<u>Definitions</u>: Recursive, recursively acceptable, recursive enumerable.

<u>Def</u>: A language L is <u>recursive</u> (aka <u>Turing-decidable</u>, recursively decidable) if there is a TM M that

1) halts eventually on every input
2) accepts all strings in L, rejects all strings not in L.

<u>Def</u>: A language L is recursively acceptable (aka Turing-acceptable) if there is a TM that

1) halts ~~and~~ on and accepts every string in L - doesn't do this to any string not in L.

Goal for today: A language L is in Turing-acceptable if and only if it is Turing-enumerable.

Easy implication: Every TE language is TA. Why?

If you have a TM M that enumerates L, we can construct a TM $\hat{M}$ that accepts L by having $\hat{M}$ compare its input to everything "printed" by M. If they match $\hat{M}$ halts and ~~says~~ accepts. If they don't match $\hat{M}$ has M continue printing.

Def: A language L is <u>recursively enumerable</u> (aka Turing-enumerable) if there is a TM M that has a special state p (the "print state") so that, ~~for every string~~ w ~~in L~~, when M is run starting from blank input, ~~eventually,~~

$$\sqcup\, w \,\#\, \text{something}$$

eventually appears on the tape while M is in state p if and only if w is in L.

(i.e. you can write a computer program that prints every string in L (and nothing else)

This TM M is said to <u>enumerate</u> L.

Why is every TA language TE?

~~We can~~ Given a TM $\hat{M}$ that accepts L (i.e. accepts everything in L but might run forever on input not in L), we could try to build a TM M that enumerates L by feeding every string (say in the order λ, a, b, aa, ab, bb, aaa, ...) into $\hat{M}$ one at a time, and "printing" ~~the~~ a string when it is accepted.

This doesn't work b/c, if $\hat{M}$ runs forever on λ, then we never get around to testing "a", so we might not ~~pr~~ ever print a even if it is in L.

Another option: cut off every computation after 1,000,000 steps, and print out only strings accepted before 1,000,000 steps.

This doesn't work b/c if a string takes 2,000,000 steps to be accepted, it's never printed.

Clever trick: Try λ for 1 minute,

then λ, a for 2 minutes (each)

λ, a, b " 3 "

~~th~~ first 4 strings " 4 "

" 5 " " 5 "

If the m-th string takes n minutes to accept, then we will find this out (and have M print it) after $\max(m,n)$ minutes.

So anything that $\hat{M}$ accepts, M will eventually print out.