Week 5 Lecture Notes: The Real World

Chris Stephenson

10 March 2016

1 Evaluation Strategies

A strange word to use
We make the languages.

The problem is that host languages have design defects, so evaluating them is confusing

When and What and How parameters evaluated?

When ?

1. myFunc (a, b, c+d)
   (a) expr1 = a
   (b) expr2 = b
   (c) expr3 = c+d

Evaluation Order Matters

myFunc (a++, a++, a++), order matters in such cases.

Are the parameters "left to right" or "right to left" ?

OR We can not evaluate parameters at all and simply pass expressions to function body. (All programming languages do this for some things.)

Racket is a strict language which means parameters evaluated.

(define (fact n)) (if (¡ n 1) 1 (* n (fact (- n 1))))

****
(define (my-if condition t f) (cond (condition t) (else f)))

(define (fact n)) (my-if (¡ n 1) 1 (* n (fact (- n 1))))

This is Wrong Evaluation !

It Works on # lang lazy

output :

(fact 7)
5040

In Java f(a) && g(b) is not a boolean expression.

They call it "short circuited evaluation". It is evaluated left to right and stop when we know the answer.

A Practical Example :

While (i ¡ a.length && a[i]!= 0){
..... i++; ....
}

What is wrong with next code template ?

int i = 0;
While (a[i]!= 0 && i ¡ a.length){
..... i++; ....
}

It gives Array Out Of Bounds Exception.

Is Evaluation Strict or Not ? Sometimes Strict ?

It is Sometimes Mostly Strict.

2 Environments

Substition is not how Real World works.
(For real reasons of efficiency.)

Our evaluation works the environment as additional parameter. When the evaluation evaluates an identifier, it looks up the value in the environment.

3 Data Definition

An environment is empty OR in identifier/ value pair plus an environment.

look up Env identifier – value or Error

extend Env environment identifier value – environment

(fundef myfunc (a b c)
(+ a (* bc)))
function definition

Example
(myfunc 3 7 (+ 2 9))
function application

An evaluator takes the environment as additional parameter when the evaluator evaluates an identifier it looks up the value in the environment.

(eval(extend Env (extend Env (empty Env, 'a, 3) 'b, 7) 'c, 11) (+ a (* b c)))

(fundef f1 x (f2 4)) x – acts like global environment
free identifiers

(fundef f2 y (+x y)) identifier capture
int f2 (int y) } Broken Invalid
(f1 (f2 3)) } wrong

this is called "dynamic scope" Dynamic Scope is a bug

evaluate (f1, f2) - x gets bound to 42, y gets bound to 4

(+ x y) looks up those values in the environment - The answer is 46 Wrong !

The Solution (for now)– When we apply a function, do not extend the current environment with formal parameter / value bindings. Extends the empty environment.

M [x := N]

1. M is an identifier
   (a) M = x then M[x:=N] is N
   (b) M = x then M[x:=N] is M
2. M is an application just do the two parts
3. M is a function definition

Case (a) is easy yM1 , y = x
Case (b) is hard yM1 , y != x (not equal)