5118014 Programming Language Theory

Ch 9. First-class Functions

Shin Hong

First-class Functions

- First-class function is a sort of values
  - a function is defined inside an expression
    - there must be a function defining expression
  - first-class function is much more expressive in program abstraction than first-order function
- Hereafter, for $f(x)$, we say that function f is applied to x
  - rather than function f is called
  - call $f(x)$ as function application rather than function call

FVAE

- let's extend VAE to FVAE by adding first-class function
  - not extending F1VAE

- syntax
  - add new expressions for function application and function definition
- semantics
  - extend the value domain
  - add new semantics rules for function application and function definition

Function Application

- In F1VAE, a function call is made with a function name identifier
- In FVAE, a function to be applied is given as an expression
  - not as an identifier
  - Ex. Scala

```
def makeAdder(x: Int): Int => Int =
  (y: Int) => x + y
makeAdder(3)(5)
```

Syntax

$$e ::= \cdots \mid \lambda x.e \mid e\ e$$

- lambda abstraction for defining anonymous function
  - x is the parameter and e is the body
- function application
  - the first expression denotes the target function, the second the argument

Value

$$V = \mathbb{Z} \cup Id \times E \times Env$$

$$v ::= \; n \in \mathbb{Z} \;\mid\; < \lambda x.e, \sigma >$$

- A value is not always an integer, but an integer or closure in FVAE
  - a closure is a pair of a lambda abstraction and an environment
  - examples
    - val $f = \lambda x.1 + x$ in $f\ 2$
    - val $f = \lambda y.\lambda x.y + x$ in $f\ 1\ 2$

Semantics (1/2)

$$Env = Id \rightharpoonup V$$

$$\Rightarrow \subseteq Env \times E \times V$$

$$\frac{\sigma \vdash e_1 \Rightarrow n_1 \qquad \sigma \vdash e_2 \Rightarrow n_2}{\sigma \vdash e_1 + e_2 \Rightarrow n_1 + n_2} \quad \text{for } n_1 \in \mathbb{Z} \text{ and } n_2 \in \mathbb{Z}$$

$$\frac{\sigma \vdash e_1 \Rightarrow v_1 \qquad \sigma[x \mapsto v_1] \vdash e_2 \Rightarrow v_2}{\sigma \vdash \mathsf{val}\ x{=}e_1\ \mathsf{in}\ e_2 \Rightarrow v_2}$$

Semantics (2/2)

$$\sigma \vdash \lambda x.e \Rightarrow \langle \lambda x.e, \sigma \rangle \quad [\textsc{Fun}]$$

- a lambda abstraction creates a closure capturing the current environment

$$\frac{\sigma \vdash e_1 \Rightarrow \langle \lambda x.e, \sigma' \rangle \qquad \sigma \vdash e_2 \Rightarrow v' \qquad \sigma'[x \mapsto v'] \vdash e \Rightarrow v}{\sigma \vdash e_1\ e_2 \Rightarrow v} \quad [\textsc{App}]$$

Example

$$\emptyset \vdash (\lambda x.\lambda y.x+y)\ 1\ 2\ \Rightarrow 3$$

$$\frac{\sigma \vdash e_1 \Rightarrow \langle \lambda x.e, \sigma' \rangle \qquad \sigma \vdash e_2 \Rightarrow v' \qquad \sigma'[x \mapsto v'] \vdash e \Rightarrow v}{\sigma \vdash e_1\ e_2 \Rightarrow v}\ [\textsc{App}]$$

- $\emptyset \vdash (\lambda x.\lambda y.x+y)1 \Rightarrow \langle \lambda y.x+y, [x \mapsto 1] \rangle$
    - $\emptyset \vdash \lambda x.\lambda y.x+y \Rightarrow \langle \lambda y.x+y,\ \emptyset \rangle$
    - $\emptyset \vdash 1 \Rightarrow 1$
    - $[x \mapsto 1] \vdash \lambda x.\lambda y.x+y \Rightarrow < \lambda y.x+y, [x \mapsto 1] >$
- $\emptyset \vdash 2 \Rightarrow 2$
- $[x \mapsto 1, y \mapsto 2] \vdash x+y \Rightarrow 3$