Simply-Typed Lambda Calculus (STLC) with extensions

Ulrik Unneberg

1 Introduction

This report presents the design and implementation of the **Simply-Typed Lambda Calculus (STLC)** with extension to support for

- Function types
- Conjunctions
- Disjunctions
- Primitive types such as integers and booleans
- Binary operation such as addition and multiplication and logical operations
- Conditional expressions
- Pairs and projections
- Injections and cases
- Simple proofs of propositional logic using the Curry-Howard Correspondence

This report explains the syntax, type rules, and evaluation rules of the STLC, and is meant to be a reference for the implementation of the STLC with extensions. Details of examples are found in the test notebook in the repository.

2 Syntax

We have the expressions in the STLC with as

$$\begin{aligned} t ::= {} & x \mid \lambda x : T.t \mid t_1\ t_2 \mid \text{if } t_1 \text{ then } t_2 \text{ else } t_3 \\ & \mid \langle t_1, t_2 \rangle \mid \text{fst}(t) \mid \text{snd}(t) \\ & \mid \text{inl}_{A \vee B}(t) \mid \text{inr}_{A \vee B}(t) \mid \text{case } t \text{ of } (\text{inl}_{A \vee B}(x) \Rightarrow t_1 \mid \text{inr}_{A \vee B}(y) \Rightarrow t_2), \end{aligned}$$

and the types as

$$T ::= \text{Int} \mid \text{Bool} \mid T_1 \to T_2 \mid T_1 \wedge T_2 \mid T_1 \vee T_2$$

3 Type System

The type system of the STLC makes sure that all expressions are well-typed, and is presented in the following.

3.1 Function Types

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x : A.t : A \to B}\ \textsc{Abs}$$

$$\frac{\Gamma \vdash t_1 : A \to B \quad \Gamma \vdash t_2 : A}{\Gamma \vdash t_1\ t_2 : B}\ \textsc{App}$$

3.2 Conjunction Types

$$\frac{\Gamma \vdash t_1 : A \quad \Gamma \vdash t_2 : B}{\Gamma \vdash \langle t_1, t_2 \rangle : A \wedge B}\ \textsc{Pair}$$

$$\frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash \mathrm{fst}(t) : A}\ \textsc{Fst}$$

$$\frac{\Gamma \vdash t : A \wedge B}{\Gamma \vdash \mathrm{snd}(t) : B}\ \textsc{Snd}$$

3.3 Disjunction Types

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \mathrm{inl}_{A \vee B}(t) : A \vee B}\ \textsc{Inl}$$

$$\frac{\Gamma \vdash t : B}{\Gamma \vdash \mathrm{inr}_{A \vee B}(t) : A \vee B}\ \textsc{Inr}$$

$$\frac{\Gamma \vdash t : A \vee B \quad \Gamma, x : A \vdash t_1 : C \quad \Gamma, y : B \vdash t_2 : C}{\Gamma \vdash \text{case } t \text{ of } (\mathrm{inl}_{A \vee B}(x) \Rightarrow t_1 \mid \mathrm{inr}_{A \vee B}(y) \Rightarrow t_2) : C}\ \textsc{Case}$$

4 Reduction rules

In these rules, we use the symbol $\rightsquigarrow$ to denote evaluation. We distinguish between evaluation of terms (which can be expressions like applications, pairs, etc.) and values (which are fully evaluated results). For terms, the rules are evaluation of sub-expressions, while for values, the rules are about substituting and reducing the function body.

4.1 Computation Rules

$$\frac{}{\mathrm{App}(\lambda x : A.t_1)\ t_2 \rightsquigarrow t_1[t_2/x]}\ \text{Eval-App}$$

$$\frac{}{\mathrm{fst}(\langle v_1, v_2\rangle) \rightsquigarrow v_1}\ \text{Eval-Fst}$$

$$\frac{}{\mathrm{snd}(\langle v_1, v_2\rangle) \rightsquigarrow v_2}\ \text{Eval-Snd}$$

$$\frac{e \rightsquigarrow e'}{\mathrm{Inl}(e) \to \mathrm{Inl}(e')}\ \text{Eval-Inl}$$

$$\frac{e \rightsquigarrow e'}{\mathrm{Inr}(e) \to \mathrm{Inr}(e')}\ \text{Eval-Inr}$$

$$\frac{}{\text{case }\mathrm{Inl}(v)\text{ of }(\mathrm{Inl}(x) \Rightarrow t_1 \mid \mathrm{Inr}(y) \Rightarrow t_2) \rightsquigarrow t_1[v/x]}\ \text{Eval-Case-Inl}$$

$$\frac{}{\text{case }\mathrm{Inr}(v)\text{ of }(\mathrm{Inl}(x) \Rightarrow t_1 \mid \mathrm{Inr}(y) \Rightarrow t_2) \rightsquigarrow t_2[v/y]}\ \text{Eval-Case-Inr}$$

$$\frac{}{\text{if true then } t_1 \text{ else } t_2 \rightsquigarrow t_1}\ \text{Eval-If-True}$$

$$\frac{}{\text{if false then } t_1 \text{ else } t_2 \rightsquigarrow t_2}\ \text{Eval-If-False}$$

4.2 Congruence Rules

$$\frac{e_1 \rightsquigarrow e_1'}{\mathrm{App}(e_1, e_2) \rightsquigarrow \mathrm{App}(e_1', e_2)}\ \text{Eval-Rator}$$

$$\frac{e_2 \rightsquigarrow e_2'}{\mathrm{App}(v_1, e_2) \rightsquigarrow \mathrm{App}(v_1, e_2')}\ \text{Eval-Rand}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\mathrm{fst}(e_1) \rightsquigarrow \mathrm{fst}(e_1')}\ \text{Cong-Fst}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\mathrm{snd}(e_1) \rightsquigarrow \mathrm{snd}(e_1')}\ \text{Cong-Snd}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\text{case } e_1 \text{ of }(\mathrm{Inl}(x) \Rightarrow t_1 \mid \mathrm{Inr}(y) \Rightarrow t_2) \rightsquigarrow \text{case } e_1' \text{ of }(\mathrm{Inl}(x) \Rightarrow t_1 \mid \mathrm{Inr}(y) \Rightarrow t_2)}\ \text{Cong-Case}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\langle e_1, e_2\rangle \rightsquigarrow \langle e_1', e_2\rangle}\ \text{Cong-Pair-Left}$$

$$\frac{e_2 \rightsquigarrow e_2'}{\langle v_1, e_2\rangle \rightsquigarrow \langle v_1, e_2'\rangle}\ \text{Cong-Pair-Right}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\text{if } e_1 \text{ then } e_2 \text{ else } e_3 \rightsquigarrow \text{if } e_1' \text{ then } e_2 \text{ else } e_3}\ \text{Cong-If}$$

$$\frac{e_1 \rightsquigarrow e_2}{\lambda(x:\tau_1).e_1 \rightsquigarrow \lambda(x:\tau_1).e_2}\ \text{Eval-}\lambda\text{-Body}$$

5 Examples (also given in test notebook in repo)

In the notebook, we will demonstrate that binary operations such as addition and multiplication, and logical operations such as conjunction and disjunction, and conditional (if) statements, can be implemented using our STLC with extensions. We will also show how to implement conditional expressions, pairs and projections, injections and cases, and simple proofs of propositional logic using the Curry-Howard Correspondence. Some proof we will go through is the following

5.1 Proof: $(A \wedge B) \to A$

$$\lambda p : A \wedge B.\ \mathrm{fst}(p)$$

5.2 Proof: $A \to (A \vee B)$

$$\lambda a : A.\ \mathrm{inl}_{A\vee B}(a)$$

5.3 Proof: $(A \to C) \wedge (B \to C) \to (A \vee B \to C)$

$\lambda f : (A \to C) \wedge (B \to C).\ \lambda x : A \vee B.\ \text{case } x \text{ of } (\mathrm{inl}_{A\vee B}(a) \Rightarrow \mathrm{fst}(f)(a) \mid \mathrm{inr}_{A\vee B}(b) \Rightarrow \mathrm{snd}(f)(b))$