Countable and uncountable sets

Def: A set S is **countable** (or enumerable) if it can be matched 1 to 1 w/ some subset of the ~~positive integers~~ nonnegative integers.

11/12/18

Examples:

1) The nonnegative integers

2) ~~Finite seq~~ (finite) strings in the alphabet $\{0,1\}$.

A way of listing all of them:

λ (0), 0 (1), 1 (2), 00 (3), 01 (4), 10 (5), 11 (6), 000 (7), 001 (8), 010 (9),
011, 100, 101, 110, 111, 0000, ...

and we do get to every string.

match w/ nonneg integ.

(or any other alphabet)

Def: A set S is uncountable if it cannot be matched w/ any subset of the nonnegative integers.

Thm: The set of infinite sequences in $\{0,1\}$ is uncountable.

Pf: Suppose for contradiction this set is countable: Then I can list the sequences:

1	0 0 0 0 0 0 0
2	1 0 0 0 0 0 0
3	0 1 0 0 . . . . .
4	1 0 1 0 0 . . . . .

Here is a sequence A not on the list:

In the i-th spot of the sequence,

$$A_i = \begin{cases} 0 & \text{if the } i\text{-th entry on } i\text{-th sequence is a 1} \\ 1 & \text{if the } i\text{-th entry on } i\text{-th sequence is a 0} \end{cases}$$

$A = 1\ 1\ 1\ 1$ doesn't match these entries

This sequence ~~is~~ A is <u>not</u> on the list since it differs from the n-th sequence at the n-th location.

This is a contradiction, since I was supposed to have a list of all sequences.

<u>Cor</u>: The set of ~~langs~~ all languages (on $\{0,1\}$) is uncountable.

or your favorite alphabet

<u>Pf</u>: I can identify every language (on $\{0,1\}$) with a sequence: the sequence for L has a $\begin{cases} 1 & \text{in i-th spot if i-th string} \in L \\ 0 & \text{" " i-th string} \notin L \end{cases}$

Turing-decidable/acceptable/enumerable languages

Def: A language L is **Turing-decidable** (or **recursive**) if there is a TM M that **halts on every input** that **accepts** L.

Def: A language L is **Turing-acceptable** if there is a TM M that accepts L (i.e. it halts in a final state for every string in L, but for strings not in L, it might halt in a non-final state or run forever)

Def: A language L is **Turing-enumerable** (~~or~~ or **recursively enumerable**) if there is a TM M_w / a special state q_e such that,

$$L = \{ w \mid (w, q_e, u) \text{ for some } u \}$$

~~is~~ is an instaneous description of M at some point when M is started w/ empty input.

So M starts w/ empty input
writes something in L - goes to state q_e to say I've written something (and not in middle of writing something) then writes something else in L, etc.

If L is infinite, M would run forever.

Prop: ~~Turing-enumerable languages are countable~~
The set of strings in a Tur-enum lang. is countable

Pf 1: Duh - the set of strings in any language is countable

Pf 2: The TM M gives a procedure for assigning a nonneg integ to every string in L - number them in the order M "outputs" them.

Thm: Every recursive (Tur-decidable) lang. is recursively enumerable (Tur-enum)

Pf: We can construct a TM that goes through every string (i.e. say in the order given 30 min ago) and goes into q_e (the stamp of approval state) ~~and~~ if the string is in the language (i.e. is accepted by the original TM that made it Tur-dec.)

Thm: Every Turing-acceptable language is Turing-enumerable

This might be surprising b/c the previous proof doesn't work. you might try a string not in L and the machine won't halt, so you never get to checking the next string.

Pf: I can get around this problem by for doing 1 step for string 1, then
2 steps for strings 1 & 2
⋮
i steps for strings 1 through i

always killing the computation if it hasn't halted by then.