CIS 425: Principles of Programming Languages
Lecture 13: Interpreter

Xiaodong Quan

21 May 2019

Today's lecture was focused on Interpreter. Also, reviewed call by value & call by name. This note will make some review first, then will head in examples from the lecture. Understand these examples and the process is the **major** target for this lecture.

- Review (see Section 1)
- Example_1 (see Section 2)
- Example_2 (see Section 3)

1 Review

In former lectures, we talked about call by value & call by name.
There would be an example to help you review this part.

- Call-by-value: arguments are evaluated before a function is entered

  (\x. \y. y x) (2 + 2) (\x. x + 1)
  => (\x. \y. y x) 4 (\x. x + 1)
  => (\y. y 4) (\x. x + 1)
  => (\x. x + 1) 4
  => 4 + 1
  => 5

- Call-by-name: arguments are passed unevaluated

  (\x. \y. y x) (2 + 2) (\x. x + 1)
  => (\y. y (2 + 2)) (\x. x + 1)
  => (\x. x + 1) (2 + 2)
  => (2 + 2) + 1
  => 4 + 1
  => 5

Note: The Interpreter is using **Pass by Name** in Example_2 Later.

2 Example_1

- Interpreter with dynamic scope

—— Interpreter ——

$\text{env} \vdash \text{n} \Rightarrow \text{n}$
$\text{env} \vdash \text{true} \Rightarrow \text{true}$
$\text{env} \vdash \text{false} \Rightarrow \text{false}$
$\text{env} \vdash (\text{fn x} \Rightarrow \text{e}) \Rightarrow (\text{fn x} \Rightarrow \text{e})$

$$\frac{\text{env} \vdash \text{e} \Rightarrow (\text{fn x} \Rightarrow e_2) \qquad \text{env} \vdash e_1 \Rightarrow v_1 \qquad \text{env}\ (\text{x}, v_1) \vdash e_2 \Rightarrow v_2}{\text{env} \vdash \text{e}\ e_1 \Rightarrow v_2}\ \text{app}$$

$$\frac{\text{env} \vdash e_1 \Rightarrow v_1 \qquad \text{env}\ (\text{x}, v_1) \vdash e_2 \Rightarrow v_2}{\text{env} \vdash \text{let x} = e_1 \text{ in } e_2 \Rightarrow v_2}\ \text{let}$$

—— Pseudo Code ——

```
let x = 1 in
  M | let f = fn z => x in
    | N | let x = 0 in
    |   | f x
```

—— Solving Steps from 1 to 5 (bottom is 1) ——
—— Read from Downward to Upward ——

$$\frac{\boxed{(\text{x}, 1), (\text{f}, \text{fn z} \Rightarrow \text{x}), (\text{x}, 0) \vdash \text{f} \Rightarrow \text{fn z} \Rightarrow \text{x}} \quad \boxed{(\text{x}, 1), (\text{f}, \text{fn z} \Rightarrow \text{x}), (\text{x}, 0) \vdash \text{z} = \text{x} \Rightarrow 0} \quad \boxed{(\text{x}, 1), (\text{f}, \text{fn z} \Rightarrow \text{x}), (\text{x}, 0), (\text{z}, 0) \vdash \text{x} \Rightarrow \boldsymbol{0}}}{}\ \underline{\textit{app}}$$

$$\frac{\boxed{(\text{x}, 1), (\text{f}, \text{fn z} \Rightarrow \text{x}) \vdash \text{let x} = 0} \qquad \boxed{(\text{x}, 1), (\text{f}, \text{fn z} \Rightarrow \text{x}), (\text{x}, 0) \vdash \text{f x} \Rightarrow \boldsymbol{0}}}{}\ \underline{\textit{let}}$$

$$\frac{\boxed{(\text{x}, 1) \vdash (\text{fn z} \Rightarrow \text{x}) \Rightarrow (\text{fn z} \Rightarrow \text{x})} \qquad \boxed{(\text{x}, 1), (\text{f}, \text{fn z} \Rightarrow \text{x}) \vdash \boldsymbol{N} \Rightarrow \boldsymbol{0}}}{}\ \underline{\textit{let}}$$

$$\frac{\boxed{(\text{x}, 1) \vdash \boldsymbol{M} \Rightarrow \boldsymbol{0}}}{\boxed{\vdash \text{let x} = 1}}\ \underline{\textit{let}}$$

Explain:

- Step_1: Bound 1 with x in environment;
- Step_2: Working on block M;
- Step_3: Bound function with f in environment;
- Step_4: Bound 0 with x, working on f x;
- Step_5: Since we need use f, take f out of environment, then we need z as parameter, we can see z = x(in step4) = 0(in environment of step4). Then we bound 0 with z, and function f return x. Find the most recent x, which is 0. Hence the result is 0;

- Interpreter with static scope

—— Interpreter ——

Note: In order to be static, we need record current environment while bound function in an environment.
We can see the application part was different.
The current environment be recorded as well;

```
env ⊢ n => n
env ⊢ true => true
env ⊢ false => false
env ⊢ (fn x => e) => (fn x => e)
```

```
env ⊢ e => (fn x => e2, env1)     env ⊢ e1 => v1     env1 (x, v1) ⊢ e2 => v2
------------------------------------------------------------------------ app
env ⊢ e e1 => v2
```

```
env ⊢ e1 => v1     env1 (x, v1) ⊢ e2 => v2
------------------------------------------ let
env ⊢ let x = e1 in e2 => v2
```

—— Pseudo Code ——

```
let x = 1 in
  M | let f = fn z => x in
    | N | let x = 0 in
    |   | f x
```

—— Solving Steps from 1 to 5 (bottom is 1) ——
—— Read from Downward to Upward ——

```
(x, 1), (f, ((fn z => x), (x, 1))), (x, 0) ⊢ f => ((fn z => x), (x, 1))   (x, 1), (f, ((fn z => x), (x, 1))), (x, 0) ⊢ z = x => 0   (x, 1) ⊢ x => 1
------------------------------------------------------------------------------------------------------------------------------------------ app
(x, 1), (f, ((fn z => x), (x, 1))) ⊢ let x = 0        (x, 1), (f, ((fn z => x), (x, 1))), (x, 0) ⊢ f x => 1
------------------------------------------------------------------------------------------------------------------------------------------ let
(x, 1) ⊢ (fn z => x) => ((fn z => x), (x, 1))        (x, 1), (f, ((fn z => x), (x, 1))) ⊢ N => 1
------------------------------------------------------------------------------------------------------------------------------------------ let
(x, 1) ⊢ M => 1
------------------------------------------------------------------------------------------------------------------------------------------ let
⊢ let x = 1
```

Explain:

- Step_1: Bound 1 with x in environment.
- Step_2: Working on block M
- Step_3: Bound function with f, and record current environment in f
- Step_4: Bound 0 with x, working on f x
- Step_5: Since we need use f, take f out of environment, then we need z as parameter, we can see z = x(in step4) = 0(in environment of step4). Then we bound 0 with z, and function f return x. Since the f have recorded environment before, we only use the recorded environment, hence x = 1;

3 Example2

- Interpreter with dynamic scope

—— Interpreter ——

$env \vdash n \Rightarrow n$

$env \vdash true \Rightarrow true$

$env \vdash false \Rightarrow false$

$env \vdash (fn\ x \Rightarrow e) \Rightarrow (fn\ x \Rightarrow e)$

$$\frac{env \vdash e \Rightarrow (fn\ x \Rightarrow e_2, env_1) \qquad env \vdash e_1 \Rightarrow v_1 \qquad env_1(x, e_1) \vdash e_2 \Rightarrow v_2}{env \vdash e\ e_1 \Rightarrow v_2}\ \text{app}$$

$$\frac{env \vdash e_1 \Rightarrow v_1 \qquad env\ (x, v_1) \vdash e_2 \Rightarrow v_2}{env \vdash \text{let } x = e_1 \text{ in } e_2 \Rightarrow v_2}\ \text{let}$$

$$\frac{\text{LookUp}\ (x, env) = e \longrightarrow env \vdash e \Rightarrow v}{env \vdash x \Rightarrow v}\ \text{LookUp}$$

—— Pseudo Code ——

```
let x = 1 in
  M | let y = x + 1 in
    | N | let x = 9 in
    |   | y
```

—— Solving Steps from 1 to 5 (bottom is 1) ——
—— Read from Downward to Upward ——

$$\frac{\dfrac{\dfrac{\dfrac{\boxed{\text{LookUp}(y, env_1) = x + 1} \longrightarrow \boxed{(x, 1), (y, x{+}1), (x, 9) \vdash x + 1 \Rightarrow \mathbf{\mathit{10}}}}{\boxed{(x, 1), (y, x + 1), (x, 9) \vdash y \Rightarrow \mathbf{\mathit{10}}}}\ \textit{app}}{\boxed{(x, 1), (y, x + 1) \vdash \text{let } x = 9}}\ \textit{let}}{\boxed{(x, 1) \vdash \mathbf{\mathit{M}} \Rightarrow \mathbf{\mathit{10}}}}\ \textit{let}}{\boxed{\vdash \text{let } x = 1}}\ \textit{let}$$

Explain:

- Step_1: Bound 1 with x in environment;
- Step_2: Working on block M;
- Step_3: Bound function with y, and 9 with x in environment;
- Step_4: Working on y;
- Step_5: We will use LookUp of Interpreter, follow the format. Part ahead Arrow, the x is y, env is environment of step4, e is function bound with y. Then we got part after Arrow. Follow after Arrow part. Find nearest x in environment, x=9, so x+1=10. hence the result is 10;

- Interpreter with static scope

—— Interpreter ——

Note: In order to be static, we need record current environment while bound function in an environment.
We can see the application part was different.
The current environment be recorded as well.
And the LookUp changed as well;

$$\text{env} \vdash \text{n} \Rightarrow \text{n}$$
$$\text{env} \vdash \text{true} \Rightarrow \text{true}$$
$$\text{env} \vdash \text{false} \Rightarrow \text{false}$$
$$\text{env} \vdash (\text{fn x} \Rightarrow \text{e}) \Rightarrow (\text{fn x} \Rightarrow \text{e})$$

$$\frac{\text{env} \vdash \text{e} \Rightarrow (\text{fn x} \Rightarrow \text{e}_2, \text{env}_1) \quad \text{env}_1(\text{x}, (\text{e}_1, \text{env})) \vdash \text{e}_2 \Rightarrow \text{v}_2}{\text{env} \vdash \text{e e}_1 \Rightarrow \text{v}_2}\,\text{app}$$

$$\frac{\text{env}_1\,(\text{x}, (\text{e}_1, \text{env})) \vdash \text{e}_2 \Rightarrow \text{v}_2}{\text{env} \vdash \text{let x} = \text{e}_1 \text{ in e}_2 \Rightarrow \text{v}_2}\,\text{let}$$

$$\frac{\text{LookUp}\,(\text{x}, \text{env}) = (\text{e}, \text{env}_1) \longrightarrow \text{env}_1 \vdash \text{e} \Rightarrow \text{v}}{\text{env} \vdash \text{x} \Rightarrow \text{v}}\,\text{LookUp}$$

—— Pseudo Code ——

```
let x = 1 in
   M | let y = x + 1 in
     | N | let x = 9 in
     |   | y
```

—— Solving Steps from 1 to 5 (bottom is 1) ——
—— Read from Downward to Upward ——

$$\dfrac{\dfrac{\dfrac{\dfrac{\boxed{\text{LookUp}(\text{y}, \text{env}_1) = (\text{x} + 1, (\text{x}, 1))} \longrightarrow \boxed{(\text{x}, 1) \vdash \text{x} + 1 \Rightarrow \boldsymbol{2}}}{\boxed{(\text{x}, 1), (\text{y}, (\text{x} + 1, (\text{x}, 1))), (\text{x}, 9) \vdash \text{y} \Rightarrow \boldsymbol{2}}}\,\underline{\textit{app}}}{\boxed{(\text{x}, 1), (\text{y}, (\text{x} + 1, (\text{x}, 1))) \vdash \text{let x} = 9}}\,\underline{\textit{let}}}{\boxed{(\text{x}, 1) \vdash \boldsymbol{M} \Rightarrow \boldsymbol{2}}}\,\underline{\textit{let}}}{\boxed{\vdash \text{let x} = 1}}\,\underline{\textit{let}}$$

Explain:

- Step_1: Bound 1 with x in environment;
- Step_2: Working on block M;
- Step_3: Bound function with y, and 9 with x in environment;
- Step_4: Working on y;
- Step_5: We will use LookUp of Interpreter, follow the format. Part ahead Arrow, the x is y, env is environment of step4, e is function bound with t, env1 is previous environment which recorded in f. Then we got part after Arrow. Follow after Arrow part. Find nearest x in environment, x=1, so x+1=2. hence the result is 2;