Chapter 5: The Untyped Lambda Calculus

What is lambda calculus for?

Basics: syntax and operational semantics

Programming in the Lambda Calculus

Formalities (formal definitions)

What is Lambda calculus for?

- A core calculus (used by Landin) for
  - capturing the language's essential mechanisms,
  - with a collection of convenient derived forms whose behavior is understood by translating them into the core

- A formal system invented in the 1920s by Alonzo Church (1936, 1941), in which all computation is reduced to the basic operations of function definition and application.

Basics

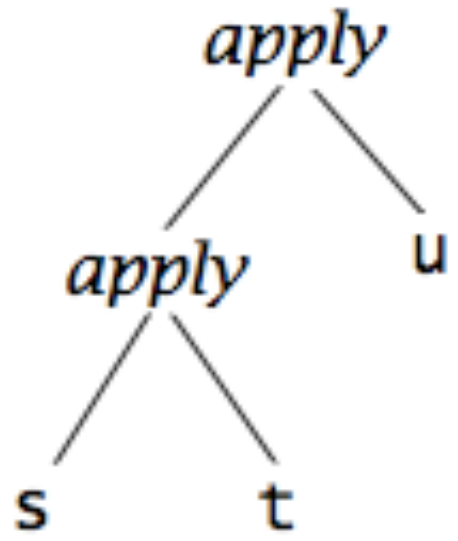
Syntax

- The lambda-calculus (or λ-calculus) embodies this kind of function definition and application in the purest possible form.

t	::=		*terms:*
		x	*variable*
		λx.t	*abstraction*
		t t	*application*

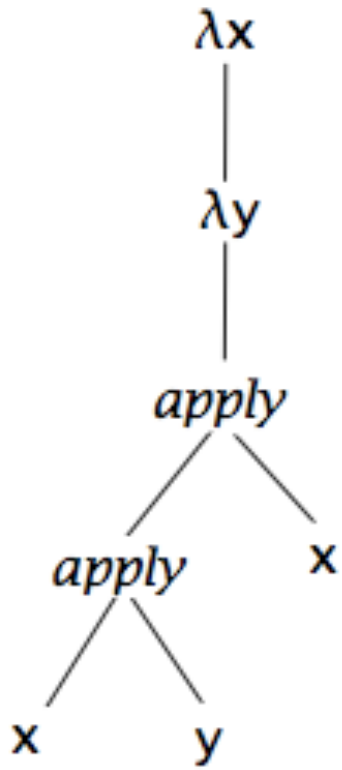
Abstract Syntax Trees

- (s t) u   (or simply written as s t u)



Abstract Syntax Trees

- $\lambda x.\ (\lambda y.\ ((x\ y)\ x))$
  (or simply written as $\lambda x.\ \lambda y.\ x\ y\ x$)



Scope

- An occurrence of the variable x is said to be bound when it occurs in the body t of an abstraction λx.t.
  - λx is a binder whose scope is t. A binder can be renamed: e.g., λx.x = λy.y.

- An occurrence of x is free if it appears in a position where it is not bound by an enclosing abstraction on x.
  - **Exercises**: Find free variable occurrences from the following terms: x y, λx.x, λy. x y, (λx.x) x.

Operational Semantics

- Beta-reduction: the only computation

$$(\lambda x.\ t_{12})\ t_2 \longrightarrow [x \mapsto t_2]t_{12},$$

"the term obtained by replacing all free occurrences of x in t_{12} by t_2 "

A term of the form (λx.t12) t2 is called a redex.

- Examples:

(λx.x) y → y

(λx. x (λx.x)) (u r) → u r (λx.x)

Evaluation Strategies

- Full beta-reduction
  - Any redex may be reduced at any time.

- Example:
  - Let id = λx.x. We can apply veta reduction to any of the following underlined redexes:

$$\underline{\mathrm{id}\ (\mathrm{id}\ (\lambda z.\ \mathrm{id}\ z))}$$

$$\mathrm{id}\ (\underline{(\mathrm{id}\ (\lambda z.\ \mathrm{id}\ z))})$$

$$\mathrm{id}\ (\mathrm{id}\ (\lambda z.\ \underline{\mathrm{id}\ z}))$$

Evaluation Strategies

- The normal order strategy
  - The leftmost, outmost redex is always reduced first.

$$
\begin{aligned}
& \underline{\mathtt{id}\ (\mathtt{id}\ (\lambda z.\ \mathtt{id}\ z))} \\
\longrightarrow\ & \underline{\mathtt{id}\ (\lambda z.\ \mathtt{id}\ z)} \\
\longrightarrow\ & \lambda z.\ \underline{\mathtt{id}\ z} \\
\longrightarrow\ & \lambda z.z \\
\not\longrightarrow\ &
\end{aligned}
$$

Evaluation Strategies

- The call-by-name strategy
  - A more restrictive normal order strategy, allowing no reduction inside abstraction.

$$\begin{array}{ll} & \underline{\mathtt{id}\ (\mathtt{id}\ (\lambda z.\ \mathtt{id}\ z))} \\ \longrightarrow & \underline{\mathtt{id}\ (\lambda z.\ \mathtt{id}\ z)} \\ \longrightarrow & \lambda z.\ \mathtt{id}\ z \\ \not\longrightarrow & \end{array}$$

Evaluation Strategies

- The call-by-value strategy
  - only outermost redexes are reduced and where a redex is reduced only when its right-hand side has already been reduced to a value, a term that cannot be reduced any more.

$$
\begin{array}{ll}
 & \mathrm{id}\ \underline{(\mathrm{id}\ (\lambda z.\ \mathrm{id}\ z))} \\
\longrightarrow & \underline{\mathrm{id}\ (\lambda z.\ \mathrm{id}\ z)} \\
\longrightarrow & \lambda z.\ \mathrm{id}\ z \\
\not\longrightarrow &
\end{array}
$$

Programming in the Lambda Calculus

Multiple Arguments

Church Booleans

Pairs

Church Numerals

Recursion

Multiple Arguments

$$f\ (x,\ y) = s$$

$\Downarrow$ currying

$$f\ x\ y = s$$

$\Downarrow$

$$f = \lambda x.\ \lambda y.\ s$$

Church Booleans

- Boolean values can be encodes as:

  tru = λt. λf. t;
  fls = λt. λf. f;

- Boolean conditional and operators can be encoded as:

  test = λl. λm. λn. l m n;
  and = λb. λc. b c fls;

Church Booleans

- An Example

$$
\begin{array}{ll}
 & \mathtt{test\ tru\ v\ w} \\
= & \underline{(\lambda l.\ \lambda m.\ \lambda n.\ l\ m\ n)\ \mathtt{tru}}\ v\ w \\
\longrightarrow & \underline{(\lambda m.\ \lambda n.\ \mathtt{tru}\ m\ n)\ v}\ w \\
\longrightarrow & \underline{(\lambda n.\ \mathtt{tru}\ v\ n)\ w} \\
\longrightarrow & \mathtt{tru}\ v\ w \\
= & \underline{(\lambda t.\lambda f.t)\ v}\ w \\
\longrightarrow & \underline{(\lambda f.\ v)\ w} \\
\longrightarrow & v
\end{array}
$$

Pairs

- Encoding

```
pair = λf.λs.λb. b f s;
fst = λp. p tru;
snd = λp. p fls;
```

- An Example

$$
\begin{aligned}
& \texttt{fst (pair v w)} \\
= \quad & \texttt{fst (}\underline{\texttt{(}\lambda\texttt{f. }\lambda\texttt{s. }\lambda\texttt{b. b f s) v}}\texttt{ w)} \\
\longrightarrow \quad & \texttt{fst (}\underline{\texttt{(}\lambda\texttt{s. }\lambda\texttt{b. b v s) w}}\texttt{)} \\
\longrightarrow \quad & \texttt{fst (}\lambda\texttt{b. b v w)} \\
= \quad & \underline{\texttt{(}\lambda\texttt{p. p tru) (}\lambda\texttt{b. b v w)}} \\
\longrightarrow \quad & \underline{\texttt{(}\lambda\texttt{b. b v w) tru}} \\
\longrightarrow \quad & \texttt{tru v w} \\
\longrightarrow^{*} \quad & \texttt{v}
\end{aligned}
$$

Church Numerals

- Encoding Church numerals:

$$
\begin{aligned}
c_0 &= \lambda s.\ \lambda z.\ z; \\
c_1 &= \lambda s.\ \lambda z.\ s\ z; \\
c_2 &= \lambda s.\ \lambda z.\ s\ (s\ z); \\
c_3 &= \lambda s.\ \lambda z.\ s\ (s\ (s\ z)); \\
&\text{etc.}
\end{aligned}
$$

- Defining functions on Church numerals:

$$
\begin{aligned}
\mathit{scc} &= \lambda n.\ \lambda s.\ \lambda z.\ s\ (n\ s\ z); \\
\mathit{plus} &= \lambda m.\ \lambda n.\ \lambda s.\ \lambda z.\ m\ s\ (n\ s\ z); \\
\mathit{times} &= \lambda m.\ \lambda n.\ m\ (\mathit{plus}\ n)\ c0;
\end{aligned}
$$

Recursion

- Terms with no normal form are said to diverge.

  omega = (λx. x x) (λx. x x);

- Fixed-point combinator

  fix = λf. (λx. f (λy. x x y)) (λx. f (λy. x x y));

  Note: fix f = f (fix f)

Recursion

- Basic Idea:

A recursive definition: h = <body containing h>

g = λ f . <body containing f>

h = fix g

Recursion

- Example:

```
fac = λn. if eq n c0
          then c1
          else times n (fac (pred n))
```

$\Downarrow$

```
g = λf . λn. if eq n c0
             then c1
             else times n (f (pred n))
fac = fix g
```

Exercise: Check that fac c3 → c6.

Formalities (Formal Definitions)

Syntax (free variables)

Substitution

Operational Semantics

Syntax

- **Definition** [Terms]: Let V be a countable set of variable names. The set of terms is the smallest set T such that

  1. $x \in T$ for every $x \in V$;
  2. if $t_1 \in T$ and $x \in V$, then $\lambda x.t_1 \in T$;
  3. If $t1 \in T$ and $t_2 \in T$, then $t_1\ t_2 \in T$.

- Free Variables

$$FV(x) = \{v\}$$
$$FV(\lambda x.t_1) = FV(t_1) \setminus \{x\}$$
$$FV(t_1\ t_2) = FV(t_1) \cup FV(t_2)$$

Substitution

$$
\begin{array}{llll}
[x \mapsto s]x & = & s & \\
[x \mapsto s]y & = & y & \text{if } y \neq x \\
[x \mapsto s](\lambda y.t_1) & = & \lambda y.\ [x \mapsto s]t_1 & \text{if } y \neq x \text{ and } y \notin FV(s) \\
[x \mapsto s](t_1\ t_2) & = & [x \mapsto s]t_1\ [x \mapsto s]t_2 &
\end{array}
$$

Example:

$$
\begin{array}{ll}
 & [x \to y\ z]\ (\lambda y.\ x\ y) \\
= & [x \to y\ z]\ (\lambda w.\ x\ w) \\
= & \lambda w.\ y\ z\ w
\end{array}
$$

Operational Semantics

Syntax

t	::=		*terms:*
		x	*variable*
		$\lambda x.t$	*abstraction*
		t t	*application*
v	::=		*values:*
		$\lambda x.t$	*abstraction value*

Evaluation $\boxed{t \longrightarrow t'}$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{\underline{v_1}\ t_2 \longrightarrow \underline{v_1}\ t_2'} \quad \text{(E-App2)}$$

$$(\lambda x.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \quad \text{(E-AppAbs)}$$

Summary

- What is lambda calculus for?
  - A core calculus for capturing language essential mechanisms
  - Simple but powerful
- Syntax
  - Function definition + function application
  - Binder, scope, free variables
- Operational semantics
  - Substitution
  - Evaluation strategies: normal order, call-by-name, call-by-value

Homework

- Understand Chapter 5.
- Do exercise 5.3.6 in Chapter 5.

Answers to Students' Questions

Q: What is the book I mentioned about Alan Turing?

A: Here is the book information.

The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine
Charles Petzold

出版社: Wiley; 1版 (2008/6/16)
言語: 英語, 英語, 英語
ISBN-10: 0470229055
ISBN-13: 978-0470229057
発売日： 2008/6/16