Un(i)typed λ-calculus

September 20, 2001

CS 131: Programming Languages

History

- In 1936,
  - Alan Turing invented Turing machines, defined a notion of computable functions
  - Alonzo Church invented λ-calculus, defined a notion of computable functions
  - Definitions of computability turn out to be the same.

Syntax

- Pure lambda calculus:

`M,N ::=`	`x`		*variables*
	|	`λx.M`	*functions*
	|	`M N`	*applications*

- That's it!

Note on Syntax

- Anonymous function values `λx.M` show up under many different guises in functional languages.

$x \longmapsto M$	(Math)
`(x) => M`	(rex)
`(lambda (x) M)`	(Scheme)
`fn x => M`	(SML)
`\x -> M`	(Haskell)

Conventions

- Terms differing only in names of bound variables are considered the same term
  - In the term $\lambda x.M$, variable x is bound in M
- Application associates leftward

$$xyzw \quad = \quad ((xy)z)w$$

- Function bodies are as large as possible.

$$\lambda x.yx = \lambda x.(yx) \quad \neq \quad (\lambda x.y)x$$

One-step β-Reduction

- The relation $\rightarrow_\beta$ is defined by:

$$\frac{}{(\lambda x.M)N \;\rightarrow_\beta\; M[x \rightarrow N]}$$

$$\frac{M \rightarrow_\beta M'}{M\ N \rightarrow_\beta M'\ N} \qquad \frac{N \rightarrow_\beta N'}{M\ N \rightarrow_\beta M\ N'}$$

$$\frac{M \rightarrow_\beta M'}{\lambda x.M \rightarrow_\beta \lambda x.M'}$$

β-Reduction Summarized

- Look for the pattern

$$(\lambda x.M)N$$

occurring anywhere, and replace it with

$$M[x \rightarrow N]$$

A Worked Example

$\underline{(\lambda b.(\lambda x.(\lambda y.((b\ y)\ x))))(\lambda w.(\lambda z.w))}$
$\rightarrow_\beta \lambda x.(\lambda y.(\underline{((\lambda w.(\lambda z.w))\ y)}\ x))$
$\rightarrow_\beta \lambda x.(\lambda y.(\underline{(\lambda z.y)\ x}))$
$\rightarrow_\beta \lambda x.(\lambda y.y)$

$\underline{(\lambda b.\lambda x.\lambda y.b\ y\ x)(\lambda w.\lambda z.w)}$
$\rightarrow_\beta \lambda x.\lambda y.\underline{(\lambda w.\lambda z.w)\ y}\ x$
$\rightarrow_\beta \lambda x.\lambda y.\underline{(\lambda z.y)\ x}$
$\rightarrow_\beta \lambda x.\lambda y.y$

Exercise

- Reduce the following term:
  $(\lambda x.x\ x)((\lambda y.y)(\lambda z.z))$

Other Relations

- The relation $\rightarrow_\beta^*$ is defined to be the reflexive, transitive closure of $\rightarrow_\beta$
  - i.e., 0 or more $\rightarrow_\beta$ steps.
- The relation $\leftrightarrow_\beta^*$ is defined to be the reflexive, transitive, *symmetric* closure of $\rightarrow_\beta$.
  - Often referred to as *conversion*
  - Provides a notion of program equivalence

Conversion Formalized

$$\frac{M_1 \rightarrow_\beta M_2}{M_1 \leftrightarrow_\beta^* M_2}$$

$$\frac{}{M \leftrightarrow_\beta^* M} \qquad \frac{M_2 \leftrightarrow_\beta^* M_1}{M_1 \leftrightarrow_\beta^* M_2}$$

$$\frac{M_1 \leftrightarrow_\beta^* M_2 \qquad M_2 \leftrightarrow_\beta^* M_3}{M_1 \leftrightarrow_\beta^* M_3}$$

Programming in λ-Calculus

- Want terms in the λ-calculus that "act like"
  - booleans
  - numbers
  - conditionals
  - arithmetic operations
  - pairs and projections
  - etc.
- Many different ways to do encodings
  - I'll just show one example of each

Encoding Booleans

- We use the following definition:

tt := λx.λy.x
ff := λx.λy.y
if := λz.z

if tt M N $\leftrightarrow_\beta^*$
if ff M N $\leftrightarrow_\beta^*$

Exercises

- Find a term **not** such that
  not tt $\leftrightarrow_\beta^*$ **ff**
  not ff $\leftrightarrow_\beta^*$ **tt**
- Define **and** and **or**

Pairs

- We use the following definition:

<M,N> := λf.f M N
fst := λp.(p (λx.λy.x))
snd := λp.(p (λx.λy.y))

- Show that
  fst <M,N> $\leftrightarrow_\beta^*$ M
  snd <M,N> $\leftrightarrow_\beta^*$ N

Natural Numbers

- Church numerals:
  ⌜0⌝ := λf.λb.b
  ⌜1⌝ := λf.λb.f(b)
  ⌜2⌝ := λf.λb.f(f(b))
  ...
  ⌜n⌝ := λf.λb.f^n(b)

⌜n⌝ x y
$\leftrightarrow_\beta^*$ x^n(y)

succ := λm.λf.λb.f(m f b)

Is this Reasonable?

- Yes!

$$
\begin{aligned}
&\textbf{succ}\ \ulcorner n \urcorner \\
&\quad = \quad (\lambda m.\lambda f.\lambda b.f(m\ f\ b))\ \ulcorner n \urcorner \\
&\quad \leftrightarrow_\beta^* \quad \lambda f.\lambda b.f(\ulcorner n \urcorner\ f\ b) \\
&\quad \leftrightarrow_\beta^* \quad \lambda f.\lambda b.f(f^n\ b) \\
&\quad = \quad \lambda f.\lambda b.f^{n+1}\ b \\
&\quad = \quad \ulcorner n+1 \urcorner
\end{aligned}
$$

Exercises

- Find an alternative definition for **succ**
  - Hint: $1+n = n+1$
- Find terms **plus** and **times** such that

  $\textbf{plus}\ \ulcorner m \urcorner\ \ulcorner n \urcorner \quad \leftrightarrow_\beta^*\ \ulcorner m+n \urcorner$

  $\textbf{times}\ \ulcorner m \urcorner\ \ulcorner n \urcorner \quad \leftrightarrow_\beta^*\ \ulcorner mn \urcorner$
- Find a term **iszero** such that

  $\textbf{iszero}\ \ulcorner 0 \urcorner \quad \leftrightarrow_\beta^*\ \textbf{tt}$

  $\textbf{iszero}\ \ulcorner n \urcorner \quad \leftrightarrow_\beta^*\ \textbf{ff}$ for any $n>0$