λ-abstraction algorithms

Mark Price, University of Bath

11th November 2004

Motivation

$$()_\lambda \ : \ \text{Combinator System} \longrightarrow \lambda - \text{calculus}$$

$$()_{\mathsf{CL}} \ : \ \lambda - \text{calculus} \longrightarrow \text{Combinator Sys}$$

Explicit Translation from Combinato
λ-terms

$\mathsf{I}_\lambda = \lambda x.x$

$\mathsf{K}_\lambda = \lambda x, y.x$

$\mathsf{S}_\lambda = \lambda x, y, z.(xz)(yz)$

$\mathsf{B}_\lambda = \lambda x, y, z.x$

$\mathsf{C}_\lambda = \lambda x, y, z.xz$

$\mathsf{W}_\lambda = \lambda x, y.xyy$

λ-Abstraction Algorithm

Definition 1. *A λ-abstraction algorithm consumes*
and an identifier x to return an x-free term $[x]\mathsf{A}$ with
properties analogous to the λ-term $\lambda x.\mathsf{A}$. Thus

$$([x]\mathsf{A})x \rhd\!\!\rhd \mathsf{A}$$

where $\rhd\!\!\rhd$ is the combinator reduction relation.

Existence of a λ-Abstraction Algorit

Theorem 1. *Given a combinator system which only*
as it's combinators, then there exists a λ-abstraction
for this system.

Health Warning!

The combinator system which contains only the co
K and I does not have a λ-abstraction algorithm.

Curry-Schönfinkel Algorithm

$$
\begin{array}{lll}
(a) & [x]x & = \mathsf{I} \\
(b) & [x]Z & = \mathsf{K}Z \\
(e) & [x](PQ) & = \mathsf{S}([x]P)([x]Q)
\end{array}
$$

Curry-Feys Algorithm

$$
\begin{array}{llll}
(a) & [x]x & = \mathsf{I} & \\
(b) & [x]Z & = \mathsf{K}Z & \text{if } x \notin \partial Z \\
(c) & [x](QP) & = \mathsf{B}Q([x]P) & \text{if } x \notin \partial Q,\ \ x \\
(d) & [x](QP) & = \mathsf{C}([x]Q)P & \text{if } x \in \partial Q,\ \ x \\
(e) & [x](QP) & = \mathsf{S}([x]Q)([x]P) & \text{if } x \in \partial Q,\ \ x
\end{array}
$$

The Raw Grzegorczyk Algorithm

$$(a) \qquad [x]x = \mathsf{I}$$

$$(b) \qquad [x]Z = \mathsf{K}Z \qquad \text{if } x$$

$$(w) \qquad [x](QP) = \mathsf{W}\Big(\Big(\mathsf{B}\big(\mathsf{C}([x]Q)\big)\Big)([x]P)\Big) \qquad \text{if } x$$

The Cooked Grzegorczyk Algorith

$$
\begin{array}{llll}
(a) & [x]x & = \mathsf{I} & \\
(b) & [x]Z & = \mathsf{K}Z & \text{if } x \notin \partial Z \\
(c') & [x](Qx) & = Q & \text{if } x \notin \partial Q \\
(c) & [x](QP) & = \mathsf{B}Q([x]P) & \text{if } x \notin \partial Q, \\
(d) & [x](QP) & = \mathsf{C}([x]Q)P & \text{if } x \in \partial Q, \\
(w) & [x](QP) & = \mathsf{W}\Big(\Big(\mathsf{B}\big(\mathsf{C}([x]Q)\big)\Big)([x]P)\Big) & \text{if } x \in \partial Q,
\end{array}
$$

Curry on Grzegorczyk

'There is a rather large number of misprints and other
apparently caused by inadvertence'

A Daft Combinator

Definition 2. *A compound combinator,* A, *is daft if whenever*

$$\mathsf{A}t_m \ldots t_1 \rhd\!\!\!\rhd t$$

then

$$\partial t = \partial t_m \cup \ldots \cup \partial t_1$$

holds.

Daftness is Contagious

Theorem 2. *If* A *is built up from daft combinators then* A *is*

$\{\mathsf{B}, \mathsf{C}, \mathsf{W}\}$-**Simulation of** S

Theorem 3. *The cooked Grzegorczyk algorithm gives us a t*
that

$$S_g xyz \rhd\!\!\!\rhd (zx)(yx)$$

for all identifiers x,y,z.

Other Choices

$$S_f = \mathsf{B(B(BW)B)C)}$$

$$S_g = \mathsf{B(BW)(BBC)}$$

$$S_h = \mathsf{B(B(BW)C)(BB)}$$

Efficiency of S

$S_h xyz \triangleright (\mathsf{B}(\mathsf{BW})\mathsf{C})(\mathsf{BB}x)yz$

$\triangleright (\mathsf{B}(\mathsf{BW})\mathsf{C})\mathsf{B}(xy)z$

$\triangleright \mathsf{BW}(\mathsf{CB})(xy)z$

$\triangleright \mathsf{W}(\mathsf{CB}xy)z$

$\triangleright \mathsf{CB}xyzz$

$\triangleright \mathsf{C}x(yz)z$

$\triangleright xz(yz)$

$S_f xyz \triangleright \mathsf{B}(\mathsf{BW})\mathsf{B}(\mathsf{C}x)yz$

$\triangleright \mathsf{BW}(\mathsf{B}(\mathsf{C}x))yz$

$\triangleright \mathsf{W}(\mathsf{B}(\mathsf{C}x)y)z$

$\triangleright \mathsf{B}(\mathsf{C}x)yzz$

$\triangleright \mathsf{C}x(yz)z$

$\triangleright xz(yz)$

$S_g xyz \triangleright ($

$\triangleright ($

$\triangleright$ W

$\triangleright ($

$\triangleright ($

$\triangleright x$