Turing Machines

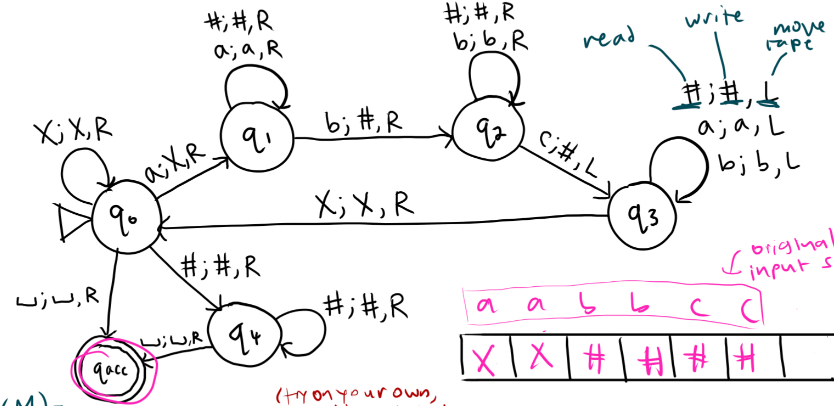
$$M = (\underbrace{Q}_{\text{set of states}}, \underbrace{\Sigma}_{\text{input alphabet}}, \underbrace{\Gamma}_{\text{tape alphabet}}, \underbrace{\delta}_{\text{transition function}}, \underbrace{q_0}_{\text{start state}}, \underbrace{q_{accept}}_{\text{accept state}}, \underbrace{q_{reject}}_{\text{reject state}})$$

$\delta((q_1, b))$

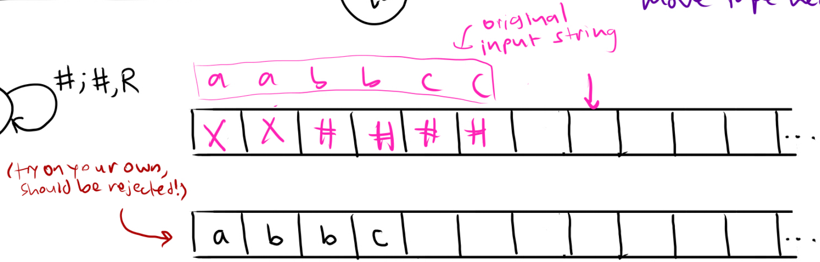
δ

2 inputs:
- state to start at
- symbol to read from tape

3 outputs:
- state to transition to
- symbol to write to the tape
- direction to move tape head



$L(M) = \{a^n b^n c^n \mid n \geq 0\}$

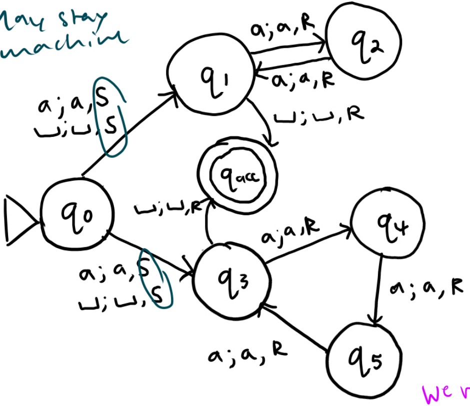


May-Stay Machines

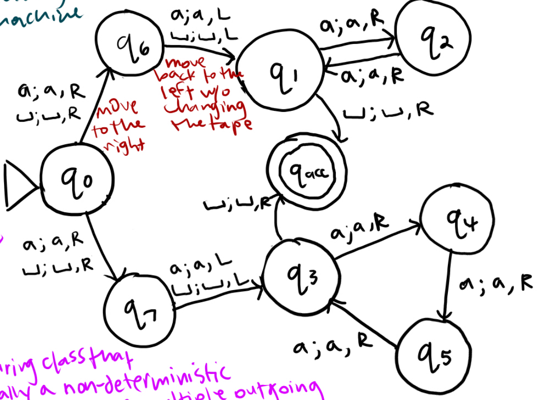
→ are equally as expressive as Turing machines

The tape head can move Left, Right, or Stay. See May 4 lecture for more details.

May stay machine



Equivalent Turing machine



Note:
We realized during class that this is actually a non-deterministic Turing machine! (There are multiple outgoing transitions from q0 for a). However, we can also show that non-deterministic TMs are also equally as expressive as deterministic TMs.

while running M, save a copy of the input using a →

Multi-tape Turing machine

Implementation-level description:

There are 2 tapes: the first has the input, and the second is blank. As M runs, move the heads for both tapes together. If the second tape reads a blank symbol, write the symbol that the first tape reads.

X	X	#	#	#	#							...

a	a	b	b	c	c							...

Note: if M's tape alphabet already includes the symbol #, choose a different symbol not in the tape alphabet. We chose ☆ in this example.

To simulate this on a single-tape Turing machine:

Move right (without changing the tape) until a blank symbol is read. Write a # symbol to indicate the end of the input. Move the tape head back to the beginning. Each time M reads a new input symbol, write it to the next blank spot after the #.

X	X	#	#	#	#	☆	a	a	b	b	c	c										...