Constantes no Cálculo Lambda Puro

- A Tese de Church implica na existência de uma maneira de definir **números inteiros positivos** em cálculo λ.

- Para definir funções de vários argumentos deve-se poder descrever **listas** em cálculo λ.

- Como muitas funções utilizam definições condicionais, deve-se poder definir **constantes e operações booleanas** em cálculo λ.

Listas em Cálculo λ

Definition. PAR

define $Pair \equiv \lambda a.\ \lambda b.\ \lambda f.\ f\ a\ b$
define $Head \equiv \lambda g.\ g\ (\lambda a.\ \lambda b.\ a)$
define $Tail \equiv \lambda g.\ g\ (\lambda a.\ \lambda b.\ b)$

Exemplo: $Head(Pair\ p\ q)$

$$
\begin{array}{ll}
\Rightarrow_\delta & (\lambda g.\ g\ (\lambda a.\ \lambda b.\ a))\ ((\lambda a.\ \lambda b.\ \lambda f.\ f\ a\ b)\ p\ q) \\
\Rightarrow_\beta & ((\lambda a.\ \lambda b.\ \lambda f.\ f\ a\ b)\ p\ q)\ (\lambda a.\ \lambda b.\ a) \\
\Rightarrow_\beta & ((\lambda b.\ \lambda f.\ f\ p\ b)\ q)\ (\lambda a.\ \lambda b.\ a) \\
\Rightarrow_\beta & (\lambda f.\ f\ p\ q)\ (\lambda a.\ \lambda b.\ a) \\
\Rightarrow_\beta & (\lambda a.\ \lambda b.\ a)\ p\ q \\
\Rightarrow_\beta & (\lambda b.\ p)\ q \\
\Rightarrow_\beta & p
\end{array}
$$

Definition. LISTA

define $Nil \equiv \lambda x.\ \lambda a.\ \lambda b.\ a$
define $[a] \equiv Pair\ a\ Nil$
define $[a, b] \equiv Pair\ a\ (Pair\ b\ Nil)$

$\vdots$

Exemplo:

$$Head(Tail\ (Tail\ [1, 2, 3, 4])) \Rightarrow 3$$

Inteiros em Cálculo λ

(Definição de Church)

Definition. INTEIROS

define $0 \equiv \lambda f.\, \lambda x.\, x$
define $1 \equiv \lambda f.\, \lambda x.\, f\ x$
define $2 \equiv \lambda f.\, \lambda x.\, f\ (f\ x)$
define $3 \equiv \lambda f.\, \lambda x.\, f\ (f\ (f\ x))$

$\vdots$

define $Succ \equiv \lambda n.\, \lambda f.\, \lambda x.\, f\ (n\ f\ x)$
define $Add \equiv \lambda m.\, \lambda n.\, \lambda f.\, \lambda x.\, m\ f\ (n\ f\ x)$

Exemplo:

$$
\begin{array}{lll}
Succ\, 2 & \Rightarrow_\delta & (\lambda n.\, \lambda f.\, \lambda x.\, f\ (n\ f\ x)\ \lambda g.\, \lambda y.\, g\ (g\ y) \\
 & \Rightarrow_\beta & \lambda f.\, \lambda x.\, f\ ((\lambda g.\, \lambda y.\, g\ (g\ y))\ f\ x) \\
 & \Rightarrow_\beta & \lambda f.\, \lambda x.\, f\ ((\lambda y.\, f\ (f\ y))\ x) \\
 & \Rightarrow_\beta & \lambda f.\, \lambda x.\, f\ (f\ (f\ x)) \\
 & = & 3
\end{array}
$$

Soma de Inteiros em Cálculo λ

Sejam N e M números inteiros:

$$\underset{N \equiv}{\lambda f.\, \lambda x.\, \underbrace{f\,(f\,\ldots(f}_{N \text{ vezes } f}\, x))} \qquad \underset{M \equiv}{\lambda f.\, \lambda x.\, \underbrace{f\,(f\,\ldots(f}_{M \text{ vezes } f}\, x))}$$

Então:

$$\begin{array}{lll} (N\ a) & \Rightarrow_\delta & (\lambda f.\, \lambda x.\, f\,(f\,\ldots(f\,x)))\,a \\ & \Rightarrow_\beta & \lambda x.\, a\,(a\,\ldots(a\,x)) \end{array}$$

$$((N\ a)\ b) \quad \Rightarrow_\beta \quad a\,(a\,\ldots(a\,b))$$

$$\begin{array}{lll} (M\ a)\,((N\ a)\ b) & \Rightarrow_\beta & a\,(a\,\ldots(a\,((N\ a)\ b))\ldots) \\ & \Rightarrow_\beta & \underbrace{a\,(a\,\ldots(a}_{M \times a}\,\underbrace{(a\ldots(a}_{N \times a}\,b)\ldots))\ldots) \end{array}$$

$$(M \times a) + (N \times a) = (M + N) \times a \equiv M + N$$

Exercício: Reduzir a expressão $Add\,2\,3$.

Booleanos em Cálculo λ

Definition. BOOLEANOS

define $T \equiv \lambda x.\, \lambda y.\, x$
define $F \equiv \lambda x.\, \lambda y.\, y$
define $Not \equiv \lambda x.\, ((x\, F)\, T)$
define $And \equiv \lambda x.\, \lambda y.\, ((x\, y)\, F)$
define $Or \equiv \lambda x.\, \lambda y.\, ((x\, T)\, y)$

Representação baseada no comando **IF**:

if B *then* $S1$ *else* $S2 \Rightarrow ((B\ S1)\ S2)$

$T \equiv$ Seletor do primeiro elemento da lista

$$\begin{array}{lll} ((T\, S1)\, S2) & \Rightarrow_\delta & (((\lambda x.\, \lambda y.\, x)\, S1)\, S2) \\ & \Rightarrow_\beta & ((\lambda y.\, S1)\, S2) \\ & \Rightarrow_\beta & S1 \end{array}$$

$F \equiv$ Seletor do segundo elemento da lista

$$\begin{array}{lll} ((F\, S1)\, S2) & \Rightarrow_\delta & (((\lambda x.\, \lambda y.\, y)\, S1)\, S2) \\ & \Rightarrow_\beta & ((\lambda y.\, y)\, S2) \\ & \Rightarrow_\beta & S2 \end{array}$$

NOT em Cálculo λ

define $Not \equiv \lambda x.\,((x\ F)\ T)$

Para esta definição estar correta deve ser verdade que

$Not\ F \Rightarrow T$ e $Not\ T \Rightarrow F$

$$
\begin{array}{lll}
Not\ F & \Rightarrow_\delta & (\lambda x.\,((x\ F)\ T))\ F \\
 & \Rightarrow_\beta & ((F\ F)\ T) \\
 & \Rightarrow_\delta & T
\end{array}
$$

$$
\begin{array}{lll}
Not\ T & \Rightarrow_\delta & (\lambda x.\,((x\ F)\ T))\ T \\
 & \Rightarrow_\beta & ((T\ F)\ T) \\
 & \Rightarrow_\delta & F
\end{array}
$$

AND e OR em Cálculo λ

define $And \equiv \lambda x.\, \lambda y.\, ((x\ y)\ F)$

$$\begin{array}{lll} And\, T\, F & \Rightarrow_\delta & (\lambda x.\, \lambda y.\, ((x\ y)\ F))\, T\, F \\ & \Rightarrow_\beta & (\lambda y.\, ((T\ y)\ F))\, F \\ & \Rightarrow_\beta & ((T\ F)\ F)) \\ & \Rightarrow_\delta & F \end{array}$$

define $Or \equiv \lambda x.\, \lambda y.\, ((x\ T)\ y)$

$$\begin{array}{lll} Or\, T\, F & \Rightarrow_\delta & (\lambda x.\, \lambda y.\, ((x\ T)\ y))\, T\, F \\ & \Rightarrow_\beta & (\lambda y.\, ((T\ T)\ y))\, F \\ & \Rightarrow_\beta & ((T\ T)\ F)) \\ & \Rightarrow_\delta & T \end{array}$$

Construções de Alto Nível

Definition.

define $Let\, x = E1\, in\, E2 \equiv (\lambda x.\, E2)\, E1$
define $E2\, Where\, x = E1 \equiv (\lambda x.\, E2)\, E1$
define $If\, B\, then\, S1\, else\, S2 \equiv ((B\ S1)\ S2)$

Exemplo:

$$Let\, x = 3\, in\, (\lambda y.\, If\, Zero\, y\, then\, 0\, else\, Add\, x\, y)\, 2$$

$$\begin{aligned}
&\Rightarrow_\delta (\lambda x.(\lambda y.\, If\, Zero\, y\, then\, 0\, else\, Add\, x\, y)\, 2)\, 3\\
&\Rightarrow_\beta (\lambda y.\, If\, Zero\, y\, then\, 0\, else\, Add\, 3\, y)\, 2\\
&\Rightarrow_\beta If\, Zero\, 2\, then\, 0\, else\, Add\, 3\, 2\\
&\Rightarrow_\delta If\, F\, then\, 0\, else\, Add\, 3\, 2\\
&\Rightarrow_\delta ((F\, 0)\ Add\, 3\, 2)\\
&\Rightarrow_\delta Add\, 3\, 2\\
&\Rightarrow_\delta 5
\end{aligned}$$