Computer Science 131
Programming Languages

November 9, 2000

Un(i)typed λ-calculus

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*
|	**lx.M**	*functions*
|	**M N**	*applications*

- That's it!

Conventions

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

  xyzw = ((xy)z)w

- Function bodies as as large as possible.

  lx.yx = lx.(yx) ¹ (lx.y)x

One-step b-Reduction

- The relation $\rightarrow_b$ is defined by:

$$\frac{}{(\lambda x.M)N \rightarrow_b M[x \rightarrow N]}$$

$$\frac{M \rightarrow_b M'}{M\ N \rightarrow_b M'\ N} \qquad \frac{N \rightarrow_b N'}{M\ N \rightarrow_b M\ N'}$$

$$\frac{M \rightarrow_b M'}{\lambda x.M \rightarrow_b \lambda x.M'}$$

b-Reduction

- The relation $\rightarrow_b^*$ is defined to be the reflexive, transitive closure of $\rightarrow_b$
  - i.e., 0 or more $\rightarrow_b$ steps.

- The relation $\leftrightarrow_b^*$ is defined to be the reflexive, transitive, *symmetric* closure of $\rightarrow_b$.

Example

- Consider the term
  (l**x.xx)((l****y.y)**(l**z.z))**

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:

$$\underline{\mathbf{tt}} \;:=\; \lambda x.\lambda y.x$$
$$\underline{\mathbf{ff}} \;:=\; \lambda x.\lambda y.y$$

$$\underline{\mathbf{tt}}\ M\ N \leftrightarrow_b^* ???$$
$$\underline{\mathbf{ff}}\ M\ N \leftrightarrow_b^* ???$$

Exercises

- Find a term not such that

  not **tt** $\leftrightarrow_b^*$ **ff**

  not **ff** $\leftrightarrow_b^*$ **tt**

- Define **and** and **or**

Pairs

- We use the following definition:

$$\langle M,N\rangle := \lambda f.f\ M\ N$$
$$M.1 := M\ \underline{tt}$$
$$M.2 := M\ \underline{ff}$$

- Show that

$$\langle M,N\rangle.1 \leftrightarrow_b^* M$$
$$\langle M,N\rangle.2 \leftrightarrow_b^* N$$

Exercises

- The following holds when **`M`** is a pair.

  `<M.1,M.2>` $\leftrightarrow_b^*$ **`M`**

- Is this true for all **`M`**?
  - Don't actually have mechanism to prove this yet
  - But do you have a guess?

Natural Numbers

- Church numerals:

0 := λb.λf.b

1 := λb.λf.f(b)

2 := λb.λf.f(f(b))

3 := λb.λf.f(f(f(b)))

...

n := λb.λf.f^n(b)

succ := λn. λb.λf.f(n b f)

Exercises

- Find an alternative definition for **succ**
  - Hint: **1+n = n+1**

- Find a term **iszero** such that

  iszero 0 $\leftrightarrow_b^*$ **tt**

  iszero n+1 $\leftrightarrow_b^*$ **ff**

- Find a terms **plus** and **times** such that

  plus m n $\leftrightarrow_b^*$ **m+n**

  times m n $\leftrightarrow_b^*$ **mn**

Predecessor

- Tricky to compute with Church numerals
  - Other encodings make it simpler.
- Key idea: consider the sequence

<0,0> <0,1> <1,2> <2,3> <3,4> ...

```
pred'  ::= λn.n <0,0>
                (λp.<p.2,succ(p.2)>)
pred   ::= λn.(pred' n).1
```

Recursive Definitions

- Consider the SML definition

```
fun fact(n) =
    if n=0 then 1 else n*fact(n-1)
```

- This is convenient syntax for

```
val rec fact =
  fn n =>
    (if n=0 then 1 else n*fact(n-1))
```

- The function we want is a solution to this equation

Fixed Points

- A *fixed point* of a function **`F`** is a value **`x`** s.t.

  `F(x) = x`

- Any solution of an equation

  `x = <stuff involving x>`

  is also a fixed point of the function

  `lx.<stuff involving x>`

Factorial

- Any solution to

```
val rec fact =
  fn n =>
    (if n=0 then 1 else n*fact(n-1))
```

  is a fixed point of

```
fn f =>
  fn n =>
    (if n=0 then 1 else n*f(n-1))
```

Facts

1. To encode recursive functions, it suffices to show how to find fixed points.
2. For *every* λ-calculus term M, exists N s.t.

$$\mathbf{M(N)} \leftrightarrow_b^* \mathbf{N}$$

3. These fixed points can be found uniformly. There is a term **Y** such that

$$\mathbf{M(Y(M))} \leftrightarrow_b^* \mathbf{Y(M)}$$

Namely,

$$\mathbf{Y} := \lambda\mathbf{f.(}\lambda\mathbf{x.f(xx))(}\lambda\mathbf{x.f(xx))}$$

Exercises

- Show that

  $$\mathbf{M(Y(M))} \leftrightarrow_b^* \mathbf{Y(M)}$$

- Define

  FACT := λ**f.**λ**n.(iszero n) 1**
  (times n (f (pred n)))

  fact := Y FACT

  show that

  $$\mathbf{fact}\ \underline{\mathbf{n}} \leftrightarrow_b^* \underline{\mathbf{n!}}$$

- What does **Y succ** look like?