Introduction

What's a programming language?

This might look like a dumb question (and it is) but it is difficult to answer.

Denotational perspective

What do we do when we have dumb questions that are difficult to answer?

We make our lives more difficult by using mathematics!

Let us say that a programming language is any system such that *some subset* of the set of all computable functions is representable in it.

This is actually a very informal definition - what is a "system"? what is "representable"?

From out intuition, we know that intuitive abstract concepts are "represented" in languages somehow. This might involve bit-manipulation magic (for floating point arithmetic), virtual tables (for runtime dispatch), among other things.

So we need a mapping that acts as this representation maker.

A computable function can be converted to the form $f : \mathbb{N} \nrightarrow \mathbb{N}$ (partial functions from and to the natural numbers). This simplifies our lives - all the details about algorithms, trees, graphs, lists, etc. can be encoded within functions of this form.

We say that a *denotation* (notation : $[\![.]\!]$) is a mapping that takes natural numbers to some denotation and computable functions to some denotation such that

$$\forall x \in \mathbb{N}. [\![f]\!][\![x]\!] = [\![f(x)]\!]$$

This is extremely abstract and might be confusing at first.

I'll try to give an intuitive breakdown of the formula above. Let's say we have a computable function f and a natural number n. We can apply this input to our function and get a new natural number $f(x)$. We can now get the denotation of this natural number $[\![f(x)]\!]$. Alternatively, we can get the denotation of f, $[\![f]\!]$ and the denotation of x, $[\![x]\!]$ and apply these two to get $[\![f]\!][\![x]\!]$. Our axiom states that this is *exactly* the denotation of $f(x)$ that we got earlier.