Functions as processes

- \( \lambda \)-calculus model of **functional programming**.
- \( \pi \)-calculus model for **concurrency**.

\( \lambda \)-calculus can be simulated in the \( \pi \)-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:

\[ t \xrightarrow{\beta} s, \quad P_t \Rightarrow \]

\[ t \xrightarrow{\beta} s, \quad P_t \Rightarrow^{*} P_s \]

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

\[ t \xrightarrow{\beta} s, \quad P_t \Rightarrow^{*} Q \sim P_s \]

where \( \sim \) is strong bisimulation (with respect to the environment).
Improved simulation

We refine $\lambda$-calculus and (head) $\beta$-reduction to a reduction $\rightarrow$ s.t.:

$$t \rightarrow^{\circ} s \quad \Rightarrow \quad t \rightarrow^{\circ} s$$

$$P_t \quad \Rightarrow \quad \exists s \text{ s.t.} \quad P_t \rightarrow^{\pi} P_s$$

and

$$t \quad \Rightarrow \exists s \text{ s.t.} \quad t \rightarrow^{\pi} Q$$

$$P_t \quad \Rightarrow \exists s \text{ s.t.} \quad P_t \rightarrow^{\pi} Q$$

**Novelty**: the translation is a strong bisimulation of reductions.
\( \pi\)-calculus evaluates terms in **small steps** (abstract machine).

Small-step evaluation \( \simeq \lambda\)-calculus + explicit substitutions (ES).

\( \lambda + ES \) **injects in linear logic** (LL).

Pfenning-Caires et al.: linear logic **injects in the \( \pi\)-calculus.**

**Schema:**

\[
(\lambda \subseteq) \quad \lambda + ES \subseteq LL \subseteq \pi
\]

**Here:** we pull back \( \pi\)-reduction to \( \lambda + ES \), hiding LL.
Outline

1. TERM(s and ) GRAPH(s)

2. $\pi$-calculus
Explicit substitutions

- Refine $\lambda$-calculus with **explicit substitutions**:
  \[
  t, s, u := x \mid \lambda x.t \mid ts \mid t[x/s]
  \]

- **Evaluation** contexts (weak head contexts):
  \[
  E ::= (\cdot) \mid Es \mid E[x/s]
  \]

- **Substitution** contexts (or Lists of substitutions):
  \[
  L ::= (\cdot) \mid L[x/s]
  \]

- Rewriting strategy (closed by evaluation contexts $E \cdot$):
  \[
  L(\lambda x.t)s \rightarrow_{db} L(t[x/s])
  \]
  \[
  E(x)[x/s] \rightarrow_{1s} E(s)[x/s]
  \]
Example of evaluation

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

$$L(\lambda x.t)s \xrightarrow{\circ_{dB}} L(t[x/s])$$
$$E(x)[x/s] \xrightarrow{\circ_{ls}} E(s)[x/s]$$

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

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

- **Example of reduction:**

  $$(\lambda x.xx)\lambda y.yy \xrightarrow{\circ_{dB}} (xx)[x/\lambda y.yy]$$
  $$\xrightarrow{\circ_{ls}} ((\lambda y.yy)x)[x/\lambda y.yy]$$
  $$\xrightarrow{\circ_{dB}} ((yy)[y/x])[x/\lambda y.yy]$$
  $$\xrightarrow{\circ_{ls}} (((\lambda y.yy)y)[y/x])[x/\lambda y.yy] \ldots$$
\[ x = x \]

\[ ts = \]

\[ \lambda x.s = \]

\[ t[x/s] = \]

\[ E ::= (\cdot) \mid Es \mid E[x/s] \]

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

\[(\lambda x.t) \ u \quad \rightarrow \quad t[x/u]\]
The translation is **not injective**, in particular if $y \notin \text{fv}(u)$:

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

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:

\[ L(\lambda x.t)s \rightarrow_{dB} L(t[x/s]) \]

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

- 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:

$$E(x)[x/s] \xrightarrow{\text{ls}} E(s)[x/s]$$

Corresponds to:

\[ S \]

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

ES at a distance and proof nets satisfy:

\[ t \rightarrow s \quad \Rightarrow \quad t \rightarrow s \]
\[ G_t \Downarrow \quad \Downarrow \quad G_t \rightarrow G_s \]

and

\[ t \Downarrow \quad \Rightarrow \exists s \text{ s.t. } t \Downarrow \quad G_t \rightarrow G' \]

\[ t \Downarrow \quad \Downarrow \quad G_t \rightarrow G' \]

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

1. TERM(s and )GRAPH(s)

2. $\pi$-calculus

Accattoli (CMU)  Evaluating functions as processes
\( \pi \)-calculus

- **Processes:**
  \[
P, Q, R \ := \ 0 \mid \overline{x}\langle y \rangle \mid \overline{x}\langle y, z \rangle \mid \nu x P \mid x(y, z).P \mid !x(y).P \mid P \mid Q
  \]

- **Non-blocking contexts:**
  \[
  N \ := \ (\cdot) \mid N \mid Q \mid P \mid N \mid \nu x N
  \]

- **Structural congruence \( \equiv \):** closure by \( N(\cdot) \) of
  \[
  \begin{array}{c}
P \mid 0 \equiv P \\
\nu x 0 \equiv 0 \\
(P \mid Q) \mid R \equiv (P \mid Q) \mid R \\
(vxQ \equiv \nu x.(P \mid Q)) \\
(P \mid Q) \equiv (P \mid Q) \mid R \\
vxvyP \equiv vyyvxP
  \end{array}
  \]
Substitution of $y$ to $x$ in $P$ is $P\{x/y\}$.

The rewriting rules (closed by $N(\cdot, \cdot)$ and $\equiv$):

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

$$\bar{x}\langle y \rangle \mid !x(z).P \rightarrow !_P P\{z/y\} \mid !x(z).Q$$

- **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 $\pi$ is **parametrized by a name**.

- Minor variation over Milner’s translation:

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

- **Red names** correspond to **multiplicative** formulas.

- Usual names ($x,y,...$) correspond to **exponential** formulas.
Proof nets as processes

\[ [x]_a = \bar{x}\langle a \rangle \]

\[ [ts]_a = \nu b \nu x (\llbracket t \rrbracket_b | \bar{b}\langle x, a \rangle | !x(c).\llbracket s \rrbracket_c) \]

\[ [\lambda x.s]_a = a(x, b).\llbracket t \rrbracket_b \]

\[ [t[x/s]]_a = \nu x(\llbracket t \rrbracket_a | !x(b).\llbracket s \rrbracket_b) \]
Lemma (action on contexts)

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

Proof.
Straightforward induction on \( E(\cdot) \).

Theorem (Strong simulation)

1. \( t \xrightarrow{\circ dB} s \) implies \([t]_a \Rightarrow \otimes \equiv [s]_a\).
2. \( t \xrightarrow{\circ ls} s \) implies \([t]_a \Rightarrow ! \equiv [s]_a\).

Proof.
By induction on \( t \xrightarrow{\circ dB} s \) and \( t \xrightarrow{\circ ls} s \), using the lemma.
Converse relation and distance

- **Distance for** $\pi \Rightarrow$ simpler converse relation.

- **The traditional rewriting rules** (closed by $N(\cdot \cdot)$ and $\equiv$):
  
  \[
  \overline{x}\langle y, z \rangle \mid x(y', z').P \quad \rightarrow \otimes \quad P\{y'/y\}\{z'/z\}
  \]
  
  \[
  \overline{x}\langle y \rangle \mid !x(z).P \quad \rightarrow ! \quad P\{z/y\} \mid !x(z).Q
  \]

- **The rewriting rules at a distance** (closed by $N'(\cdot \cdot)$ only):
  
  \[
  N(\overline{x}\langle y, z \rangle) \mid M(x(y', z').P) \quad \mapsto \otimes \quad M(N(P\{y'/y\}\{z'/z\}))
  \]
  
  \[
  N(\overline{x}\langle y \rangle) \mid M(!x(z).P) \quad \mapsto ! \quad M(N(P\{z/y\} \mid !x(z).P))
  \]
Lemma (reflection of reduction contexts)

If $N(P) \Rightarrow N(Q)$ then $\exists E(\cdot) \ s.t. \ \llbracket E(\cdot) \rrbracket_a = N(\cdot)$.

Theorem (strong converse simulation)

1. If $\llbracket t \rrbracket_a \Rightarrow_\otimes Q$ then exists s s.t. $t \xrightarrow{\text{dB}} s$ and $\llbracket s \rrbracket_a \equiv Q$.
2. If $\llbracket t \rrbracket_a \Rightarrow_! Q$ then exists s s.t. $t \xrightarrow{\text{ls}} s$ and $\llbracket s \rrbracket_a \equiv Q$. 
Summing up we obtain:

\[ t \xrightarrow{\circ} s \Rightarrow [t]_a \]

and

\[ t \xrightarrow{\circ} s \Rightarrow \exists s \text{ s.t.} [t]_a \xrightarrow{\pi} Q \]
The same approach can be applied to the **call-by-value \( \lambda \)-calculus**.

There is a **CBV translation** of \( \lambda \)-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 $\pi$.
- Distance provides a **simple** and **elegant** re-understanding of $\lambda \hookrightarrow \pi$.
- Catching also the **call-by-value case**.
- **Unification** of $\lambda +$ ES, linear logic, $\pi$-calculus (session types).
THANKS!