Computation; Notes

March 8, 2020

Contents

1 Computable Functions

1.1 Basic Concepts

§ Partial Functions A *partial function* generalizes the usual definition of function, the idea being that this kind of function is potentially not defined on the entire domain. Formally:

Definition 1.1. A *partial funciton* f from X to Y (written as $f : X \nrightarrow Y$) is a triple (g, X, Y) such that $X' \subseteq X$ and $g : X' \to Y$ is a function. Furthermore:

- The *domain* of f is denoted by $\mathsf{Dom}(f)$ and is equal to X';
- If $\mathsf{Dom}(f) = X$ then f is a *total function*[1];
- If $x \in (X \setminus \mathsf{Dom}\, f)$ then $f(x)$ is said to be *undefined*, denoted $f(x) = -$, on the other hand, if $x \in \mathsf{Dom}\, f$ then we write $f(x) = y$ with $y = g(x)$ and say that f is *defined* at x.

Henceforth the word "function" will always mean "partial function." As an example, consider the (partial) function:

$$\begin{aligned} f : \mathbb{N}_0 &\nrightarrow \mathbb{N}_0 \\ n &\mapsto \sqrt{n}. \end{aligned}$$

If $n \in \mathbb{N}_0$ is not a perfect square, then $f(n)$ is undefined.

§ Lambda Notation

1.2 What is a computable function?

§ Informal Discussion An *algorithm* is a finite sequence of discrete mechanical instructions. A numerical function is *effectively computable* (or simply *computable*) if an algorithm exists that can be used to calculate the value of the function for any given input from its domain.

§ The Unlimited Register Machine The *unlimited register machine* has an infinite number of *registers* labelled $R_1, R_2, \ldots$, each containing a natural number, if R_i is a register then r_i is the number it contains. It can be represented as follows

R_1	R_2	R_3	R_4	R_5	R_6	R_7	$\cdots$
r_1	r_2	r_3	r_4	r_5	r_6	r_7	$\cdots$

The contents of a register may be altered by the URM in response to certain *instructions*.

[1]Total functions and functions are equivalent.

§ Instruction set

Name of Instruction	Instruction	URM response
Zero	`Z(`n`)`	$r_n \leftarrow 0$
Successor	`S(`n`)`	$r_n \leftarrow r_n + 1$
Transfer	`T(`m`,`n`)`	$r_n \leftarrow r_m$
Jump	`J(`m`,`n`,`q`)`	if $r_m = r_n$ then jump to q-th instruction; otherwise proceed to next instruction.