SYRUS YEUNG/RASMI LAMICHHANE/EHSAN KARIMI

SYRUS YEUNG

What is a continuation?

- A continuation saves the execution state of a program by passing a continuation, usually in the form of a callback to the parameter of the function. This callback will be used to pass in a function to let the program know what to do in a certain case such as a success case (i.e. success continuation)

Scala continuations

- In our Lab 6 project we utilized Success continuations in scala to tell our retest what to do in the case of success. We passed it as a curried function that took one argument and outputted another. This was useful because in certain cases we wanted to change or call the success continuation based on which case we were in.

Typically when we define functions they implicitly return some value:

```
Def foo( x ):
        Return x + 1
```

Whereas a continuations explicitly states what to do with the result :

```
Def foo(x,c):
        Return c(x+1)
```

Where c is the continuation function.

Recreating the function:

```
def twoxplusy(x,y):
      return 2*x+y
```

Using the functions:

```
def add(x,y,c): c(x+y)
def mul(x,y,c): c(x*y)
```

To create the following function:

```
def twoxplusy(x,y,c):
    mul(2,x,lambda v,y=y,c=c: add(v,y,c))
```

This function the the computation of of 2*x catches that in v and uses v to compute the addition portion in the continuation.

RASMI LAMICHHANE

#Python

```
#regular
x=4
def add(x):
  return x+1
```

```
#continuation
def c(a):
  return a

def cont_add(x,c):
  return c(x+1)
```

```
#lamds
 cont_add_lamda = lambda x: x + 1
```

#simple scala vs python code with and without continuation

This basically shows that lambda is anonymous function in python. Which is efficeient.
Lamda

```python
#python
mylist=[0,2,3]
def Mullist(mylist):
  def Mul(sc,mylist):
    if not len(mylist):
      return sc(1)
    elif mylist[0] == 0:
      return 0
    else: #case h::t
      return Mul(lambda acc: sc(mylist[0] * acc), mylist[1:])
  print Mul(lambda acc: acc, mylist)
MullistDelay(mylist)
```

```scala
#Scala

def MulListt(mylist: List[Int]): Int = {
 def mul(sc: Int => Int, mylist: List[Int]): Int = mylist match {
  case Nil => sc(1)
  case 0 :: _ => 0
  case h :: t => mul(acc => sc(h * acc), t)
 }
 mul(acc => acc, mylist)
}
```

I have grabbed the MulList code from the lab6. The code in the left is a conversion of the scala code to python. The goal of this conversion is to show how continuation works with list in python and scala because they are fairly similar to each other. The only difference is the syntax. We are defining a function MulList which take list and inside there we are defining and Mul which takes a sc and list.

Now the important things are here to describe. All the cases are color coded for more readability.

The empty case when the list is empty it's returning continuation of 1. When the head of the list/1st item in the list is 0 then return the 0. If other we want to do the actual calculation. In this case I'm using success continuation with the help of anonymous function.

In python I'm defining a lambda. Lambda is just a way of representing anonymous function in python. Instead of acc => in scala we are using lambda acc in python.

Main : Our function multiplies the element of the list when necessary.

EHSAN KARIMI

Applications:

Continuation is commonly used in programming web servers that supports multiple pages, simplifying the code and world view of the web server.
In C continuation used with *longjmp* to jump from the middle of function to another.
Continuation is also used in implementing Generators in languages such as python and Icon.

Reference

- https://en.wikipedia.org/wiki/Continuation-passing_style
- https://www.ps.uni-saarland.de/~duchier/python/continuations.html
- Lab5-ppl
- https://thelackthereof.org/docs/library/cs/continuations.pdf
- http://www.di.unipi.it/~nids/docs/longjump_try_trow_catch.html