HW/LAB2 REVIEW

NJU SICP22 TAs

关于遇到的困难

- **打破固有思维**：“函数是一等公民” ->函数也是数据的一种，并不特殊。“调用”会执行函数体，且严格遵守Lecture 2中“函数的运算顺序”，不调用时它与其他数无异。
- **正向分析**：遇到难以理解的部分->学会像程序一样思考，机器应该怎么执行，何时evaluate，如何lookup，走到哪步会解析出什么，是Function，是被调用Function的返回值，还是普通的int——“机器永远是对的”。
- **逆向思考**：我该如何破解这道题？->分析程序需要什么输出，它传入的参数会起到什么样的作用。如果输入的是函数，需要在什么阶段被调用？如果输出的是函数，它需要满足什么功能？

  (适当时候def helper创造)

Lab2p4

```
def cycle(f1, f2, f3):
    def f(n):
        def g(x):
            i=0
            while i<n:
                if i%3==0:
                    x=f1(x)
                if i%3==1:
                    x=f2(x)
                if i%3==2:
                    x=f3(x)
                i+=1
            return x
        return g
    return f
```

发现n用不了？

如果需要对n的值进行操作：

```
def cycle(f1,f2,f3):
    def helper(n,x):
        ......
    return lambda n:lambda x:helper(n,x)
```

```
def cycle(f1, f2, f3):
    def f(n):
        def g(x):
            t=n
            ...#对t进行操作
            return x
        return g
    return f
```

Hw2p4

- 卡在mission2的： return lambda g: mission2(g(f))

  lambda g:mission2(g(f)) 被X调用的时候：

  1.g与X绑定，并运算mission2(X(f))

  2. 先运算操作数X(f)

  3.得到的返回值再与mission2中的f绑定

  ->X=lambda func:(某个函数)

  ->X=lambda func:func（或者自己写一个lambda func:lambda x:x*2)

Hw2p5

```
def protected_secret(password, secret, num_attempts):
    def get_secret(password_attempt):
        ......
        return get_secret
    return get_secret
```

怎么改变num_attempts?
->无法改变就创造一个新的frame

```
def protected_secret(password, secret, num_attempts):
    def get_secret(password_attempt):
        ......
        return protected_secret(password,secret,num_attempts-1)
    return get_secret
```

Church_number

- $zero = \lambda f.\ \lambda \mathrm{x}.\ x$
- $one = \lambda f.\ \lambda \mathrm{x}.\ f(x)$
- $two = \lambda f.\ \lambda \mathrm{x}.\ f(f(x))$
- $church\_n = \lambda f.\ \lambda \mathrm{x}.\ f^n(x)$

```
def zero(f):
    return lambda x: x

def successor(n):
    return lambda f: lambda x: f(n(f)(x))
    #return lambda f:lambda x:n(f)(f(x))

def one(f):
    """Church numeral 1: same as successor(zero)"""
    return lambda x:f(x)

def two(f):
    """Church numeral 2: same as successor(successor(zero))"""
    return lambda x:f(f(x))

three = successor(two)

def church_to_int(n):
    return n(lambda x:x+1)(0)
```

- 定义church number的后继：如何改变f的嵌套层数？
- ->将上一元素的x的变为f(x)
- ->在上一元素的$f^n(x)$外嵌套f

Church_add

$$m = \lambda f.\ \lambda \mathrm{x}.\ f^m(x)$$
$$n = \lambda f.\ \lambda \mathrm{x}.\ f^n(x)$$
$$=> m + n = \lambda f.\ \lambda \mathrm{x}.\ f^{m+n}(x)$$

如何实现？

=>以m为基础，让其中的参数x变成$f^n(x)$

①拿到$f^n(x)$：n(f)(x)

②将其传入m的参数中：m(f)(n(f)(x))

③还原外层f与x：$\lambda f.\ \lambda \mathrm{x}.\ \mathrm{m(f)(n(f)(x))}$

Church_mul

$$m = \lambda f.\ \lambda \mathrm{x}.\ f^m(x)$$
$$n = \lambda f.\ \lambda \mathrm{x}.\ f^n(x)$$
$$=> m \times n = \lambda f.\ \lambda \mathrm{x}.\ f^{mn}(x)$$

如何实现？
=>以m为基础，让其中的参数f变成f^n
①拿到f^n：n(f)
②将其传入m的参数中：m(n(f))
③还原外层f：$\lambda f.\ \mathrm{m(n(f))}$

Church_pow

$$m = \lambda f.\ \lambda \mathrm{x}.\ f^m\ x$$

$$n = \lambda f.\ \lambda \mathrm{x}.\ f^n\ x$$

$$=> m^n = \lambda f.\ \lambda \mathrm{x}.\ f^{m^n}\ x$$

·如果我们对x进行m次f^n操作，就会得到$f^{mn}\ x$

·但是，x可不可以是f？

=>对f进行m次$(\lambda f.\ f^n)$操作？（每次乘1个 f^n，最终就会得到f^{n^m}）

=>可以构造$(\lambda f.\ \lambda \mathrm{x}.\ f^n\ x)^m\ f \rightarrow\ \lambda \mathrm{x}.\ f^{n^m}\ x$

=>发现m(n)刚好能形成上述结构！

此时m(n)=$\lambda \mathrm{x}_1.\ (\lambda f.\ \lambda \mathrm{x}.\ f^n\ x)^m\ x_1 = \lambda \mathrm{x}_1.\lambda \mathrm{x}_2.\ {\mathrm{x}_1}^{n^m}\ x_2$

$=\lambda f.\ \lambda \mathrm{x}.\ f^{m^n}\ x$！参数名不影响语义！

非常优雅地我们就会得到答案m(n)！