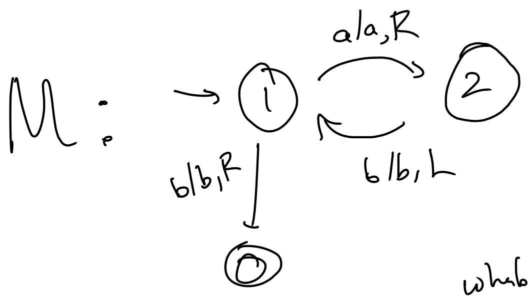
The limits of TMs

A language which is accepted by a TM is **recursively enumerable**

$L(M)$ is the set of strings accepted by some TM M.

M:



$L(M) = \{ bx \mid x \in \{a,b\}^* \}$

whatabout ab?

$$B1abB \mapsto Ba2bB \mapsto B1abB$$

ab makes an infinite loop.

aa is rejected (the TM halts in a non-accepting state)

"accept" $\to L(M) = \{ bx \mid x \in \{a,b\}^* \}$

"reject" $\to R(M) = \{ \varepsilon, a \} \cup \{ aax \mid x \in \{a,b\}^* \}$

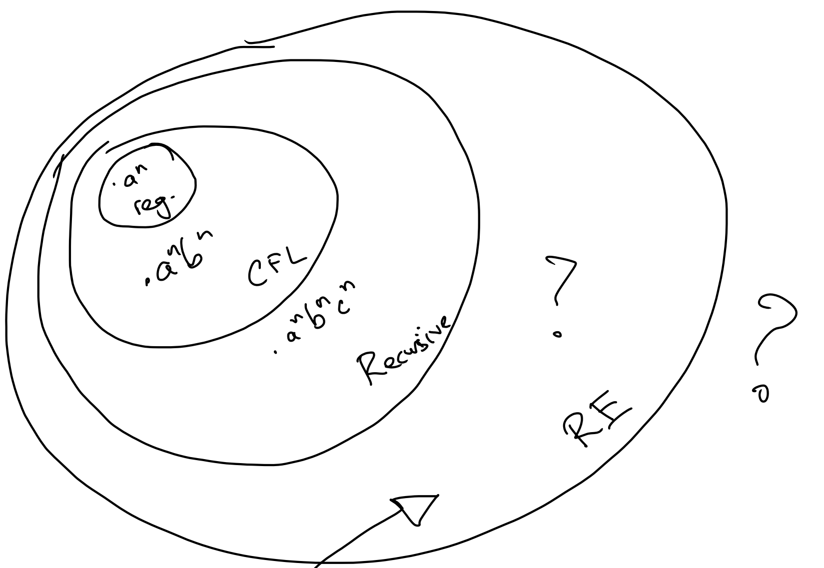
"loops" $\to F(M) = \{ abx \mid x \in \{a,b\}^* \}$

Real-life useful machines are intended to have $F(M) = \emptyset$.

If M is a TM with $F(M) = \emptyset$, we say M is a <u>total</u> TM.

real world comps. are meant to be like this.

If L is the lang. accepted by a <u>total</u> <u>TM</u>, then L is <u>recursive</u>.



can't do on any TM.

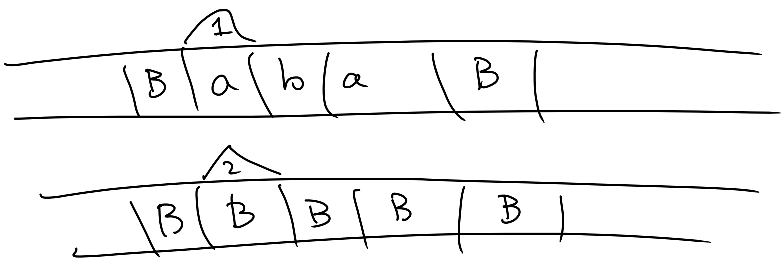
RE, but not recursive: could be accepted by a TM, but it would allow loops.

Some simple TM variations

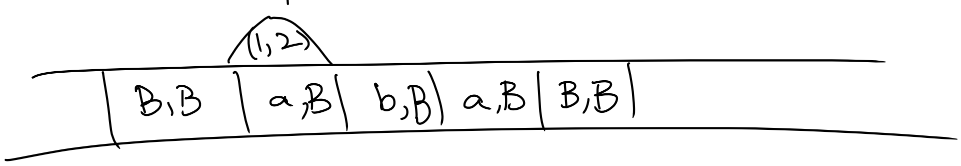
What would we get from a TM with 2 tapes?

A 2-tape TM could be simulated by a 1 taper.

Doing products:



Combine it in pairs:



The diagram of states becomes bigger, but it'll still work.

This product machine will have the same

$L(M)$
$R(M)$ as the original 2-tape machine.
$F(M)$

Any n-tape TM can be simulated by a 1-tape TM.

"The Monkey Multiplier"

Another variation:

Allowing the head to jump several (or no) positions in each step.

Say our TM can "Stay"

$$\bigcirc \xrightarrow{x/y,\,S} \bigcirc$$

we can simulate it with a normal TM:

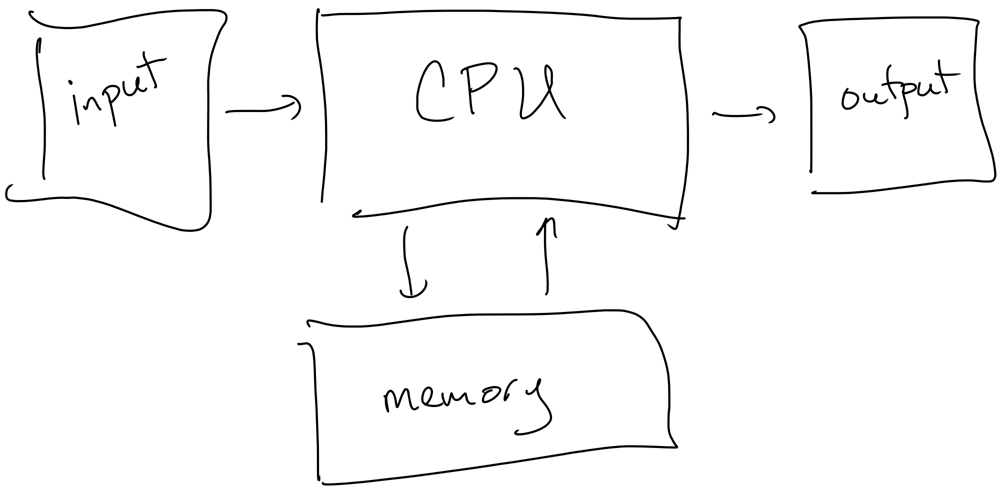
$$\bigcirc \xrightarrow{x/y,\,R} \bigcirc \xrightarrow[\ ]{\substack{B/B,\,L \\ b/b,\,L \\ a/a,\,L}} \bigcirc$$

Any kind of multi-tape, jumping, etc.

any "reasonable" improvement to a TM results in an equivalent machine.

Real computers are fancy TMs,
but have the same capabilities.

Most are based on the Von Neumann architecture:



A Von N. machine is equivalent to a TM,
but more useful.

Different computers can emulate one another.

my PC can emulate NES.

PCs can emulate TMs

Turing imagined a "Universal" TM which
would simulate other TMs.

This machine $\mathcal{U}$ would take input of

$M \# x$

Some binary encoding of a TM diagram

input string

Then $\mathcal{U}$ will run and result in the same output as M run with input x.

The simplest computation which cannot be done on a total TM is "the halting problem"