# Simply-typed Lambda Calculus

Todd Millstein

November 5, 2001

This document formally defines the call-by-value simply-typed lambda calculus (with booleans) and provides a proof of type soundness. It is meant only as a reference, and assumes familiarity with the basic notions involved.

# 1 Syntax

The metavariable $x$ ranges over an infinite set of variable names. The metavariable $e$ ranges over expressions (terms). The metavariable $T$ ranges over types. The metavariable $v$ ranges over values.

$$
\begin{array}{lcl}
e & ::= & x \mid \lambda x : T.e \mid e_1\ e_2 \\
 & & \text{true} \mid \text{false} \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3 \\
T & ::= & \text{Bool} \mid T_1 \to T_2 \\
v & ::= & \lambda x : T.e \mid \text{true} \mid \text{false}
\end{array}
$$

# 2 Operational Semantics

## 2.1 Substitution

The substitution function is defined below. We assume that renaming of bound variables is applied as necessary to make the side conditions of the third case hold.

$$
\begin{array}{lcll}
[x \mapsto e]x & = & e & \\
[x \mapsto e]x' & = & x' & \text{if } x \neq x' \\
[x \mapsto e](\lambda x' : T'.e') & = & \lambda x' : T'.[x \mapsto e]e' & \text{if } x \neq x' \text{ and } x' \text{ not free in } e \\
[x \mapsto e](e_1\ e_2) & = & [x \mapsto e]e_1\ [x \mapsto e]e_2 & \\
[x \mapsto e]\text{true} & = & \text{true} & \\
[x \mapsto e]\text{false} & = & \text{false} & \\
[x \mapsto e]\text{if } e_1 \text{ then } e_2 \text{ else } e_3 & = & \text{if } [x \mapsto e]e_1 \text{ then } [x \mapsto e]e_2 \text{ else } [x \mapsto e]e_3 &
\end{array}
$$

## 2.2 Inference Rules

The notation $e \longrightarrow e'$ means "expression $e$ evaluates to $e'$ in one step."

$$\frac{}{(\lambda x : T.e)v \longrightarrow [x \mapsto v]e} \text{ (E-AppRed)}$$

$$\frac{}{\text{if true then } e_2 \text{ else } e_3 \longrightarrow e_2} \text{ (E-IfTrue)}$$

$$\frac{e_1 \longrightarrow e_1'}{e_1\ e_2 \longrightarrow e_1'\ e_2} \text{ (E-App1)}$$

$$\frac{}{\text{if false then } e_2 \text{ else } e_3 \longrightarrow e_3} \text{ (E-IfFalse)}$$

$$\frac{e \longrightarrow e'}{v\ e \longrightarrow v\ e'} \text{ (E-App2)}$$

$$\frac{e_1 \longrightarrow e_1'}{\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \longrightarrow \text{if } e_1' \text{ then } e_2 \text{ else } e_3} \text{ (E-If)}$$

### 2.3 Stuck Expressions

An expression $e$ is *stuck* if it is not a value but there is no $e'$ such that $e \longrightarrow e'$. The stuck expressions can be thought of as the set of possible run-time "type" errors. The grammar of stuck expressions is as follows:

stuck ::= $x$
stuck $e$ | true $v$ | false $v$
$v$ stuck
if stuck then $e_2$ else $e_3$
if $\lambda x : T.e$ then $e_2$ else $e_3$

# 3 Typechecking Rules

The metavariable $\Gamma$ represents a *type environment*, which is a set of (variable name, type) pairs. Each pair with variable name $x$ and type $T$ is denoted $x : T$. We assume that a type environment has at most one pair for a given variable name; this can always be ensured via renaming of bound variables. If $\Gamma = \{x_1 : T_1, \ldots, x_n : T_n\}$, then we define dom($\Gamma$) = $\{x_1, \ldots, x_n\}$.

A judgement of the form $\Gamma \vdash e : T$ means "expression $e$ has type $T$ under the typing assumptions in $\Gamma$." If the $\Gamma$ component is missing from a judgement, the type environment is assumed to be the empty set.

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \text{ (T-Var)}$$

$$\frac{}{\Gamma \vdash \text{true} : \text{Bool}} \text{ (T-True)}$$

$$\frac{\Gamma \cup \{x : T_1\} \vdash e : T_2}{\Gamma \vdash (\lambda x : T_1.e) : T_1 \to T_2} \text{ (T-Abs)}$$

$$\frac{}{\Gamma \vdash \text{false} : \text{Bool}} \text{ (T-False)}$$

$$\frac{\Gamma \vdash e_1 : T_2 \to T \quad \Gamma \vdash e_2 : T_2}{\Gamma \vdash e_1\ e_2 : T} \text{ (T-App)}$$

$$\frac{\Gamma \vdash e_1 : \text{Bool} \quad \Gamma \vdash e_2 : T \quad \Gamma \vdash e_3 : T}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : T} \text{ (T-If)}$$

# 4 Type Soundness

**Lemma** (Canonical Forms):

a. If $\Gamma \vdash v : T_1 \to T_2$ then $v$ has the form $\lambda x : T_1.e$.

b. If $\Gamma \vdash v$ : Bool then $v$ is either true or false.

**Proof**: Immediate from rules T-Abs, T-True, and T-False, and the fact that no other typing rules apply to values.

**Theorem** (Progress): If $\vdash e : T$, then either $e$ is a value or there exists $e'$ such that $e \longrightarrow e'$ (equivalently, If $\vdash e : T$, then $e$ is not stuck).
**Proof**: By (strong) induction on the depth of the derivation of $\vdash e : T$. Case analysis of the last rule in the derivation:

- Case T-Var: Then $e = x$ and $x : T \in \emptyset$, so we have a contradiction. Therefore, T-Var cannot be the last rule in the derivation.
- Case T-Abs: Then $e = \lambda x : T_1.e_1$, so $e$ is a value.
- Case T-App: Then $e = e_1\ e_2$ and $\vdash e_1 : T_2 \to T$ and $\vdash e_2 : T_2$. By the inductive hypothesis, we have that either $e_1$ is a value or there exists $e_1'$ such that $e_1 \longrightarrow e_1'$. Similarly, either $e_2$ is a value or there exists $e_2'$ such that $e_2 \longrightarrow e_2'$. We perform a case analysis on these possibilities:

- Case there exists $e_1'$ such that $e_1 \longrightarrow e_1'$: Then by E-App1 we have $e_1\ e_2 \longrightarrow e_1'\ e_2$.
- Case $e_1$ is a value $v_1$: There are two sub-cases.
  * Case there exists $e_2'$ such that $e_2 \longrightarrow e_2'$: Then by E-App2 we have $v_1\ e_2 \longrightarrow v_1\ e_2'$.
  * Case $e_2$ is a value $v_2$: Since $\vdash e_1 : T_2 \to T$ and $e_1$ is a value $v_1$, by the Canonical Forms lemma we have that $e_1$ has the form $\lambda x : T'.e_3$. Therefore by E-AppRed we have $(\lambda x : T'.e_3)v_2 \longrightarrow [x \mapsto v_2]e_3$.

- Case T-True: Then $e =$ true, so $e$ is a value.
- Case T-False: Then $e =$ false, so $e$ is a value.
- Case T-If: Then $e =$ if $e_1$ then $e_2$ else $e_3$ and $\vdash e_1 :$ Bool and $\vdash e_2 : T$ and $\vdash e_3 : T$. By the inductive hypothesis, we have that either $e_1$ is a value, or there exists $e_1'$ such that $e_1 \longrightarrow e_1'$. In the latter case, by E-If we have that if $e_1$ then $e_2$ else $e_3 \longrightarrow$ if $e_1'$ then $e_2$ else $e_3$. In the former case, by the Canonical Forms lemma we have that $e_1$ is either true or false. If $e_1$ is true, then by E-IfTrue we have that if $e_1$ then $e_2$ else $e_3 \longrightarrow e_2$. If $e_1$ is false, then by E-IfFalse we have that if $e_1$ then $e_2$ else $e_3 \longrightarrow e_3$.

**Lemma** (Weakening): If $\Gamma \vdash e : T$ and $x_0 \notin \text{dom}(\Gamma)$, then $\Gamma \cup \{x_0 : T_0\} \vdash e : T$.
**Proof**: By (strong) induction on the depth of the derivation of $\Gamma \vdash e : T$. Case analysis of the last rule in the derivation:

- Case T-Var: Then $e = x$ and $x : T \in \Gamma$. Since $x_0 \notin \text{dom}(\Gamma)$, we have that $x_0 \neq x$. Therefore $x : T \in \Gamma \cup \{x_0 : T_0\}$, so by T-Var we have $\Gamma \cup \{x_0 : T_0\} \vdash x : T$.
- Case T-Abs: Then $e = \lambda x_1 : T_1.e_2$ and $T = T_1 \to T_2$ and $\Gamma \cup \{x_1 : T_1\} \vdash e_2 : T_2$. We assume that $x_1 \neq x_0$, renaming $x_1$ if necessary. Since $x_0 \notin \text{dom}(\Gamma)$, also $x_0 \notin \text{dom}(\Gamma \cup \{x_1 : T_1\})$. Therefore by the inductive hypothesis we have $\Gamma \cup \{x_1 : T_1\} \cup \{x_0 : T_0\} \vdash e_2 : T_2$. So by T-Abs we have $\Gamma \cup \{x_0 : T_0\} \vdash (\lambda x_1 : T_1.e_2) : T_1 \to T_2$.
- Case T-App: Then $e = e_1\ e_2$ and $\Gamma \vdash e_1 : T_2 \to T$ and $\Gamma \vdash e_2 : T_2$. By the inductive hypothesis we have $\Gamma \cup \{x_0 : T_0\} \vdash e_1 : T_2 \to T$ and $\Gamma \cup \{x_0 : T_0\} \vdash e_2 : T_2$, so by T-App we have $\Gamma \cup \{x_0 : T_0\} \vdash e_1\ e_2 : T$.
- Case T-True: Then $e =$ true and $T =$ Bool. Therefore by T-True we have $\Gamma \cup \{x_0 : T_0\} \vdash$ true : Bool.
- Case T-False: Then $e =$ false and $T =$ Bool. Therefore by T-False we have $\Gamma \cup \{x_0 : T_0\} \vdash$ false : Bool.
- Case T-If: Then $e =$ if $e_1$ then $e_2$ else $e_3$ and $\Gamma \vdash e_1 :$ Bool and $\Gamma \vdash e_2 : T$ and $\Gamma \vdash e_3 : T$. By the inductive hypothesis we have $\Gamma \cup \{x_0 : T_0\} \vdash e_1 :$ Bool and $\Gamma \cup \{x_0 : T_0\} \vdash e_2 : T$ and $\Gamma \cup \{x_0 : T_0\} \vdash e_3 : T$, so by T-If we have $\Gamma \cup \{x_0 : T_0\} \vdash$ if $e_1$ then $e_2$ else $e_3 : T$.

**Lemma** (Substitution): If $\Gamma \cup \{x : T\} \vdash e' : T'$ and $\Gamma \vdash v : T$, then $\Gamma \vdash [x \mapsto v]e' : T'$.
**Proof**: By (strong) induction on the depth of the derivation of $\Gamma \cup \{x : T\} \vdash e' : T'$. Case analysis of the last rule in the derivation:

- Case T-Var: Then $e' = x'$ and $x' : T' \in \Gamma \cup \{x : T\}$. There are two subcases:
  - Case $x' = x$: Then $[x \mapsto v]e' = [x \mapsto v]x = v$. Since we assume that $\Gamma \cup \{x : T\}$ has at most one element for each variable name, we have that $T' = T$. Finally, since $\Gamma \vdash v : T$, this case is proven.
  - Case $x' \neq x$: Then $[x \mapsto v]e' = x'$. Since $x' : T' \in \Gamma \cup \{x : T\}$ and $x' \neq x$, we have $x' : T' \in \Gamma$. Therefore by T-Var we have $\Gamma \vdash x' : T'$.

- Case T-Abs: Then $e' = \lambda x_0 : T_0.e_1$ and $T' = T_0 \to T_1$ and $\Gamma \cup \{x : T\} \cup \{x_0 : T_0\} \vdash e_1 : T_1$. Since $\Gamma \vdash v : T$, by Weakening (renaming $x_0$ if necessary) we have $\Gamma \cup \{x_0 : T_0\} \vdash v : T$, so by the inductive hypothesis we have $\Gamma \cup \{x_0 : T_0\} \vdash [x \mapsto v]e_1 : T_1$. Therefore by T-Abs we have $\Gamma \vdash \lambda x_0 : T_0.[x \mapsto v]e_1 : T_0 \to T_1$. Since we can assume that $x \neq x_0$ and $x_0$ not free in $v$, performing renaming as necessary, we have $[x \mapsto v]e' = \lambda x_0 : T_0.[x \mapsto v]e_1$, so the result follows.
- Case T-App: Then $e' = e_1\ e_2$ and $\Gamma \cup \{x : T\} \vdash e_1 : T_2 \to T'$ and $\Gamma \cup \{x : T\} \vdash e_2 : T_2$. Then by the inductive hypothesis we have $\Gamma \vdash [x \mapsto v]e_1 : T_2 \to T'$ and $\Gamma \vdash [x \mapsto v]e_2 : T_2$, so by T-App we have $\Gamma \vdash [x \mapsto v]e_1\ [x \mapsto v]e_2 : T'$. Since $[x \mapsto v](e_1\ e_2) = [x \mapsto v]e_1\ [x \mapsto v]e_2$, the result follows.
- Case T-True: Then $e' =$ true and $T' =$ Bool. Then by T-True we have $\Gamma \vdash$ true : Bool. Since $[x \mapsto v]$true = true, the result follows.
- Case T-False: Then $e' =$ false and $T' =$ Bool. Then by T-False we have $\Gamma \vdash$ false : Bool. Since $[x \mapsto v]$false = false, the result follows.
- Case T-If: Then $e' =$ if $e_1$ then $e_2$ else $e_3$ and $\Gamma \cup \{x : T\} \vdash e_1$ : Bool and $\Gamma \cup \{x : T\} \vdash e_2 : T'$ and $\Gamma \cup \{x : T\} \vdash e_3 : T'$. By the inductive hypothesis we have $\Gamma \vdash [x \mapsto v]e_1$ : Bool and $\Gamma \vdash [x \mapsto v]e_2 : T'$ and $\Gamma \vdash [x \mapsto v]e_3 : T'$. Since $[x \mapsto v]$if $e_1$ then $e_2$ else $e_3 =$ if $[x \mapsto v]e_1$ then $[x \mapsto v]e_2$ else $[x \mapsto v]e_3$, the result follows.

**Theorem** (Type Preservation): If $\Gamma \vdash e : T$ and $e \longrightarrow e'$, then $\Gamma \vdash e' : T$.
**Proof**: By (strong) induction on the depth of the derivation of $\Gamma \vdash e : T$. Case analysis of the last rule in the derivation:

- Case T-Var: Then $e = x$. By inspection of the operational semantics, there is no $e'$ such that $x \longrightarrow e'$, so this case is satisfied trivially.
- Case T-Abs: Similar to the previous case.
- Case T-App: Then $e = e_1\ e_2$ and $\Gamma \vdash e_1 : T_2 \to T$ and $\Gamma \vdash e_2 : T_2$. We're given that $e \longrightarrow e'$. Case analysis of the last rule used in the derivation of this reduction step:
  - Case E-App1: Then $e' = e_1'\ e_2$ and $e_1 \longrightarrow e_1'$. By the inductive hypothesis we have that $\Gamma \vdash e_1' : T_2 \to T$. Therefore, by T-App we have $\Gamma \vdash e_1'\ e_2 : T$.
  - Case E-App2: Then $e' = e_1\ e_2'$ and $e_2 \longrightarrow e_2'$. By the inductive hypothesis we have that $\Gamma \vdash e_2' : T_2$. Therefore, by T-App we have $\Gamma \vdash e_1\ e_2' : T$.
  - Case E-AppRed: Then $e_1 = \lambda x : T_1.e_3$ and $e_2 = v$ and $e' = [x \mapsto v]e_3$. Since $\Gamma \vdash e_1 : T_2 \to T$ and $e_1$ is a value, by the Canonical Forms lemma we have that $T_1 = T_2$, so we have $\Gamma \vdash \lambda x : T_2.e_3 : T_2 \to T$. By inspection, this derivation must end with rule T-Abs. Therefore we have that $\Gamma \cup \{x : T_2\} \vdash e_3 : T$. Since $\Gamma \vdash e_2 : T_2$ and $e_2 = v$ we have $\Gamma \vdash v : T_2$. Therefore by the Substitution lemma we have $\Gamma \vdash [x \mapsto v]e_3 : T$.
- Case T-True: Then $e =$ true. By inspection, there is no $e'$ such that true $\longrightarrow e'$, so this case is satisfied trivially.
- Case T-False: Similar to the previous case.
- Case T-If: Then $e =$ (if $e_1$ then $e_2$ else $e_3$) and $\Gamma \vdash e_1$ : Bool and $\Gamma \vdash e_2 : T$ and $\Gamma \vdash e_3 : T$. We're given that $e \longrightarrow e'$. Case analysis of the last rule used in the derivation of this reduction step:
  - Case E-IfTrue: Then $e' = e_2$, so we have $\Gamma \vdash e' : T$.
  - Case E-IfFalse: Then $e' = e_3$, so we have $\Gamma \vdash e' : T$.

- Case E-If: Then if $e_1$ then $e_2$ else $e_3 \longrightarrow$ if $e_1'$ then $e_2$ else $e_3$, where $e_1 \longrightarrow e_1'$. By the inductive hypothesis we have $\Gamma \vdash e_1'$ : Bool. Therefore by T-If we have $\Gamma \vdash$ if $e_1'$ then $e_2$ else $e_3 : T$.

**Theorem** (Type Soundness #1): If $\vdash e : T$ then either $e$ is a value or there exists $e'$ such that $e \longrightarrow e'$ and $\vdash e' : T$.
**Proof**: Since $\vdash e : T$, by Progress either $e$ is a value or there exists $e'$ such that $e \longrightarrow e'$. In the latter case, by Type Preservation we have $\vdash e' : T$.

Let $\stackrel{*}{\longrightarrow}$ denote the reflexive, transitive closure of the $\longrightarrow$ relation.
**Corollary** (Type Soundness #2): If $\vdash e : T$ and the evaluation of $e$ terminates, then there exists $v$ such that $e \stackrel{*}{\longrightarrow} v$ and $\vdash v : T$.
**Proof**: Since $\vdash e : T$, by Type Soundness #1 we have that either $e$ is a value or there exists $e'$ such that $e \longrightarrow e'$ and $\vdash e' : T$. Since the evaluation of $e$ terminates, some evaluation of $e$ has finite length (number of reduction steps). We prove this corollary by induction on the length of this evaluation of $e$.

- Case length = 0: Then there does not exist $e'$ such that $e \longrightarrow e'$, so $e$ must be a value. Therefore, this case is proven by taking $v = e$.
- Case length = $n$, where $n > 0$: Then there is at least one reduction step in the evaluation, so $e$ is not a value. Therefore there exists $e'$ such that $e \longrightarrow e'$ and $\vdash e' : T$. Since the evaluation of $e$ terminates, so does the evaluation of $e'$. Further, the evaluation of $e'$ has length $n - 1$. Therefore, by the inductive hypothesis we have that there exists $v$ such that $e' \stackrel{*}{\longrightarrow} v$ and $\vdash v : T$. Since $e \longrightarrow e'$ and $e' \stackrel{*}{\longrightarrow} v$, we have $e \stackrel{*}{\longrightarrow} v$.