \qquad \frac{\Gamma \vdash P}{\Gamma \vdash !P} \qquad \frac{}{\Gamma \vdash 0} \qquad \frac{\Gamma(a) \leq T}{\Gamma \vdash a : T}$$