Lambda Calculus & Type Soundness

KC Sivaramakrishnan

Spring 2025

Language

$$
\begin{array}{rcl}
\text{Variables} & x & \in & \mathsf{Strings} \\
\text{Expressions} & e & ::= & x \mid \lambda x.\ e \mid e\ e
\end{array}
$$

Free Variables

$$\begin{array}{rcl} \mathsf{FV}(x) & = & \{x\} \\ \mathsf{FV}(\lambda x.\ e) & = & \mathsf{FV}(e) - \{x\} \\ \mathsf{FV}(e_1\ e_2) & = & \mathsf{FV}(e_1) \cup \mathsf{FV}(e_2) \end{array}$$

Substitution

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

- 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

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

Turing Completeness

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

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

Church Numerals

$$
\begin{array}{rcl}
\textsf{zero} & = & \lambda f.\ \lambda x.\ x \\
\textsf{plus1} & = & \lambda n.\ \lambda f.\ \lambda x.\ f\ (n\ f\ x)
\end{array}
$$

- Let's show that the church numerals do encode natural numbers as we know them
- First, relate nats to church numerals

$$
\begin{array}{rcl}
\lfloor 0 \rfloor & = & x \\
\lfloor n+1 \rfloor & = & f\ ((\lambda f.\ \lambda x.\ \lfloor n \rfloor)\ f\ x)
\end{array}
$$

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

Canonical Representation $\underline{n} = \lambda f.\ \lambda x.\ \lfloor n \rfloor$

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. $\mathsf{zero} \sim 0$.

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

$$\mathsf{add} \quad = \quad \lambda n.\ \lambda m.\ n\ \mathsf{plus1}\ m$$

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

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

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

Small-step semantics

Key Reductions

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

Administrative Reductions

$$\frac{e_1 \rightarrow e_1'}{e_1\ e_2 \rightarrow e_1'\ e_2} \qquad \frac{e_2 \rightarrow e_2'}{v\ e_2 \rightarrow v\ e_2'}$$

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

$$\text{Types} \quad \tau \quad ::= \quad \mathbb{N} \mid \tau \to \tau$$

$$\begin{array}{rcll}
\text{Variables} & x & \in & \mathsf{Strings} \\
\text{Numbers} & n & \in & \mathbb{N} \\
\text{Expressions} & e & ::= & n \mid e + e \mid x \mid \lambda x.\ e \mid e\ e \\
\text{Values} & v & ::= & n \mid \lambda x.\ e
\end{array}$$

STLC: Dynamic Semantics

Key Reductions

$$\frac{}{(\lambda x.\ e)\ v \rightarrow [v/x]e} \qquad \frac{}{n + m \rightarrow n{+}m}$$

Administrative Reductions

$$\frac{e_1 \rightarrow e_1'}{e_1\ e_2 \rightarrow e_1'\ e_2} \qquad \frac{e_2 \rightarrow e_2'}{v\ e_2 \rightarrow v\ e_2'}$$

$$\frac{e_1 \rightarrow e_1'}{e_1 + e_2 \rightarrow e_1' + e_2} \qquad \frac{e_2 \rightarrow e_2'}{v + e_2 \rightarrow v + e_2'}$$

STLC: Static Semantics

Types $\tau \quad ::= \quad \mathbb{N} \mid \tau \to \tau$

Typing Judgement $\Gamma \vdash x : T$

Typing context

Typing Rules

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau} \qquad \frac{}{\Gamma \vdash n : \mathbb{N}} \qquad \frac{\Gamma \vdash e_1 : \mathbb{N} \quad \Gamma \vdash e_2 : \mathbb{N}}{\Gamma \vdash e_1 + e_2 : \mathbb{N}}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x.\, e : \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1\ e_2 : \tau_2}$$

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* $\neg$*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** —> **e'** then **e'** has type **t**

LEMMA 10.13 (Preservation). *If* $e_1 \to e_2$ *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. □