

Lambda Calculus

KC Sivaramakrishnan

Spring 2020

IIT
MADRAS



MADRAS

Language

Variables	x	\in	Strings
Expressions	e	$::=$	$x \mid \lambda x. e \mid e e$

Free Variables

$$\begin{aligned} \text{FV}(x) &= \{x\} \\ \text{FV}(\lambda x. e) &= \text{FV}(e) - \{x\} \\ \text{FV}(e_1 e_2) &= \text{FV}(e_1) \cup \text{FV}(e_2) \end{aligned}$$

Substitution

$$\begin{aligned}[e'/x]x &= e' \\ [e'/x]y &= y, \text{ if } y \neq x \\ [e'/x]\lambda x. e &= \lambda x. e \\ [e'/x]\lambda y. e &= \lambda y. [e'/x]e, \text{ if } y \neq x \\ [e'/x]e_1 e_2 &= [e'/x]e_1 [e'/x]e_2\end{aligned}$$

- This definition is partial — *does not work with open-terms.*

$$[x/y]\lambda x. y = \lambda x. x$$

is incorrect

- For how to do it right, see CS3100 Lecture on Lambda Calculus syntax
- Our fix: we will *substitute only on closed terms*

OpSem: Big Step

$$\frac{}{\lambda x. e \Downarrow \lambda x. e} \quad \frac{e_1 \Downarrow \lambda x. e \quad e_2 \Downarrow v \quad [v/x]e \Downarrow v'}{e_1 e_2 \Downarrow v'}$$

Turing Completeness

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

THEOREM 10.1. *Ω does not evaluate to anything. In other words, $\Omega \Downarrow v$ implies a contradiction.*

Church Numerals

$$\begin{aligned}\text{zero} &= \lambda f. \lambda x. x \\ \text{plus1} &= \lambda n. \lambda f. \lambda x. f (n f x)\end{aligned}$$

- Let's show that the church numerals do encode natural numbers as we know them

- First, relate nats to church numerals

$$\begin{aligned}[0] &= x \\ [n + 1] &= f ((\lambda f. \lambda x. [n]) f x)\end{aligned}$$

- $n+1$ definition seems unnecessarily large.
 - That is what call-by-value (cbv) reduction produces.

Canonical Representation $\underline{n} = \lambda f. \lambda x. [n]$

Correctness of Encoding

- Given an encoding e and a natural number n , we say that e is a correct encoding of n if

$$e \sim n \equiv e \Downarrow \underline{n}$$

THEOREM 10.2. $\text{zero} \sim 0$.

THEOREM 10.3. *If $e_n \sim n$, then $\text{plus1 } e_n \sim n + 1$.*

$$\text{add} = \lambda n. \lambda m. n \text{ plus1 } m$$

THEOREM 10.4. *If $e_n \sim n$ and $e_m \sim m$, then $\text{add } e_n e_m \sim n + m$.*

$$\text{mult} = \lambda n. \lambda m. n (\text{add } m) \text{zero}$$

THEOREM 10.5. *If $e_n \sim n$ and $e_m \sim m$, then $\text{mult } e_n e_m \sim n \times m$.*

Small-step semantics

Evaluation contexts $C ::= \square \mid C e \mid v C$

$$\overline{C[(\lambda x. e) v] \rightarrow C[[v/x]e]}$$

THEOREM 10.6. *If $e \rightarrow^* v$, then $e \Downarrow v$.*

THEOREM 10.7. *If $e \Downarrow v$, then $e \rightarrow^* v$.*

Simply Typed Lambda Calculus

Types $\tau ::= \mathbb{N} \mid \tau \rightarrow \tau$

Variables $x \in \text{Strings}$

Numbers $n \in \mathbb{N}$

Expressions $e ::= n \mid e + e \mid x \mid \lambda x. e \mid e e$

Values $v ::= n \mid \lambda x. e$

STLC: Dynamic Semantics

Evaluation contexts $C ::= \square \mid C e \mid v C \mid C + e \mid v + C$

$$\overline{(\lambda x. e) v \rightarrow_0 [v/x]e} \quad \overline{n + m \rightarrow_0 n+m}$$

$$\frac{e \rightarrow_0 e'}{C[e] \rightarrow C[e']}$$

STLC: Static Semantics

$$\begin{array}{c} \frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \quad \frac{}{\Gamma \vdash n : \mathbb{N}} \quad \frac{\Gamma \vdash e_1 : \mathbb{N} \quad \Gamma \vdash e_2 : \mathbb{N}}{\Gamma \vdash e_1 + e_2 : \mathbb{N}} \\[1em] \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x. e : \tau_1 \rightarrow \tau_2} \quad \frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2} \end{array}$$

Type Soundness

- **Well-typed programs do not get stuck**

- ✦ stuck: not a value, but cannot reduce further

THEOREM 10.14 (Type Soundness). *If $\vdash e : \tau$, then $\neg \text{stuck}$ is an invariant of $\mathbb{T}(e)$.*

- **Progress:** If lambda expression **e** has type **t**, then **e** isn't stuck.

LEMMA 10.8 (Progress). *If $\vdash e : \tau$, then e isn't stuck.*

- **Preservation:** If expression **e** has type **t**, and **e** \longrightarrow **e'** then **e'** has type **t**

LEMMA 10.13 (Preservation). *If $e_1 \rightarrow e_2$ and $\vdash e_1 : \tau$, then $\vdash e_2 : \tau$.*

Type Soundness: Other Lemmas

LEMMA 10.9 (Weakening). *If $\Gamma \vdash e : \tau$ and every mapping in Γ is also included in Γ' , then $\Gamma' \vdash e : \tau$.*

PROOF. By induction on the derivation of $\Gamma \vdash e : \tau$. □

LEMMA 10.10 (Substitution). *If $\Gamma, x : \tau' \vdash e : \tau$ and $\vdash e' : \tau'$, then $\Gamma \vdash [e'/x]e : \tau$.*

PROOF. By induction on the derivation of $\Gamma, x : \tau' \vdash e : \tau$, with appeal to Lemma 10.9. □

LEMMA 10.11. *If $e \rightarrow_0 e'$ and $\vdash e : \tau$, then $\vdash e' : \tau$.*

PROOF. By inversion on the derivation of $e \rightarrow_0 e'$, with appeal to Lemma 10.10. □

LEMMA 10.12. *If any type of e_1 is also a type of e_2 , then any type of $C[e_1]$ is also a type of $C[e_2]$.*

PROOF. By induction on the structure of C . □