The Machinery of Interaction

Beniamino Accattoli[1] Ugo Dal Lago[2] Gabriele Vanoni[2]

[1]INRIA & LIX, École Polytechnique
[2]Università di Bologna & INRIA

PPDP, 10 September 2020

A paradigmatic functional PL: the Call-by-Name λ-calculus

$$
\begin{array}{rrcl}
\textsc{Terms} & t, u & ::= & x \mid \lambda x.t \mid tu \\
\textsc{Weak Head Contexts} & H & ::= & \langle\cdot\rangle \mid Hu \\
\textsc{Weak Head Reduction} & \multicolumn{3}{l}{H\langle(\lambda x.t)u\rangle \to_\beta H\langle t\{x{\leftarrow}u\}\rangle}
\end{array}
$$

Weak head β-reduction is **not** atomic!

How can we *implement* it?

One Solution: Abstract Machines

Abstract machines $\mathtt{M}$ implement strategies (*e.g.* weak head reduction).

$$\begin{tikzcd} t \arrow[r, "n"] \arrow[d] & u \\ \mathtt{M}_t \arrow[r, "*"] & \mathtt{M}_u \arrow[u] \end{tikzcd}$$

Every β-step is decomposed is micro-steps on $\mathtt{M}$.

A typical example for call-by-name is Krivine's abstract machine.

An Unusual Machine: the IAM

- Linear logic and the Geometry of Interaction inspired a different machine: the IAM.
- Pioneered by Mackie [POPL1995] and Danos & Regnier [LICS1996].
- It has always been defined on linear logic proof nets.
- Our contribution: new presentation on λ-terms and new correctness proof.

Why the IAM?

- **Mackie** [POPL1995]: small runtime system for PCF.
- **Ghica** [POPL2007]: compilation of higher-order functions to digital circuits.
- **Dal Lago** and **Schöpp** [ESOP2010]: functional programming in LOGSPACE.

The IAM: a Token Machine

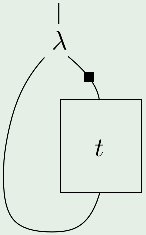
Ideas

1. No tracing of β-redexes
2. **Backtracking** to retrieve β-redexes
3. Computation is **local**
4. The **code** never changes.

Positions

The position of the token inside the term t is represented via a pair (u, C) such that $C\langle u\rangle = t$.

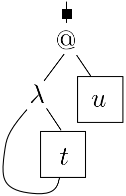
Example



The position of the token ■ in the term $\lambda x.t$ is represented by the position $(t, \lambda x.\langle\cdot\rangle)$.

Traversing β-redexes

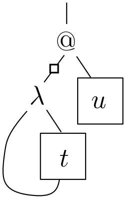
No information is saved, traversing a β-redex $(\lambda x.t)u$. The token remains untouched.



The token ■ is at the root of the redex.

Traversing β-redexes

No information is saved, traversing a β-redex $(\lambda x.t)u$. The token remains untouched.

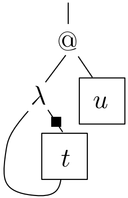


The token $\blacksquare$ moves to the left-hand side of the application, changing color.

$$ru \mid C \mid \blacksquare \quad \rightarrow_{\bullet 1} \quad r \mid C\langle\langle\cdot\rangle u\rangle \mid \square$$

Traversing β-redexes

No information is saved, traversing a β-redex $(\lambda x.t)u$. The token remains untouched.



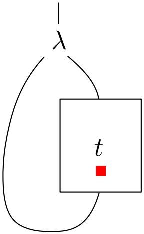
The token $\square$ moves to the body of the abstraction, changing color again.

$$\lambda x.t \mid C \mid \square \quad \rightarrow_{\bullet 2} \quad t \mid C\langle \lambda x.\langle \cdot \rangle \rangle \mid \blacksquare$$

Quering Variables and their Arguments

Computation in the λ-calculus is done by substituting variables for arguments.

Our machine first looks for variables, in red mode, going deep inside the term.

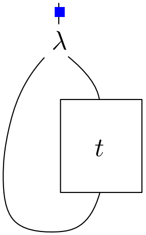


The token ■ is pointing a variable.

Quering Variables and their Arguments

Computation in the λ-calculus is done by substituting variables for arguments.

Our machine first looks for variables, in red mode, going deep inside the term.

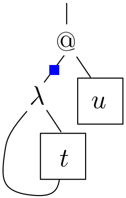


The token ■ moves locally to the binder, changing its mode to blue, *i.e.* the machine is now looking for the argument of the variable.

$$x \mid C\langle \lambda x.D \rangle \mid \blacksquare \quad \rightarrow_{\mathsf{var}} \quad \lambda x.D\langle x \rangle \mid C \mid \blacksquare$$

Evaluating Arguments

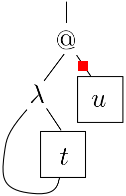
When an argument is found, it has to be evaluated.



The token ■, *i.e.* looking for an argument is on the left-hand side of an application.

Evaluating Arguments

When an argument is found, it has to be evaluated.



The token moves to the argument, changing its mode to red, *i.e.* looking for the head variable.

$$t \mid C\langle\langle\cdot\rangle u\rangle \mid \textcolor{blue}{\blacksquare} \quad \to_{\mathsf{arg}} \quad u \mid C\langle t\langle\cdot\rangle\rangle \mid \textcolor{red}{\blacksquare}$$

Hiding Some Details

There is also a **backtracking** mechanism (not explained here).

It uses position **enriched** with history/log.

The token is actually made of two data structures, tape and log.

The Lambda Interaction Abstract Machine: Transitions

Sub-term	Context	Log	Tape		Sub-term	Context	Log	Tape
$\underline{tu}$	C	L	T	$\to_{\bullet 1}$	$\underline{t}$	$C\langle\langle\cdot\rangle u\rangle$	L	$\bullet \cdot T$
$\underline{\lambda x.t}$	C	L	$\bullet \cdot T$	$\to_{\bullet 2}$	$\underline{t}$	$C\langle\lambda x.\langle\cdot\rangle\rangle$	L	T
$\underline{x}$	$C\langle\lambda x.D_n\rangle$	$L_n \cdot L$	T	$\to_{\mathsf{var}}$	$\lambda x.D_n\langle x\rangle$	$\underline{C}$	L	$(x, \lambda x.D_n, L_n) \cdot T$
$\underline{\lambda x.D_n\langle x\rangle}$	C	L	$(x, \lambda x.D_n, L_n) \cdot T$	$\to_{\mathsf{bt2}}$	x	$\underline{C\langle\lambda x.D_n\rangle}$	$L_n \cdot L$	T
t	$\underline{C\langle\langle\cdot\rangle u\rangle}$	L	$\bullet \cdot T$	$\to_{\bullet 3}$	tu	$\underline{C}$	L	T
t	$\underline{C\langle\lambda x.\langle\cdot\rangle\rangle}$	L	T	$\to_{\bullet 4}$	$\lambda x.t$	$\underline{C}$	L	$\bullet \cdot T$
t	$\underline{C\langle\langle\cdot\rangle u\rangle}$	L	$l \cdot T$	$\to_{\mathsf{arg}}$	$\underline{u}$	$C\langle t\langle\cdot\rangle\rangle$	$l \cdot L$	T
t	$\underline{C\langle u\langle\cdot\rangle\rangle}$	$l \cdot L$	T	$\to_{\mathsf{bt1}}$	$\underline{u}$	$C\langle\langle\cdot\rangle t\rangle$	L	$l \cdot T$

Implementation Theorem

The λ-term t has weak head normal form if and only if the IAM terminates on $(\underline{t}, \langle\cdot\rangle, \epsilon, \epsilon)$.

Proof idea: if $t \to_\beta u$, the path followed by the token travelling inside t is bisimilar to path followed inside u.

Further Details

- The IAM is correct also for head reduction.
- Moreover, the rules can accomodate the linear substitution calculus.
- Our formulation is isomorphic to the presentation on proof nets.

Conclusions

Work in progress

A **non-idempotent intersection type** system that characterizes the space consumption of the λIAM.

Already submitted work

- A **non-idempotent intersection type** system that characterizes the time consumption of the λIAM.
- A comparison of different machines w.r.t. time efficiency.

Future work

Formalize correctness and complexity properties in suitable proof-assistants.