Untyped Lambda Calculus

Programming Languages CS442

David Toman

School of Computer Science
University of Waterloo

Syntax

Idea

We use only the structural parts of the language

$\Rightarrow$ *parameter abstraction and application*

$$T ::= x \mid (\lambda x.T) \mid (T_1\ T_2)$$

- $FV(T)$ free variables in T (not *bound* by λ)
- substitution $[T/x]$ in term T'

Reductions

Idea

Computation = substitution of actual parameters for formal parameters.

Computation Rules:

α-rule: $\lambda x.T \rightarrow \lambda y.[y/x]T$ for $y \notin FV(T)$

β-rule: $(\lambda x.T)\ T' \rightarrow [T'/x]T$

η-rule: $\lambda s.T\ x \rightarrow T$ for $x \notin FV(T)$

Reduction Sequences and Normal Forms

- (β-)redex $(\lambda x.T)\ T'$
- one-step: $E\{(\lambda x.T)\ T'\} \rightarrow E\{[T'/x]T\}$
- reduction sequence: $E \rightarrow E' \rightarrow E'' \rightarrow \cdots$
- normal form: E with no redexes

Coding Useful Things

- Booleans: $\mathit{true} = \lambda x.\lambda y.x \qquad \mathit{false} = \lambda x.\lambda y.y$

  $\Rightarrow \textbf{if}\ B\ \textbf{then}\ T_1\ \textbf{else}\ T_2\ \textbf{fi} = B\ T_1\ T_2$

- Pairs: $\langle T_1, T_2 \rangle = \lambda z.z\ T_1\ T_2$

  $\Rightarrow \textbf{proj}_1(T) = T\ \mathit{true}, \qquad \textbf{proj}_2(T) = T\ \mathit{false}$

- Natural Numbers (Church numerals): $n = \lambda s.\lambda z.s^n\ z$

  $\Rightarrow \textbf{succ} = \lambda n.\lambda s.\lambda z.s(n\ s\ z)$

  $\Rightarrow \textbf{add} = \lambda m.\lambda n.m\ \ \textbf{succ}\ \ n$

Loops?

Idea

Terms that produce their own copies as contractum.

- $\Omega = (\lambda x.x\ x)(\lambda x.x\ x)$

  $\Rightarrow \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \Omega \rightarrow \ldots$

- $Y = \lambda f.(\lambda x.f(x\ x))(\lambda x.f(x\ x))$

  $\Rightarrow Y\ f \rightarrow f(Y\ f) \rightarrow f(f(Y\ f)) \rightarrow \ldots$

Properties

Theorem (Confluence/CR)

For every $E \to^* E_1$ and $E \to^* E_2$ there is E' such that $E_1 \to^* E'$ and $E_2 \to^* E'$.

Theorem (Uniqueness of NF)

For every $E \to^* E'$, if E' is a normal form then E' is unique.

Rewriting Strategies

- *leftmost-outermost* (call-by-name)
- *call-by-value*

  $\Rightarrow$ values = identifiers (x) and abstractions ($\lambda x.T$)

  $\Rightarrow$ β-val: $(\lambda x.T)\ T' \rightarrow [T'/x]T$ if T' is a *value*.

Theorem (Standardization)

If E has a NF, then it is found by the leftmost-outermost rewriting.