Programming Turing Machines

11/2

$L = \{a^n b^n \mid n \in \mathbb{Z}\}$

Program a TM to accept L

Convention: The ~~input~~ tape of the TM starts with the read head at the beginning of the input. Everything on the tape other than the input is blank symbols:

i.e. the instantaneous description at the beginning is

(λ, q_0, w)

Idea for the program -

match an a & a b; mark the matched a's & b's w/ some other symbol - If we're all matched at the end, accept, otherwise, don't accept.

~~Idea~~:

$\Gamma = \{a, b, x, y, \#, \square\}$

Transition function:

$(q_0, a) \longrightarrow (q_1, x, R)$

$(q_1, a) \longrightarrow (q_1, a, R)$

$(q_1, b) \longrightarrow (q_2, y, L)$

~~$(q_1, \square) \longrightarrow (q_R, \square, R)$~~

$(q_2, a) \longrightarrow (q_2, a, L)$

$(q_2, x) \longrightarrow (q_0, x, R)$

only if the string starts w/ b → $(q_0, b) \longrightarrow (q_N, b, L)$

~~$(q_0, \square) \longrightarrow (q_R, \square, R)$~~

$(q_1, y) \longrightarrow (q_3, y, R)$

$(q_3, a) \longrightarrow (q_N, a, R)$

$(q_3, y) \longrightarrow (q_3, y, R)$

$(q_3, b) \longrightarrow (q_2, y, L)$

$(q_2, y) \longrightarrow (q_2, y, L)$

q_1 is moving right to find a b state

q_2 is moving left after matching

q_3 is same, but after hitting a y.

$q_N = N_0$

$(q_3, \sqcup) \longrightarrow (q_N, \sqcup, R)$

$(q_N, \text{anything}) \longrightarrow$ ~~$?f$~~

$(q_0, y) \longrightarrow (q_4, y, R)$

this happens when all a's have been marked

$(q_4, y) \longrightarrow (q_4, y, R)$

$(q_4, \sqcup) \longrightarrow (q_Y, \sqcup, R)$

$(q_4, a) \longrightarrow (q_N, a, R)$

$(q_4, b) \longrightarrow (q_N, a, R)$

$(q_Y, \text{anything}) \longrightarrow$ ~~$?f$~~

q_4 is moving to right at end to check no extra b's (or a's after b's)

q_Y = Yes

alternative:

$(q_4, \sqcup) \longrightarrow ?f$ and q_4 is final

$(q_0, \sqcup) \longrightarrow (q_Y, \sqcup, R)$

$(q_1, \sqcup) \longrightarrow (q_N, \sqcup, R)$

$L = \{a^n b^n c^n \mid n \in \mathbb{Z}\}$

We can do this by matching a's b's c's simultaneously, marking matched letters w/ x's, y's, & z's.

(Writing this out takes about twice as much space as the previous TM)

Note - with TMs, we can go back and forth on our input, and we have arbitrary access to memory. (though not "random access")

Next time:

Arithmetic - we need a way to represent numbers

Binary is more compact, but harder to program

We'll program in unary - a # is rep. by that number of 1's.