Logic in Computer Software
Course Note #1: Functional Programming Language

Yukiyoshi Kameyama.

We introduce a minimal core calculus for typed functional programming languages. It is based on lambda calculus.

1 Syntax (構文)

$$
\begin{array}{ll}
x, y, z, \ldots & \text{variable (変数)} \\
c ::= 0 \mid 1 \mid -1 \mid \cdots \mid \texttt{true} \mid \texttt{false} & \text{constant (定数)} \\
e ::= c \mid x \mid e_1 + e_2 \mid e_1 - e_2 \mid e_1 = e_2 & \text{term (項)} \\
\quad \mid \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 & \\
\quad \mid \lambda x.\ e \mid e_1\ e_2 & \text{unary function (1 引数関数)} \\
\quad \mid \lambda(x, y, ..., z).\ e \mid e_1\ (e_2, e_3, ..., e_n) & \text{n-ary function (n 引数関数)} \\
\quad \mid \texttt{fix } f(x).\ e \mid \texttt{fix } f(x, y, ..., z).\ e & \text{recursive function (再帰関数)}
\end{array}
$$

構文上の曖昧さをなくすために、適宜、かっこを使ったり省略したりします。(We use parenthesis to disambiguate syntax.)

The term $\lambda(x, y, z).\ e$ represents a function which takes three arguments. If e_1 is such a function, it may be used in the form $e_1(e_2, e_3, e_4)$ where e_2, e_3, e_4 are actual parameters.

Examples of terms:

$$
\begin{array}{l}
1 + x \\
\texttt{if } x = 0 \texttt{ then } 1 \texttt{ else } 2 \\
\texttt{if } x = 0 \texttt{ then } y + 1 \texttt{ else } f(2 + z) \\
f\ 0 \\
f(g(h\ 0)) \\
\lambda x.\ x \\
\lambda x.\ x + 1 \\
(\lambda x.\ x + 1)\ 3 \\
(\lambda x.\ x + 1)\ ((\lambda x.\ x + 1)\ 3) \\
(\lambda f.\ f(f\ 3))(\lambda x.\ x + 1) \\
\lambda x.\ \texttt{if } x = 0 \texttt{ then } 1 \texttt{ else } f(x - 1) \\
\texttt{fix } f(x).\ \texttt{if } x = 0 \texttt{ then } 1 \texttt{ else } f(x - 1)
\end{array}
$$

Example 1:

$$\texttt{fix}\ f(x,y).\ \ \texttt{if}\ x=0\ \texttt{then}\ \texttt{true}\ \texttt{else}\ \texttt{if}\ y=0\ \texttt{then}\ \texttt{false}\ \texttt{else}\ f(x-1,y-1)$$

```
In OCaml:
let rec f x y  =
  if x=0 then true
  else if y=0 then false
  else f (x-1) (y-1)
```

Example 2:

$$\texttt{fix}\ f(x,y).\ \ \texttt{if}\ x=y\ \texttt{then}\ x\ \texttt{else}\ \texttt{if}\ x\langle y\ \texttt{then}\ f(x,y-x)\ \texttt{else}\ f(x-y,y)$$

```
In OCaml:
let rec f x y =
  if x=y then x
  else if x<y then f x (y-x)
  else f (x-y) y
```

Computation Rules (計算規則)

The notation $e_1 \rightsquigarrow e_2$ means that, by evaluating (computing) the term e_1 by 1 step, we obtain the term e_2 as its result.

A value (the final result of a computation) is defined as follows:

$$v ::= c \mid \lambda x.\ e \mid \lambda(x_1,\cdots,x_n).\ e \mid \texttt{fix}\ f(x).\ e \mid \texttt{fix}\ f((x_1,\cdots,x_n)).\ e$$

$$\begin{aligned}
v_1 + v_2 &\rightsquigarrow \cdots \text{(ordinary addition)}\\
v_1 - v_2 &\rightsquigarrow \cdots \text{(ordinary subtraction)}\\
v_1 = v_2 &\rightsquigarrow \cdots \text{(ordinary equality test)}\\
\texttt{if}\ \texttt{true}\ \texttt{then}\ e_2\ \texttt{else}\ e_3 &\rightsquigarrow e_2\\
\texttt{if}\ \texttt{false}\ \texttt{then}\ e_2\ \texttt{else}\ e_3 &\rightsquigarrow e_3\\
(\lambda x.\ e)\ v &\rightsquigarrow e[x := v]\\
(\lambda(x_1,\cdots,x_n).\ e)\ (v_1,\cdots,v_n) &\rightsquigarrow e[x_1,\cdots,x_n := v_1,\cdots,v_n]\\
(\texttt{fix}\ f(x).\ e)\ v &\rightsquigarrow e[x := v][f := \texttt{fix}\ f(x).\ e]\\
(\texttt{fix}\ f((x_1,\cdots,x_n)).\ e)\ (v_1,\cdots,v_n) &\rightsquigarrow e[(x_1,\cdots,x_n) := (v_1,\cdots,v_n)][f := \texttt{fix}\ f((x_1,\cdots,x_n)).\ e]
\end{aligned}$$

The notation $e_1 \rightsquigarrow^* e_2$ is a computation by 1 or more steps.

Example of computation:

$$(\lambda f.\ f(f(10)))\ (\lambda x.\ x+1) \rightsquigarrow f(f(10))[f := \lambda x.\ x+1] = (\lambda x.\ x+1)((\lambda x.\ x+1)(10)) \rightsquigarrow (\lambda x.\ x+1)(10{+}1) \rightsquigarrow^* 12$$

Type System (型システム)

A type (a simple type) is defined as follows:

$$\sigma, \tau ::= \texttt{int} \mid \texttt{bool} \mid \sigma \to \tau \mid (\sigma_1, \sigma_2, \cdots, \sigma_n) \to \tau$$

Example of types: $\texttt{int}$, $\texttt{bool}$, $(\texttt{int} \to \texttt{bool}) \to (\texttt{int} \to \texttt{bool})$, $(\texttt{int}, \texttt{int}, \texttt{bool}) \to \texttt{int}$.
Rules to derive types of terms:

$$\frac{(x : \sigma) \text{ is an element of } \Gamma}{\Gamma \vdash x : \sigma}\ \text{var} \qquad \frac{(m \text{ is an integer constant})}{\Gamma \vdash m : \texttt{int}}\ \text{const1} \qquad \frac{(b \text{ is a boolean constant})}{\Gamma \vdash b : \texttt{bool}}\ \text{const2}$$

$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 + e_2 : \texttt{int}}\ \text{plus} \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 - e_2 : \texttt{int}}\ \text{minus}$$

$$\frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda(x).\ e\ :\ \sigma \to \tau}\ \text{fun} \qquad \frac{\Gamma \vdash e : \sigma \to \tau \quad \Gamma \vdash f : \sigma}{\Gamma \vdash e\ f : \tau}\ \text{apply}$$

$$\frac{\Gamma, x_1 : \sigma_1, \cdots, x_n : \sigma_n \vdash e : \tau}{\Gamma \vdash \lambda(x_1, \cdots, x_n).\ e\ :\ \sigma \to \tau}\ \text{fun2} \qquad \frac{\Gamma \vdash e : \sigma \to \tau \quad \Gamma \vdash f : \sigma}{\Gamma \vdash e\ f : \tau}\ \text{apply2}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \sigma \quad \Gamma \vdash e_3 : \sigma}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \sigma}\ \text{if}$$

$$\frac{\Gamma, f : \sigma \to \tau, x : \sigma \vdash e : \tau}{\Gamma \vdash \texttt{fix}(f : \sigma \to \tau)(x : \sigma).\ e\ :\ \sigma \to \tau}\ \text{fix}$$

Example of type derivation (型の導出の例).

$$\frac{\dfrac{}{\cdot \vdash 0 : \texttt{int}} \quad \dfrac{}{\cdot \vdash 1 : \texttt{int}}}{\cdot \vdash 0 + 1 : \texttt{int}}$$

Let $\Gamma = x : \texttt{int}, y : \texttt{bool}$.

$$\frac{\dfrac{\dfrac{\dfrac{}{\Gamma \vdash y : \texttt{bool}} \quad \dfrac{}{\Gamma \vdash x : \texttt{int}} \quad \dfrac{\dfrac{}{\Gamma \vdash x : \texttt{int}} \quad \dfrac{}{\Gamma \vdash 1 : \texttt{int}}}{\Gamma \vdash x + 1 : \texttt{int}}}{\Gamma \vdash \texttt{if}\ y\ \texttt{then}\ x\ \texttt{else}\ x + 1 : \texttt{int}}}{y : \texttt{bool} \vdash \lambda(x : \texttt{int}).\ \texttt{if}\ y\ \texttt{then}\ x\ \texttt{else}\ x + 1 : \texttt{int} \to \texttt{int}}}{\cdot \vdash \lambda(y : \texttt{bool}).\ \lambda(x : \texttt{int}).\ \texttt{if}\ y\ \texttt{then}\ x\ \texttt{else}\ x + 1 : \texttt{int} \to \texttt{int} \to \texttt{int}}$$

Let $\Delta = x : \texttt{bool} \to \texttt{bool}, y : \texttt{bool}$.

$$\frac{\dfrac{\dfrac{\dfrac{}{\Delta \vdash x : \texttt{bool} \to \texttt{bool}} \quad \dfrac{\dfrac{}{\Delta \vdash x : \texttt{bool} \to \texttt{bool}} \quad \dfrac{}{\Delta \vdash y : \texttt{bool}}}{\Delta \vdash xy : \texttt{bool}}}{\Delta \vdash x(xy) : \texttt{bool}}}{x : \texttt{bool} \to \texttt{bool} \vdash \lambda(y : \texttt{bool}).\ x(xy) : \texttt{bool} \to \texttt{bool}}}{\cdot \vdash (\lambda(x : \texttt{bool} \to \texttt{bool}).\ \lambda(y : \texttt{bool}).\ x(xy)) : (\texttt{bool} \to \texttt{bool}) \to (\texttt{bool} \to \texttt{bool})}$$

Exercise 1. (練習問題)

(1) Give type inference rules for other constructs , e.g., $e - f$, $e = f$ and let-in.

Exercise 2. (練習問題)

(1) Derive $x : \texttt{int} \to \texttt{int}, y : \texttt{int} \vdash x(10) + x(y) : \texttt{int}$.

(2) Derive $\cdot \vdash \lambda(x : \texttt{int}).\ x + 1\ :\ \texttt{int} \to \texttt{int}$.

(3) (optional) Derive the type of factorial function.