Evaluating functions as processes

Beniamino Accattoli

Carnegie Mellon University

Functions as processes

- λ-**calculus** model of **functional programming**.
- π-**calculus** model for **concurrency**.
- λ-calculus can be simulated in the π-calculus (Milner, 1992).
- The simulation is **subtle**, not tight as one would expect.
- Here refined using:
  - **Linear logic**;
  - DeYoung-Pfenning-Toninho-Caires **session types** (but no types here);
  - A **novel approach** to relate terms and proof nets.
- **Contribution**:
  Original and simple presentation, revisiting a work by Damiano Mazza.

The simulation

The expected simulation:

$$
\begin{tikzcd}
t \arrow[r, "\beta"'] \arrow[d, dashed] & s \\
P_t &
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
t \arrow[r, "\beta"'] \arrow[d, dashed] & s \arrow[d, dashed] \\
P_t \arrow[r, "*", "\pi"'] & P_s
\end{tikzcd}
$$

Does not hold, there is a mismatch about reduction. One gets:

$$
\Rightarrow \quad
\begin{tikzcd}
t \arrow[rr, "\beta"'] \arrow[d, dashed] & & s \arrow[d, dashed] \\
P_t \arrow[r, "*", "\pi"'] & Q \sim & P_s
\end{tikzcd}
$$

where $\sim$ is **strong bisimulation** (with respect to the environment).

Improved simulation

We refine λ-calculus and (head) β-reduction to a reduction $\multimap$ s.t.:

$$\begin{tikzcd} t \arrow[r, multimap] \arrow[d, dashed] & s \\ P_t & \end{tikzcd} \quad \Rightarrow \quad \begin{tikzcd} t \arrow[r, multimap] \arrow[d, dashed] & s \arrow[d, dashed] \\ P_t \arrow[r, "\pi"'] & P_s \end{tikzcd}$$

and

$$\begin{tikzcd} t \arrow[d, dashed] & \\ P_t \arrow[r, "\pi"'] & Q \end{tikzcd} \quad \Rightarrow \exists s \text{ s.t.} \quad \begin{tikzcd} t \arrow[r, multimap] \arrow[d, dashed] & s \arrow[d, dashed] \\ P_t \arrow[r, "\pi"'] & Q \end{tikzcd}$$

Novelty: the translation is a **strong bisimulation of reductions**.

Intuitions

- **π-calculus** evaluates terms in **small steps** (abstract machine).
- Small-step evaluation $\simeq$ **λ-calculus + explicit substitutions** (ES).
- λ + ES **injects in linear logic** (LL).
- **Pfenning-Caires et al.**: linear logic **injects in the π-calculus**.
- **Schema**:

$$(\lambda \quad \subseteq) \quad \lambda + ES \quad \subseteq \quad LL \quad \subseteq \quad \pi$$

- **Here**: we **pull back** π-reduction to λ + ES, hiding LL.

Outline

1. TERM(s and)GRAPH(s)

2. π-calculus

Explicit substitutions

- Refine λ-calculus with **explicit substitutions**:

$$t, s, u \quad := \quad x \quad | \quad \lambda x.t \quad | \quad ts \quad | \quad t[x/s]$$

- **Evaluation** contexts (**weak head** contexts):

$$E \quad := \quad \langle\!\langle \cdot \rangle\!\rangle \quad | \quad Es \quad | \quad E[x/s]$$

- **Substitution** contexts (or **L**ists of substitutions):

$$L \quad := \quad \langle\!\langle \cdot \rangle\!\rangle \quad | \quad L[x/s]$$

- Rewriting strategy (closed by evaluation contexts $E\cdot$):

$$L\langle\!\langle \lambda x.t \rangle\!\rangle s \quad \multimap_{\mathtt{dB}} \quad L\langle\!\langle t[x/s] \rangle\!\rangle$$

$$E\langle\!\langle x \rangle\!\rangle [x/s] \quad \multimap_{\mathtt{ls}} \quad E\langle\!\langle s \rangle\!\rangle [x/s]$$

- Rewriting strategy (closed by evaluation contexts $E\cdot$):

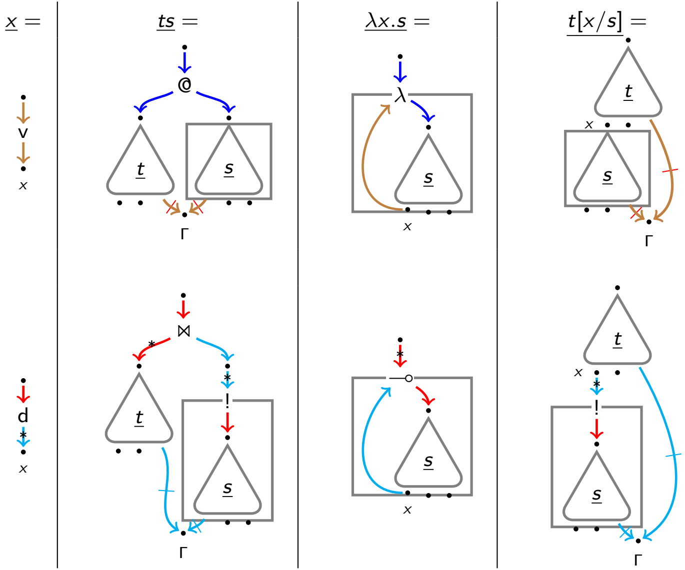
$$\begin{array}{rcl} L\langle\!\langle \lambda x.t \rangle\!\rangle s & \multimap_{\mathtt{dB}} & L\langle\!\langle t[x/s] \rangle\!\rangle \\ E\langle\!\langle x \rangle\!\rangle [x/s] & \multimap_{\mathtt{ls}} & E\langle\!\langle s \rangle\!\rangle [x/s] \end{array}$$

  is **linear weak head reduction**
  (Game semantics, KAM, Bohm's theorem, Geometry of interaction).

- Use of contexts in rules = **Distance**.

- **Example of reduction**:

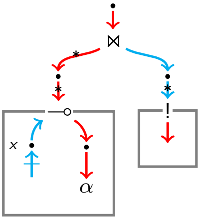
$$\begin{array}{rcl} (\lambda x.xx)\lambda y.yy & \multimap_{\mathtt{dB}} & (xx)[x/\lambda y.yy] \\ & \multimap_{\mathtt{ls}} & ((\lambda y.yy)x)[x/\lambda y.yy] \\ & \multimap_{\mathtt{dB}} & ((yy)[y/x])[x/\lambda y.yy] \\ & \multimap_{\mathtt{ls}} & ((xy)[y/x])[x/\lambda y.yy] \\ & \multimap_{\mathtt{ls}} & (((\lambda y.yy)y)[y/x])[x/\lambda y.yy] \quad \ldots \end{array}$$



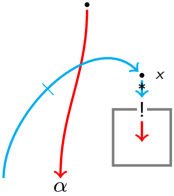
NOTE: the hole of an evaluation context is out of all boxes.

$$E \quad := \quad \llparenthesis \cdot \rrparenthesis \quad | \quad Es \quad | \quad E[x/s]$$

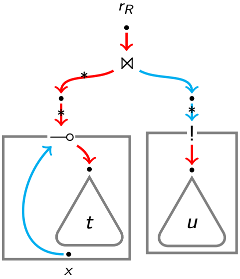
Multiplicative rule



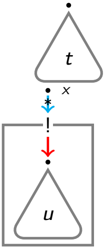
$\multimap_{\mathtt{dB}}$



$$(\lambda x.t)\ u \qquad \longrightarrow \qquad t[x/u]$$



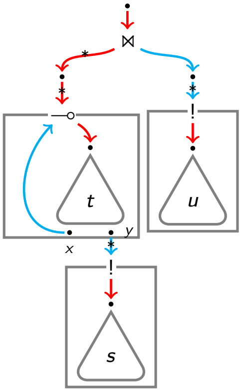
$\multimap_{\mathtt{dB}}$



Distance

The translation is **not injective**, in particular if $y \notin \mathtt{fv}(u)$:

$$\underline{((\lambda x.t)u)[y/s]} \;=\; \underline{(\lambda x.t)[y/s]u} \;=$$



So, both $((\lambda x.t)u)[y/s]$ and $(\lambda x.t)[y/s]u$ **have a redex!**

More on distance

- Rule **at a distance**:

$$L\langle\!|\lambda x.t|\!\rangle s \quad \multimap_{\mathtt{dB}} \quad L\langle\!|t[x/s]|\!\rangle$$

$$(\lambda x.t)[\cdot/\cdot]\ldots[\cdot/\cdot]\ u \rightarrow_{\mathtt{B-distance}} t[x/u][\cdot/\cdot]\ldots[\cdot/\cdot]$$

- Traditionally a configuration like:

$$(\lambda x.t)[y/v]\ u$$

is not a redex, as it is **blocked** by $[y/v]$.

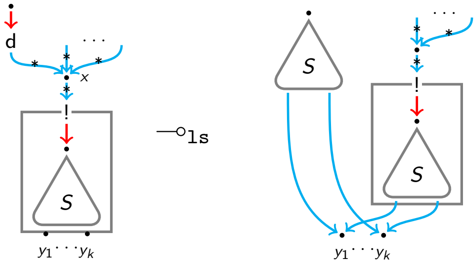
- Here, instead, **it is a redex**.

Substitution rule

The substitution rule:

$$E\langle\!(x)\!\rangle[x/s] \quad \multimap_{\mathtt{ls}} \quad E\langle\!(s)\!\rangle[x/s]$$

Corresponds to:



Note: the substituted variable/dereliction is **out of all boxes**.

Strong bisimulation

ES at a distance and **proof nets** satisfy:

$$
\begin{tikzcd}
t \arrow[r, -o] \arrow[d, dashed] & s \\
G_t &
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
t \arrow[r, -o] \arrow[d, dashed] & s \arrow[d, dashed] \\
G_t \arrow[r, -o] & G_s
\end{tikzcd}
$$

and

$$
\begin{tikzcd}
t \arrow[d, dashed] & \\
G_t \arrow[r, -o] & G'
\end{tikzcd}
\quad \Rightarrow \exists s \text{ s.t.} \quad
\begin{tikzcd}
t \arrow[r, -o] \arrow[d, dashed] & s \arrow[d, dashed] \\
G_t \arrow[r, -o] & G'
\end{tikzcd}
$$

Idea: distance turns terms in an algebraic language for graphs.

Outline

π-calculus

- **Processes**:

$$P, Q, R \quad := \quad 0 \mid \overline{x}\langle y\rangle \mid \overline{x}\langle y, z\rangle \mid \nu x P \mid x(y,z).P \mid !x(y).P \mid P \mid Q$$

- **Non-blocking contexts**:

$$N \quad := \quad (\!| \cdot |\!) \quad \mid \quad N \mid Q \quad \mid \quad P \mid N \quad \mid \quad \nu x N$$

- **Structural congruence** $\equiv$: closure by $N(\!| \cdot |\!)$ of

$$\frac{}{P \mid 0 \equiv P} \qquad \frac{}{P \mid (Q \mid R) \equiv (P \mid Q) \mid R} \qquad \frac{}{P \mid Q \equiv Q \mid P}$$

$$\frac{}{\nu x 0 \equiv 0} \qquad \frac{x \notin \mathtt{fn}(P)}{P \mid \nu x Q \equiv \nu x.(P \mid Q)} \qquad \frac{}{\nu x \nu y P \equiv \nu y \nu x P}$$

π-calculus

- **Substitution** of y to x in P is $P\{x/y\}$.
- **The rewriting rules** (closed by $N(\!|\cdot|\!)$ and $\equiv$):

$$\overline{x}\langle y, z\rangle \mid x(y', z').P \quad \rightarrow_{\otimes} \quad P\{y'/y\}\{z'/z\}$$

$$\overline{x}\langle y\rangle \mid !x(z).P \quad \rightarrow_{!} \quad P\{z/y\} \mid !x(z).Q$$

- **Binary** communication $=$ **multiplicative** cut-elimination
- **Unary** communication $=$ **exponential** cut-elimination
- Variation on the rules due to **Pfenning-Caires et al**.

Milner's translation with ES

- The translation from $\lambda + ES$ to π is **parametrized by a name**.
- Minor variation over Milner's translation:

$$
\begin{array}{rcll}
[\![x]\!]_a & := & \overline{x}\langle a\rangle & \\
[\![\lambda x.t]\!]_a & := & a(x,b).[\![t]\!]_b & \\
[\![ts]\!]_a & := & \nu b \nu x([\![t]\!]_b \mid \overline{b}\langle x,a\rangle \mid !x(c).[\![s]\!]_c) & x \text{ is fresh} \\
[\![t[x/s]]\!]_a & := & \nu x([\![t]\!]_a \mid !x(b).[\![s]\!]_b) &
\end{array}
$$

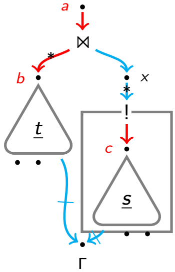
- **Red names** correspond to **multiplicative** formulas.
- Usual names (x,y,...) correspond to **exponential** formulas.

Proof nets as processes

$[\![x]\!]_a =$

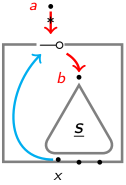
$= \overline{x}\langle a\rangle$

$[\![ts]\!]_a =$



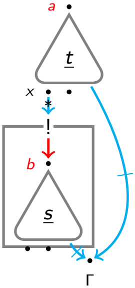
$= \nu b \nu x([\![t]\!]_b \mid \overline{b}\langle x, a\rangle \mid !x(c).[\![s]\!]_c)$

$[\![\lambda x.s]\!]_a =$



$= a(x, b).[\![t]\!]_b$

$[\![t[x/s]]\!]_a =$



$= \nu x([\![t]\!]_a \mid !x(b).[\![s]\!]_b)$

Term reductions to process reductions

Lemma (action on contexts)

$[\![E(\!| \cdot |\!)]\!]_a = N(\!|[\![\cdot]\!]_{a'}|\!)$

Proof.

Straightforward induction on $E(\!| \cdot |\!)$. □

Theorem (Strong simulation)

1. $t \multimap_{\mathtt{dB}} s$ *implies* $[\![t]\!]_a \Rightarrow_{\otimes} \equiv [\![s]\!]_a$.
2. $t \multimap_{\mathtt{ls}} s$ *implies* $[\![t]\!]_a \Rightarrow_{!} \equiv [\![s]\!]_a$.

Proof.

By induction on $t \multimap_{\mathtt{dB}} s$ and $t \multimap_{\mathtt{ls}} s$, using the lemma. □

Converse relation and distance

- **Distance for π** $\Rightarrow$ simpler converse relation.
- **The traditional rewriting rules** (closed by $N\llparenthesis \cdot \rrparenthesis$ and $\equiv$):

$$\overline{x}\langle y, z\rangle \mid x(y', z').P \quad \to_{\otimes} \quad P\{y'/y\}\{z'/z\}$$

$$\overline{x}\langle y\rangle \mid !x(z).P \quad \to_{!} \quad P\{z/y\} \mid !x(z).Q$$

- The **rewriting rules at a distance** (closed by $N'\llparenthesis \cdot \rrparenthesis$ only):

$$N\llparenthesis\overline{x}\langle y, z\rangle\rrparenthesis \mid M\llparenthesis x(y', z').P\rrparenthesis \quad \mapsto_{\otimes} \quad M\llparenthesis N\llparenthesis P\{y'/y\}\{z'/z\}\rrparenthesis\rrparenthesis$$

$$N\llparenthesis\overline{x}\langle y\rangle\rrparenthesis \mid M\llparenthesis !x(z).P\rrparenthesis \quad \mapsto_{!} \quad M\llparenthesis N\llparenthesis P\{z/y\} \mid !x(z).P\rrparenthesis\rrparenthesis$$

Reflecting process reductions

Lemma (reflection of reduction contexts)

If $N(\!|P|\!) \Rightarrow N(\!|Q|\!)$ *then* $\exists E(\!|\cdot|\!)$ *s.t.* $[\![E(\!|\cdot|\!)]\!]_a = N(\!|\cdot|\!)$.

Theorem (strong converse simulation)

1. *If* $[\![t]\!]_a \Rightarrow_{\otimes} Q$ *then exists* s *s.t.* $t \multimap_{\mathtt{dB}} s$ *and* $[\![s]\!]_a \equiv Q$.
2. *If* $[\![t]\!]_a \Rightarrow_{!} Q$ *then exists* s *s.t.* $t \multimap_{\mathtt{ls}} s$ *and* $[\![s]\!]_a \equiv Q$.

Improved simulation

Summing up we obtain:

$$
\begin{tikzcd}
t \arrow[r, multimap] \arrow[d, dashed] & s \\
\color{red}{[\![} t \color{red}{]\!]_a} &
\end{tikzcd}
\quad \Rightarrow \quad
\begin{tikzcd}
t \arrow[r, multimap] \arrow[d, dashed] & s \arrow[d, dashed] \\
\color{red}{[\![} t \color{red}{]\!]_a} \arrow[r, "\pi"'] & \color{red}{[\![} s \color{red}{]\!]_a}
\end{tikzcd}
$$

and

$$
\begin{tikzcd}
t \arrow[d, dashed] & \\
\color{red}{[\![} t \color{red}{]\!]_a} \arrow[r, "\pi"'] & Q
\end{tikzcd}
\quad \Rightarrow \exists s \text{ s.t.} \quad
\begin{tikzcd}
t \arrow[r, multimap] \arrow[d, dashed] & s \arrow[d, dashed] \\
\color{red}{[\![} t \color{red}{]\!]_a} \arrow[r, "\pi"'] & Q
\end{tikzcd}
$$

Call-by-value

- The same approach can be applied to the **call-by-value λ-calculus**.
- There is a **CBV translation** of λ-calculus in **linear logic**.
- I obtained a calculus strongly bisimilar to **CBV proof nets** [LSFA'12].
- One gets exactly the same **strong bisimulation**.
- A notion of CBV **linear weak head reduction**, which is **new**.

Conclusions

- **Distance** $=$ **rewriting rules via contexts** $=$
  Term rewriting matching **graph rewriting**.
- **General technique**, developed for ES, **working also for** π.
- Distance provides a **simple** and **elegant** re-understanding of $\lambda \hookrightarrow \pi$.
- Catching also the **call-by-value case**.
- **Unification** of λ + ES, linear logic, π-calculus (session types).

THANKS!