

Lecture 22: Writing Interpreters 2/Undecidability

CSCI 101
Spring, 2019

Kim Bruce

PCF Semantics w/Environments

- Substitution slow & space consuming
- Can't handle terms w/free variables
- Environment allows to evaluate once.
- Meaning now separate set of values -- not just rewriting
- Meaning of function is closure, which carries around its environment of definition.

The Problem

- Program:
 - $y = 4$
 - $f\ x = x + y$
 - $g\ (h) = \text{let } y = 5 \text{ in } (h\ 2) + y$
 - $g(f)$
- When evaluate $(h\ 2)$, the needed y is out of scope!

Values of Answers

- Key difference w/ new interpreter
 - Update environment, not rewrite term!
 - Not destructive!
- Mutually recursive type definitions:

```
data Value = NUM Int | BOOL Bool | SUCC | PRED |
           ISZERO | CLOSURE (String, Term, Env) |
           THUNK (Term, Env) | ERROR (String, Value)
type Env = [(String, Value)]
```

Solving the Problem

- Program:

- $y = 4$
- $f\ x = x + y$
- $g\ (h) = \text{let } y = 5 \text{ in } (h\ 2) + y$
- $g(f)$

- f evaluates to $\langle \text{fn } x \Rightarrow x + y, [y \rightarrow 4] \rangle$

- $g(f)$ partially evaluates to $(h\ 2) + y$ in environment where $\text{env} = [y \rightarrow 5, h \rightarrow \langle \text{fn } x \Rightarrow x + y, [y \rightarrow 4] \rangle]$

PCF Syntax & Semantics with Environments

$\text{env} :: \text{string} \rightarrow \text{value}$

- (0) $(\text{id}, \text{env}) \Rightarrow \text{env}(\text{id})$
- (1) $(n, \text{env}) \Rightarrow n$ for n an integer.
- (2) $(\text{true}, \text{env}) \Rightarrow \text{true}, (\text{false}, e) \Rightarrow \text{false}$
- (3) $(\text{error}, \text{env}) \Rightarrow \text{error}$
- (4) $(\text{succ}, \text{env}) \Rightarrow \text{succ}$, similarly for other initial functions
- (5)
$$\frac{(b, \text{env}) \Rightarrow \text{true} \quad (e_1, \text{env}) \Rightarrow v}{(\text{if } b \text{ then } e_1 \text{ else } e_2, \text{env}) \Rightarrow v}$$

More PCF Semantics

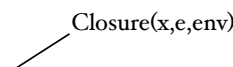
$$(6) \frac{(b, \text{env}) \Rightarrow \text{false} \quad (e_2, \text{env}) \Rightarrow v}{(\text{if } b \text{ then } e_1 \text{ else } e_2, \text{env}) \Rightarrow v}$$

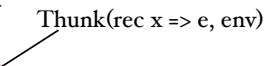
$$(7) \frac{(e_1, \text{env}) \Rightarrow \text{succ} \quad (e_2, \text{env}) \Rightarrow n}{((e_1\ e_2), \text{env}) \Rightarrow (n+1)}$$

(8) ...

(9) ...

Revised PCF Semantics

- (10) $((\text{fn } x \Rightarrow e), \text{env}) \Rightarrow \langle \text{fn } x \Rightarrow e, \text{env} \rangle$

- (11)
$$\frac{(e_1, \text{env}) \Rightarrow \langle \text{fn } x \Rightarrow e_3, \text{env}' \rangle \quad (e_2, \text{env}) \Rightarrow v_1 \quad (e_3, \text{env}'[v_1/x]) \Rightarrow v}{((e_1\ e_2), \text{env}) \Rightarrow v}$$
- (12)
$$\frac{(e, \text{env}[(\text{rec } x \Rightarrow e)/x]) \Rightarrow v}{((\text{rec } x \Rightarrow e), \text{env}) \Rightarrow v}$$



See code on-line in
[PCFEnvInterpreter.hs](#)

Imperative Languages

Adding State For Assignment

$$\frac{(e_1, ev, s) \Rightarrow (m, s') \quad (e_2, ev, s') \Rightarrow (n, s'')}{(e_1 + e_2, ev, s) \Rightarrow (m+n, s'')}$$

$$\frac{(M, ev, s) \Rightarrow (v, s')}{(X := M, ev, s) \Rightarrow (v, s'[v / ev(X)])}$$

$$(fn\ x \Rightarrow M, ev, s) \Rightarrow (< fn\ x \Rightarrow M, ev >, s)$$

$$(f, ev, s) \Rightarrow (< fn\ x \Rightarrow M, ev' >, s'), \quad (N, ev, s') \Rightarrow (v, s''), \\ (M, ev'[v/X], s'') \Rightarrow (v', s''')$$

$$\frac{}{(f(N), ev, s) \Rightarrow (v', s''')}$$

Summary of Operational Semantics

- Meaning of program is sequence of states go through during execution
- Useful for compiler writers, complexity analysis
- Ideal is abstract machine that is simple enough that it is impossible to misunderstand operation.
- Should be easy to map to any computer.

If have time, come back later
to talk about axiomatic
semantics