Simply Implemented System F1

Aoyang Yu

1 Introduction

System F1 is the type system of *first-order typed λ-calculus*. By first-order, we mean that variables can only represent terms of types generated by a set of *basic types*. They cannot represent types, which are terms of universes.

The author was first exposed to F1 when reading [1], an introductory article on type systems. He was excited to find that he can implement this system rather easily on a computer and make it really works. This is the document of the simply implemented system F1.

2 Specification

The specification of the system F1 can be found in [1], but it is restated here for completeness.

The syntax of F1 can be described as λ-calculus with type annotations, and can be formally specified by the following rules.

$$\begin{aligned} \langle \text{Type} \rangle &\models \texttt{BasicType} \mid \langle \text{Type} \rangle \to \langle \text{Type} \rangle \\ \langle \text{Term} \rangle &\models \texttt{Variable} \mid \lambda.\texttt{Variable} : \langle \text{Type} \rangle.\langle \text{Term} \rangle \mid \langle \text{Term} \rangle \ \langle \text{Term} \rangle \end{aligned}$$

There are three kinds of judgements in F1.

$\Gamma \vdash \diamond$	Γ is a well-formed environment
$\Gamma \vdash A$	A is a well-formed type in Γ
$\Gamma \vdash x : A$	x is a well-formed term of type A under Γ

The type system is a collection of five type rules which can be divided into three groups, deriving three kinds of judgements respectively.

(Env-Empty)

$$\frac{}{\emptyset \vdash \diamond}$$

(Env-x)

$$\frac{\Gamma \vdash A \qquad x \notin \operatorname{dom} \Gamma}{\Gamma \vdash x : A}$$

(Type-Const)

$$\frac{\Gamma \vdash \diamond \qquad K \in \text{BasicTypes}}{\Gamma \vdash K}$$

(Type-Arrow)

$$\frac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \to B}$$

(Val-x)

$$\frac{\Gamma, x : A, \Delta \vdash \diamond}{\Gamma, x : A, \Delta \vdash x : A}$$

(Val-Func)

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M \ : \ A \to B}$$

(Val-Appl)

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

3 Usage

Make sure you have Rust toolchain installed. Type `cargo run` anywhere inside the code directory to start the program.

The program interacts with the user through command line. Each type rule above corresponds to a command, whose name is the rule's name written in lowercase, with dash replaced by underscore. For example, a command with no argument is `env_empty`.

After issuing a command correctly, a list of all the valid judgements we have so far will be printed to the screen, labeled from 0. The last judgement in the list is the one derived by the latest command. Labels are used to refer to previous judgements in command arguments.

The arguments of commands match the premises of the type rules. A special case is `val_x <jid> <var>`, where you should provide the variable to be extracted as the second argument.

An example of commands deriving the following judgement is given in `example.txt` in the code directory.

$$y : K \to K \vdash \lambda z : K.(yz) \ : \ K \to K$$

Finally, type `exit` to gently stop the program.

References

[1] Luca Cardelli. "Type Systems". In: *The Computer Science and Engineering Handbook*. Ed. by Allen B. Tucker. CRC Press, 2004, pp. 2277–2306.