# First Step to Computability

## Is Every Mathematical Function Computable?

## A Restricted and More Precise Question

**Can** *functions* **be written in C Programming Language to compute every** *function* **from the set of** *natural numbers* **to itself :** $[\mathbb{N} \longrightarrow \mathbb{N}]$?

# Functions from $\mathbb{N}$ to Itself

$$\mathbb{N} = \{0, 1, 2, 3, \cdots\}$$

.

- *Constant functions* : $cons_k(x) = k$, for all $x \in \mathbb{N}$, where $k \in \mathbb{N}$. There are infintely many of them.
- The *successor* function : $s(x) = x + 1$, for all $x \in \mathbb{N}$.
- The *predecessor* function :

$$p(x) = \begin{cases} 0 & \text{if } x = 0, \\ x - 1 & \text{if } x > 0. \end{cases}$$

## Functions from $\mathbb{N}$ to Itself (*cont.*)

- The *factorial* function : $x!$, for all $x \in \mathbb{N}$.
- The *fibonacci* function : $\mathit{fib}(x)$, for all $x \in \mathbb{N}$.
- *Power functions* : $pow_n(x) = x^n$, for all $x \in \mathbb{N}$, where $n \in \mathbb{N}$. There are infinitely many of them.

**There are *infinite* number of such functions.**

# C Language Function : *A Serious Problem*

- A variable in the C programming language cannot store a natural number of very large size due to the finite word size of the CPU.
- Finite memory of a computer cannot store a natural number of very large size. Also it cannot store a very long computation (e.g. a very large stack size).

*Conclusion* : **No function from the set of natural numbers to itself can be computed by a *real* computer.**

## Abstraction for Better Understanding

**Sometimes infinite objects are good approximations of finite objects -** ***Yuri Gurevich*****.**

## An Ideal Language and Computer

- Let the variable of a new programming language **C**$^*$ (which is otherwise similar to C programming language) can store a natural number of any arbitrary size.
- Also assume that the memory of our computer is not finite (This immediately will come in conflict with the standard Physics. Why?), but can store any arbitrarily large but finite data and computation (potentially infinite).

## Question Restated

**Can** *functions* **be written in** $C^*$ **Programming Language to compute every** *function* **from the set of** *natural numbers* **to itself :** $[\mathbb{N} \longrightarrow \mathbb{N}]$**?**

# $C^*$ Functions

- *Constant* function :

  `int cons5(int x) {return 5;}`

- The *successor* function :

  `int succ(int x) { return x+1;}`

- The *factorial* function :

  ```
  int fact(int x) { if x=0 return 1; else
  return x*fact(x-1) }
  ```

## Infinitely Many $C^*$ Functions

There can be infinitely many $C^*$ functions to compute a function from $\mathbb{N}$ to $\mathbb{N}$. Consider the function $cons_5(x)$. Following are a few $C^*$ functions that compute them.

- `int cons5_0(int x) {x = 0; return 5;}`
- `int cons5_1(int x) {x = 1; return 5;}`
- `int cons5_2(int x) {x = 2; return 5;}`
- $\cdots$

## Arithmetic Expressions

Infiniteily many $C^*$ functions from $\mathbb{N}$ to itself may be compared with the arithmetic expressions. An infinite number of such expressions may have the same value e.g.

$$5, 1 + 4, 2 + 3, 3 * 2 - 1, 10/2, \cdots$$

## How to Compare?

- There are infinitely many functions from the set of natural numbers to itself. Let the collection be $\mathcal{F}_{\mathbb{N}}$.
- Infinitely many functions can also be written in $C^*$ programming language to compute functions from the set of natural numbers to itself. Let the collection be $\mathcal{F}_{C^*}$.
- How do we compare these two collections, $\mathcal{F}_{\mathbb{N}}$ and $\mathcal{F}_{C^*}$?
- How do we know whether for every $f \in \mathcal{F}_{\mathbb{N}}$, there is a program $C^*_f \in \mathcal{F}_{C^*}$, that computes $f$?

## George Cantor Proved ···

- There are different species of infinite sets (in fact there are infinitely many of them) and they can be compared.
- Following Cantor[a] we shall examine the question we have raised.
- But to do that we have to be familier with some of the mathematical machinery.

[a] German Mathematecian, 1845 - 1918.