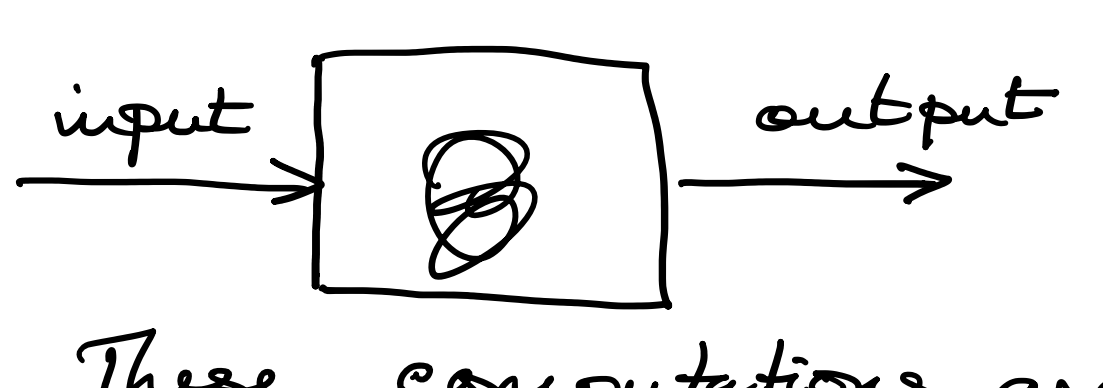
Computability Theory

Thursday, March 18, 2021 11:32 AM

Strings, numbers, data structures, lists etc can all be encoded one in the other: so it does not matter which data type I am using.

I will use numbers $\mathbb{N} = \{0, 1, 2, \dots\}$ or strings without remark.



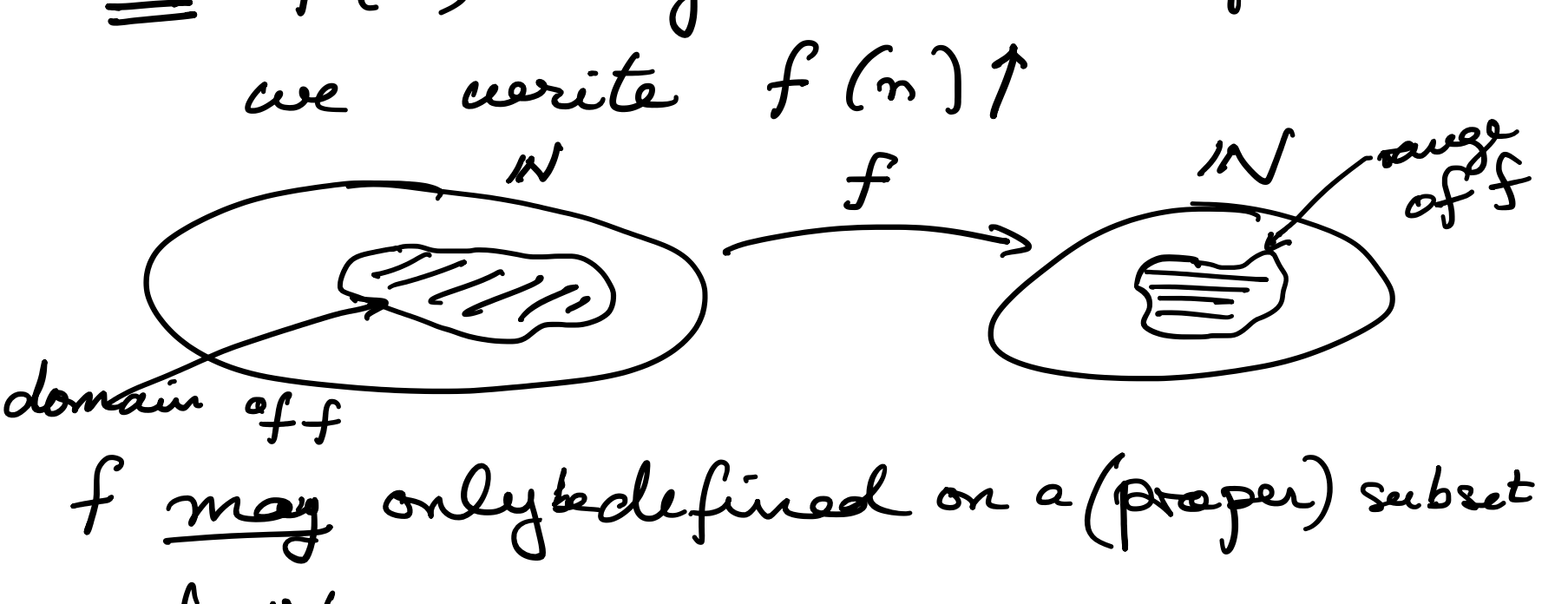
These computations are modelled as <u>functions</u>. <u>BUT</u> we have seen some computations <u>may</u> not halt. So we have to use <u>partial</u> <u>f^{ns}</u>

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

Given input $n \in \mathbb{N}$

$f(n)$ may halt and produce output m; we write $f(n) = m$

<u>or</u> $f(n)$ may not be defined: we write $f(n)\uparrow$



f <u>may</u> only be defined on a (proper) subset of $\mathbb{N}$

$dom(f) = \{n \in \mathbb{N} \mid f(n) \text{ is defined}\}$

$f(n)\downarrow$ means $f(n)$ is defined.

When we write $f = g$ we mean

$\forall n \; f(n)\downarrow \Leftrightarrow g(n)\downarrow$ and

$\forall n \in dom(f) \quad f(n) = g(n)$.

What should I use to describe algs?

I will use natural language pseudocode.

A set X of natural numbers is called <u>computable</u> or <u>decidable</u> if its characteristic function is <u>total</u> <u>computable</u>.

Given X; its characteristic f^n

$$\chi_X(n) = \begin{cases} 1 & \text{if } n \in X \\ 0 & \text{if } n \notin X \end{cases}$$

A function is <u>computable</u> if there is an algorithm to compute it.

A function f is <u>total</u> if $dom(f) = \mathbb{N}$

A function is <u>total</u> <u>computable</u> if the algorithm to compute it <u>halts</u> on <u>every</u> input.

A set is decidable means we have an algorithm that <u>always</u> terminates and answers the membership question.

<u>Ex</u> (1) The set of odd numbers

(2) The set of prime numbers

Given a TM M we write $\langle M \rangle$ for the encoding of M as an integer.

$\langle M, x \rangle$: encoding of M and input string x as an integer.

Ex $S = \{\langle M, x \rangle \mid M \text{ halts on input } x\}$

We cannot have an algorithm to decide membership in S.

We call this set <u>H_{TM}</u>

$$A_{TM} = \{\langle M, x \rangle \mid M \text{ accepts } x\}$$

H_{TM} is <u>not</u> decidable but it is <u>semi</u> <u>decidable</u>.

There is an algorithm that will answer <u>YES</u> <u>every</u> <u>time</u> if $n \in H_{TM}$ but if $n \notin H_{TM}$ it may or may not answer.

<u>THM</u> An <u>infinite</u> set X of natural numbers is <u>computable</u> (decidable) if and only if it is the range of some <u>total</u> <u>non-decreasing</u> <u>computable</u> function f.

<u>PROOF</u> Suppose we have an f as described and A is an algorithm that computes it. I will give you a <u>decision procedure</u> for X. Given $x \in \mathbb{N}$, Run A with $0, 1, 2, \dots$ as inputs and check if the output is ever x. If the output is $< x$, move on to the next number; if output $= x$ STOP and say "YES"; if output $> x$ STOP and say "NO".

Suppose we know X is decidable and that B is a decision procedure for membership in X. We define f as follows. We run B on $0, 1, 2, \dots$ every time we find a number in X we store it in a list. Then define $f(n)$ to be the n^{th} element on this list.

<u>DEF</u> A set $X \subseteq \mathbb{N}$ is called <u>computably</u> <u>enumerable</u> (CE) if there is an algorithm that lists all the members of X in some order; not necessarily increasing order.

<u>REMARKS</u>: Producing the whole list is an infinite process, but any given member of the list is produced at some finite stage.

<u>THM</u> A set X is CE iff any of the following equivalent conditions hold:

(i) X is the <u>domain</u> of a computable function

(ii) the semi-characteristic function of X

$$\delta_X(n) = \begin{cases} 1 & \text{if } n \in X \\ \text{undefined} & \text{if } n \notin X \end{cases}$$

is computable

(iii) X is the ~~range~~ of a computable f^n.

<u>PROOF</u> (ii) $\Rightarrow$ (i) immediately.

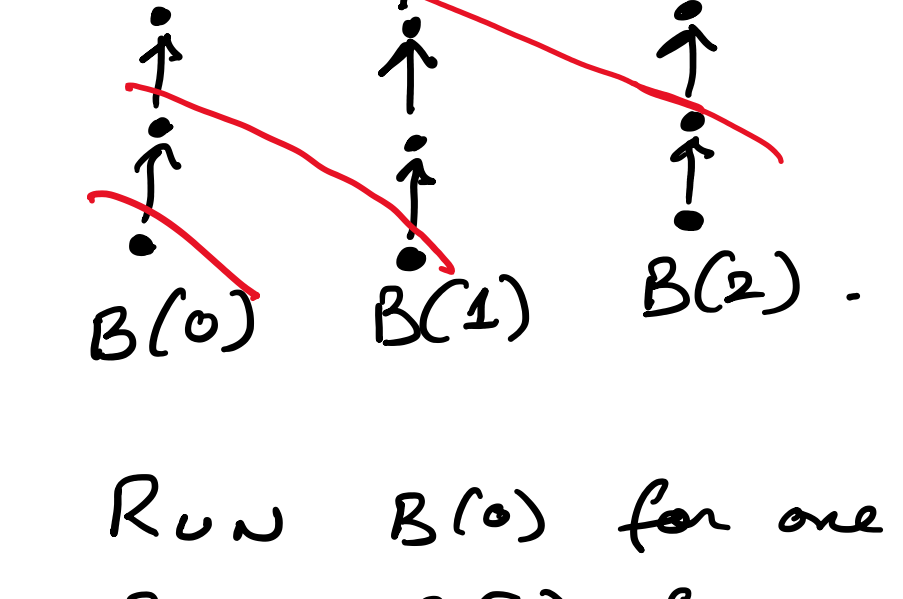
(CE) $\Rightarrow$ (ii) Suppose A enumerates X. To compute $\delta_X(n)$; run A and wait for it to produce n. It may or may not do so; if A does produce n we set $\delta_X(n) = 1$ otherwise we wait <u>forever</u>.

(i) $\Rightarrow$ (CE) Suppose B is an algorithm to compute f and $dom(f) = X$.

Idea (BAD): Run B on $0, 1, 2, 3, \dots$ whenever B halts on an n we output n as our enumeration process.

BUT this will not work because perhaps $B(0)$ does not halt.

NEW IMPROVED IDEA: DOVETAILING



RUN B(0) for one step; then store the state
RUN B(1) for one step; then store
RUN B(0) for 2 steps, then store
RUN B(1) for 2 steps, then store
RUN B(2) for 2 steps then store
RUN B(0) . . . 3 . . . . . .
RUN B(1) . . . 3 . . . . . .
RUN B(2) . . . 3 . . . . .
RUN B(3) . . . 3 . . . . . .

For any $n \in dom(f)$ this procedure will eventually run long enough to find out that B terminates on n.

This is an enumerator for X

<u>THM</u> The union of 2 CE sets is CE

The intersection of 2 CE sets is CE.

<u>THM</u> (POST)

(a) If X is a computable (decidable) set then X is CE. [TRIVIAL]

(b) If X is CE and $\overline{X}$ is also CE then X is computable (decidable).

PROOF (b): Run the enumerators for X and $\overline{X}$ in parallel; any $n \in \mathbb{N}$ must lie in one of these sets so if they are both CE one of them will eventually enumerate it.

<u>THM</u> $X \subseteq \mathbb{N}$ is CE iff $\exists$ a <u>computable</u> set Y of <u>pairs</u> of natural numbers s.t. $\forall x, \; x \in X \Leftrightarrow \exists y \in \mathbb{N} \; (\langle x, y \rangle \in Y)$.

<u>PROOF</u> Suppose X is CE, A enumerates it we define Y as follows

$$Y = \{\langle x, n \rangle \mid A \text{ enumerates } x \text{ within } n \text{ steps}\}$$

Clearly Y is <u>decidable</u> but clearly

$$\forall x, \; x \in X \Leftrightarrow \exists n \in \mathbb{N}. \; (\langle x, n \rangle \in Y).$$
