Corrado Böhm on Normal Forms, Strong Normalization and Other

Benedetto Intrigila
University of "Tor Vergata", Rome, Italy
Programming in $\lambda$-calculus

1) Untyped $\lambda$-calculus (no type discipline)

2) Normal forms have a special status

3) Strong normalization (termination of every reduction sequence – when n.f. exists) is required
Some consequences

No fixed point operator $Y$ is available:

$Yf = f(Yf)$ and s.n. fails

Observe that $Y$ is (after Turing) the standard solution for implementing recursion in $\lambda$-calculus (and in functional programming)
Some consequences (2)

E.g.

If \( R_{fg} \) is the (unary) primitive recursion schema:

\[
X = \lambda f g x. \ if \ x = 0 \ then \ gx \ else \ f(X(x-1))
\]
Our “Solution”

“The Ant-Lion Paradigm for Strong Normalization”
by C. Böhm and B. Intrigila 1994
Why “Ant-lion” ???

Since we want to work with normal forms, it is natural to look for a reactive behavior like the ant-lion’s one:

- it waits for an ant to fall in its trap
- the fallen ant is eaten
- the ant-lion returns to the “wait state”
Why “Ant-lion” ????
A Simple Example

Let $I$ be the identity combinator, we want a term in normal form $M$ with the following behavior: $MI \rightarrow M$

We define $M$ as: $\langle H, H \rangle$

with $H \equiv \lambda x y. yxx$ and $\langle \ , \ \rangle$ is Church pair

(observe the similarity with $Y$)
A simple example (2)

The “ant-lion” $M$ waits for a $I$, “eats” the $I$ and returns to a “wait” state:

$$M \ I \rightarrow \langle \ H, \ H \rangle \ I \rightarrow \ IHH \rightarrow \HH \rightarrow \langle \ H, \ H \rangle$$
A More Complex Example

Let $I$ be the identity combinator, now we want a term $M ≡ \langle H, H, U \rangle$ with the following behavior:

$$M I \rightarrow \langle H, H, \langle O, U \rangle \rangle$$

so that $M$, when “eats” a $I$, “stores” one $O$ inside itself.

As “initial state” we choose $U ≡ \langle O, O \rangle$
A More Complex Example

To obtain this

we define $H \equiv \lambda x y z. zxx(\langle O, y \rangle)$

so that:

$$M \ I \rightarrow \langle H, H, \langle O, O \rangle \rangle \ I \rightarrow IHH\langle O, O \rangle \rightarrow$$

$$\rightarrow HH\langle O, O \rangle \rightarrow \langle H, H, \langle O, \langle O, O \rangle \rangle \rangle$$
Starting from the previous examples, we can ask whether this approach can cope with any kind of program, that is whether it is Turing complete.

We require a numeral system, with numerals in normal form, and such that for every recursive function $f$ there is a term $F$ (in n.f.) such that for every natural number $n$:

- $F \ n$ strongly normalizes to $m$ if $f(n) = m$, for some $m$;
- $F \ n$ has not normal form if $f(n)$ is undefined.
Turing completeness (2)

We gave a positive answer to problem stated before, in a very strong form:

Inside the $\lambda I$-calculus, if a numeral system is such that

- all the numerals have normal form;
- there exist a successor, predecessor and test for zero in normal form and strongly normalizing;

then it is possible to represent every recursive function in such a way that when the function is defined the corresponding term strongly normalizes to the value.
Is the “Ant-lion” useful?

We proved Turing completeness, but – besides some simple examples – we did not work out a good formalism to experiment a programming style in the direction sketched before.

This seems an aspect of the more general problem of a better understanding (and exploitation) of the fascinating double character of untyped \( \lambda \)-terms.
The CUCH Reloaded

For extensive computational experimentations there the need of a suitable \textit{reduction machine}.

What is the actual status of the CUCH Machine?

Stefano Guerrini et al. 1991-1996 (original development)

Luigi Mazzucchelli 1997-1998 (multiplatform)

Luigi Mazzucchelli 2018 (multiplatform and mobile)
The CUCH Reloaded (2)