1 Lecture

1.1 Foundations

Halting problem: given a Turing machine, decide whether it halts on a given input or not.

Halting problem is not decidable. Given the correspondence between the lambda calculus and Turing machines, normalisation of lambda expressions is also not decidable.

1.2 Operational Semantics

$$(\lambda X.E)F \to E[X \mapsto F] \tag{1}$$

$$\frac{E \to E'}{\lambda X.E \to \lambda X.E'} \tag{2}$$

$$\frac{E \to E'}{EF \to E'F} \tag{3}$$

$$\frac{F \to F'}{EF \to EF'} \tag{4}$$

1.3 Reduction Strategies

An expression may be either non-normalizable, normalizable or strongly normalising.

- Normal order (call-by-name) reduction strategy always reduces leftmost outermost redex first.
- Lazy reduction strategy always reduces leftmost top-level redex.
- Applicative order (call-by-value) strategy always chooses the leftmost redex $(\lambda X.E)F$ such that F is not a redex itself.

Normal order reduction strategy always find the normal form of a normalizable term.

1.4 Type Inference

1.4.1 F1

$$\begin{aligned} Expr ::= {} & X \mid \\ & \lambda X : Type.Expr \mid \\ & Expr\ Expr \end{aligned} \tag{5}$$

$$\begin{aligned} Type ::= {} & BasicType \mid \\ & Type \to Type \end{aligned} \tag{6}$$

$$Type(\Gamma, x) = t \text{ if } (x : t) \in \Gamma \tag{7}$$

$$Type(\Gamma, \lambda x : t.e) = t \to Type(\Gamma \cup \{(x : t)\}, e) \text{ if } x \notin dom(\Gamma) \tag{8}$$

$$Type(\Gamma, ef) = t \text{ if } Type(\Gamma, e) = Type(\Gamma, f) \to t \tag{9}$$

1.4.2 F2

$$\begin{aligned} Expr ::= & \ X \mid \\ & \lambda X : Type.Expr \mid \\ & Expr \ Expr \mid \\ & \lambda T.Expr \mid \\ & Expr \ Type \end{aligned} \tag{10}$$

$$\begin{aligned} Type ::= & \ T \mid \\ & Type \to Type \mid \\ & \forall T.Type \end{aligned} \tag{11}$$

$$Type(\Gamma, x) = t \text{ if } (x : t) \in \Gamma \tag{12}$$

$$Type(\Gamma, \lambda x : t.e) = t \to Type(\Gamma \cup \{(x : t)\}, e) \text{ if } wellFormed(\Gamma, t) \wedge x \notin dom(\Gamma) \tag{13}$$

$$Type(\Gamma, ef) = t \text{ if } Type(\Gamma, e) = Type(\Gamma, f) \to t \tag{14}$$

$$Type(\Gamma, \lambda t.e) = \forall t.Type(\Gamma \cup \{t\}, e) \tag{15}$$

$$Type(\Gamma, e \ t) = v[u \mapsto t] \text{ if } Type(\Gamma, e) = \forall u.v \wedge wellFormed(\Gamma, t) \tag{16}$$

$$wellFormed(\Gamma, t) \equiv t \in dom(\Gamma) \tag{17}$$

$$wellFormed(\Gamma, t \to u) \equiv wellFormed(\Gamma, t) \wedge wellFormed(\Gamma, u) \tag{18}$$

$$wellFormed(\Gamma, \forall t.u) \equiv wellFormed(\Gamma \cup \{t\}, u) \tag{19}$$

2 Seminar

1. Evaluate the following expressions under various evaluation strategies: $(true \ I \ I)x$, $(true \ I \ \Omega)x$, $(true \ \Omega \ I)x$, $(true \ \Omega \ \Omega)x$.
2. Implement the first type inference algorithm.