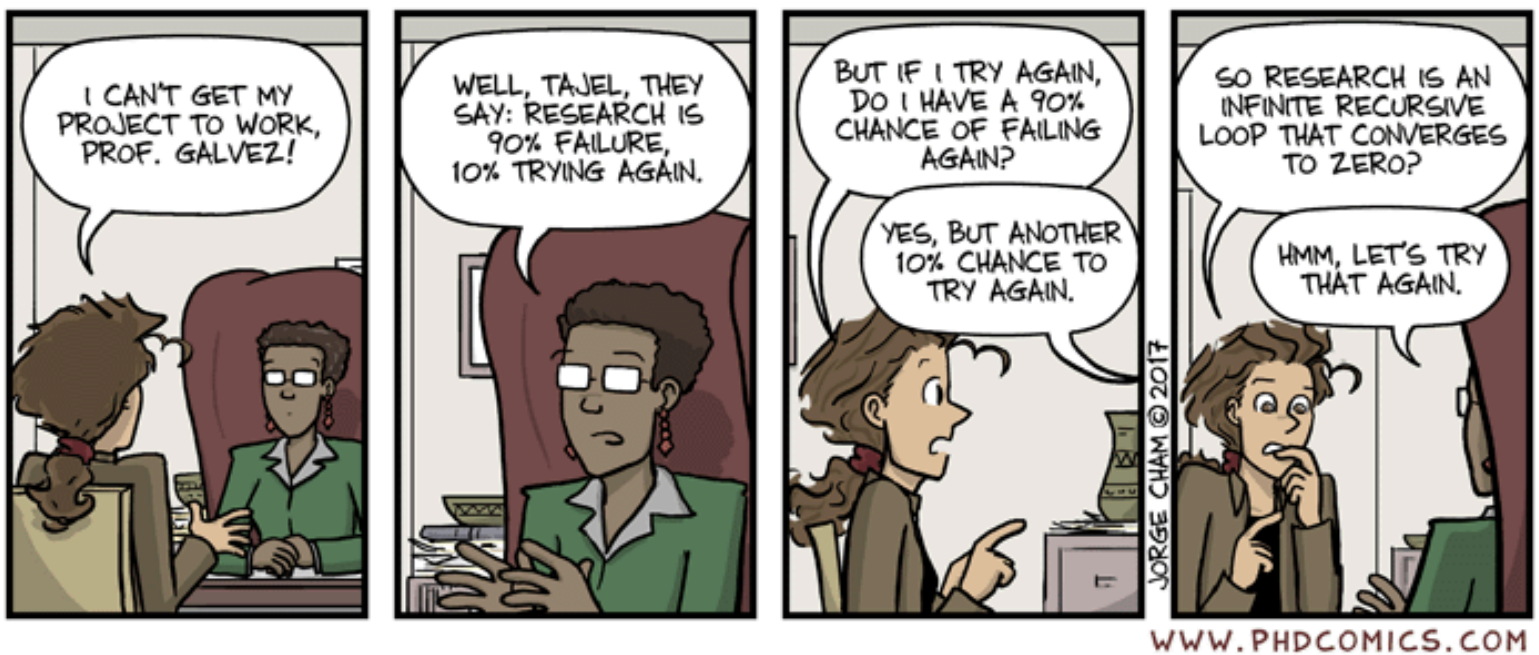
Meeting 05 - Recursion

Bor-Yuh Evan Chang

Tuesday, September 10, 2024

Meeting 05 - Recursion



- In-Class Slides
- In-Class Jupyter
- Book Chapter

Does your neighbor have questions?

Announcements

- HW 1 + Quiz 1 was due ~~Friday 9/6~~ Monday 9/9 6pm
  - How was it? Do you want to go over parts of it?
- Lab 1 due this Friday 9/13 6pm
  - Use GitHub and VS Code: Submit `Lab1.scala`. Just read Jupyter notebook or use it for scratch work.
- "Autograder" is 'sbt test' or Lab1Spec.scala
  - ↳ There's also a GitHub Action
- Looking into Gradescope integration

Announcements

- A proposal: Lawrence (CM) will come to the classroom (ECCR 265) 1:45-2 and 3:15-3:30 for his “administrative office hours” so that you can more easily get your administrative issues resolved.
- A proposal: Would you go to evening review sessions run by your CAs? Think: an extra, optional recitation to review a planned topic (e.g., go over solutions to a past assignment, do extra practice on a difficult topic)

Today

- Triage Your Questions
  - HW1?
- Preview Lab 1 (using coding.csel.io)
- Questions on Binding and Scope: A Scala crash course.
- Parts of Data Types: A Scala crash course.
- Recursion: A Scala crash course.

Your Questions?

- Review:
  - How do *environments* (type or value) relate to *scope*?
  - What is `Nil`, `::`, and `foreach`?

- HW1 Q1 — Piazza @161 — Leaining Activity First
  — Explanatory Matter

- match expressions (Java) HW1

Your Questions?

Factorial

Factorial: Some Evaluation Steps

Factorial: Pattern Matching

Factorial: Preconditions

Factorial: Tail Recursive

Tail-Recursive Factorial: Some Evaluation Steps

Tail-Recursive Factorial

```scala
def factorial(n: Int): Int = {
  require(n >= 0)
  println(s"factorial(n = $n)")
  def loop(acc: Int, n: Int): Int = {
    println(s"-->* loop(acc = $acc, n = $n)")
    n match {
      case 0 => acc
      case _ => loop(acc * n, n - 1)
    }
  }
  val r = loop(1, n)
  println(s"-->* $r")
  r
}
factorial(3)
```

```
factorial(n = 3)
-->* loop(acc = 1, n = 3)
-->* loop(acc = 3, n = 2)
-->* loop(acc = 6, n = 1)
-->* loop(acc = 6, n = 0)
-->* 6
```

Exercise: Exponentiation

Exercise: Tail-Recursive Fibonacci

```scala
def fibonacci(n: Int): Long = {
  require(n >= 0)
  n match {
    case 0 | 1 => 1
    case _ => fibonacci(n - 1) + fibonacci(n - 2)
  }
}
fibonacci(6)
//fibonacci(50)
```

```
defined function fibonacci
res13_1: Long = 13L
```