

L9: First-class Function

Rachata Ausavarungnirun

(rachata.a@tggs.kmutnb.ac.th)

February 4th, 2020

Architecture Research Group

Software System Engineering

Thai-German Graduate School, KMUTNB

Before We Begin

- Assignment 2 is up
- Three more classes before the midterm
 - Then I will do one review session on Feb 13th
- Midterm: Tuesday Feb 18th ok?

Recap: First Class Function *- treat function as value*

- Functions become values
- Conceptually, this allows you to pass functions in, and return a function

- Example: Repeat a function n times

- `def nTimes[A](f: A => A, n: Int, x: A): A =`
`if (n==0) x else f(nTimes(f, n-1, x))`

f(f(f(x)))

● Base Case

Examples: Functions as Inputs

- Let's define:

```
def triple(x: Int) = 3*x
```

```
def addTwo(x: Int) = x+2
```

```
def doTail[T](xs: List[T]) = xs.tail
```

- What do these do?

```
nTimes(triple, 7, 11) 11 x 3 (repeat 7 time)
```

```
nTimes(addTwo, 4, 9) 9 + 2^4
```

```
nTimes(doTail, 2, List(3,5,2,4,9,7)) [2,4,9,7]
```

```
nTimes(doTail[Int], 2, List(3,5,2,4,9,7)) - Prevent bug of different type
```

Examples: Functions as Outputs

- def tripleNTimes(n: Int, x: Int) = {
 def triple(x: Int) = 3*x
 nTimes(triple, n, x) - Do triple n number of time
}

// use the shorthand form for defining a function

```
def tripleNTimes(n: Int, x: Int) =  
  nTimes((x: Int) => 3*x, n, x)
```

Input What function do

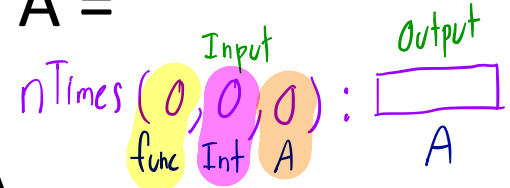
Scala: Methods vs. Functions

- When we write `def inc(x: Int) = x+1`
 - This is not really a function
 - `def` with parameters is a method
- In Scala, **method can be polymorphic**
- Also in Scala, **functions are never polymorphic**
 - They will have a type
- `inc _` gives a functional form, it takes an `Int`, and will return an `Int`

Types

- For now, let's assume functions are polymorphic

- `def nTimes[A](f: A => A, n: Int, x: A): A =
 if (n==0) x else f(nTimes(f, n-1, x))`



- This has the type `((A => A), Int, A) => A`

- What does this mean? *function that takes A produce A*

- In this same example, `A` is a placeholder for a type
- But, these functions does not have to be polymorphic
 - `def timesUntilZero(f: Int => Int, x: Int): Int =
 if (x==0) 0 else 1 + timesUntilZero(f, f(x))`

Reducing the Function

- Consider this example
 - if $((x*y+2 < 10) == \text{true})$ true else false
- Rewrite once
 - if $(x*y+2 < 10)$ true else false
- Rewrite again to
 - $(x*y+2 < 10)$

Reducing the Function: Example 2

- Can I rewrite the following?
 - `nTimes(doTail[Int], 2, List(3,2,1))`
- `nTimes((xs: List[Int]) => xs.tail, 2, List(3,2,1))`
- `nTimes[[Type]List[Int]](_.tail, 2, List(3,2,1))`

↑
Take any list (can always call tail)

More Abstraction

- Consider this example

- ```
def sillyLottery(f: Int => Int, n: Int) =
 if (f(n)%2 == 0) {
 (x: Int) => x/2
 } else {
 (x: Int) => 2*x+1 }
```

*return (Int => Int)*

- What is the type?

- $((\text{Int} \Rightarrow \text{Int}), \text{Int}) \Rightarrow (\text{Int} \Rightarrow \text{Int})$

- If we give  $\text{Int} \Rightarrow \text{Int}$  and one  $\text{Int}$ , we will get  $\text{Int} \Rightarrow \text{Int}$

- Which we can bind to a variable

- Let's consider `val magic = sillyLottery(x=>3*x-9, 25)`

- What is `magic(21)`?

$$\text{if } (f_{(25)} \% 2) == 0 \{$$

$3 * x - 9$   
 $3 * 25 - 9 = 66$

$(x: \text{Int}) \Rightarrow x/2 \}$

$\rightarrow \text{magic}(x)$

●  $\text{magic}(21) \Rightarrow 21/2 = 10$

# Scala with I/O

- You can import `scala.io.Source` to deal with I/O

# Example

- What if I want to count the number of word in a file?

```
import scala.io.Source
object SimpleWordCount extends App {
 def countPerLine(line: String): Int =
 line.split("\\W+") .length
 val wordsPerLine =
 Source.stdin .getLines.map(countPerLine).toSeq
 val lineCount = wordsPerLine.length
 val wordCount = wordsPerLine.sum
 println(s"lineCount: $lineCount")
 println(s"wordCount: $wordCount")
}
```

**Before We Leave Today**

# In-class Exercise 9

- Finish the remainders of In-class Exercise 8
- Write `def countInRange(xs: List[Int], lo: Int, hi: Int)` which **counts how many numbers in xs are between lo and hi (inclusive)**.
  - You will do it in **three ways** (i.e. write 3 separate solutions):
    - Use filter and length
    - Use map and sum
    - Use foldLeft