Φ lang: Whitepaper

Rishi Kothari

Contents

Abstract

Mathematics and Computer Science are deeply intertwined; some universities even offer CS *as* a math course. However,

1 Motivations

Computer Science is arguably one of the most interesting fields of **math**, so why is there such a big disconnect between the math one learns in university and high school CS and the programming language of choice?

Take the example of a simple sum function in math:

$$f(a,b) = \sum_{b}^{a} b, \{\, a, b \,\} \in \mathbb{N}$$

This is an example of Sigma notation, a much-used concept in many parts of math. Taking a look at the equivalent expression in Python,

```
def sum_loop (top_bound, start):
    accumulator = 0
    for i in range(start, top_bound+1):
        accumulator += i
        i+=1
    return accumulator
```

One might see that the two have absolutely nothing in common.

This may not seem like a problem; the programmer just needs to learn programmatical intuition. However, this can pose a challenge for a *mathematician* to learn programming, because of the existing mathematical intuition that needs to be replaced

2 Design

2.1 Syntax

Φ is a mathematical language, and so needs to be designed with rigour in mind. To understand what is mean by this, reading ”The design side of programming language design”, by Tomas Petricek is advised.

2.2 Basics

2.2.1 Ambiguity

Φ is a language that doesn’t allow for any ambiguity. All characters and keywords must have *one* purpose.

For instance, in Swift, the : operator serves two functions: One for type annotation, and the other for assigning values to function parameters:

```
//Use one: Type annotation
var x : String = "Hello World"
```

```
func printString(str : String) -> Void {
    print(str)
}

//Use two: Value assignment
printString(str: x)
```

The use of the : operator in multiple cases creates ambiguity, and

2.3 Relations to Mathematics

2.3.1 Lambda Calculus

2.4 Tokenization