



THE UNIVERSITY
OF QUEENSLAND
AUSTRALIA

CREATE CHANGE

Tutorial 3 – Turing Machines

COMP2048

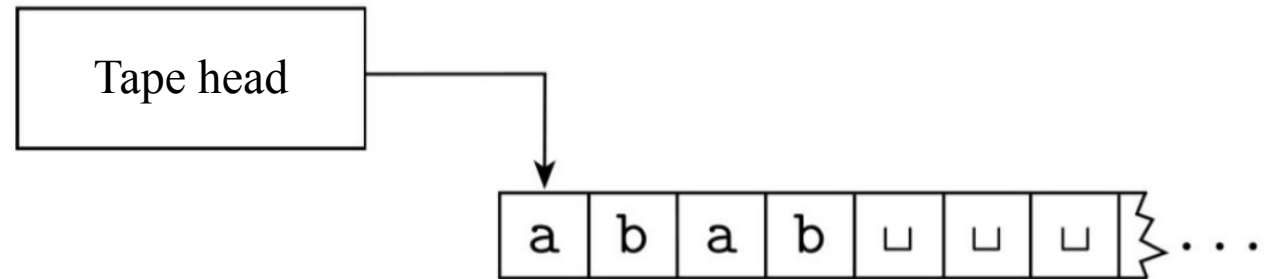
Why didn't we like FSMs too much?

- Finite number of states → cannot solve many problems
- Finite memory (only for current state) → cannot work with unbounded inputs
- Finite memory → no keeping track of past inputs
- Not very powerful

What is a Turing Machine?

- Abstract model of computation
- Proposed by Alan Turing in 1936
- An extension of FSMs with unlimited, unrestricted memory
- Can do anything a real computer can do

Turing Machine Schematic



- Infinite tape
- Tape head can read and write symbols (only one at a time)
- Tape head can move left or right
- Tape initially contains input string and everything else is blank
- Halts as soon as it reaches an *accept* or *reject* state

Turing Machine Formal Definition

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where Q, Σ, Γ are finite sets, and

1. Q is the set of states,
2. Σ is the input alphabet not containing the blank symbol \sqcup
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function,
5. $q_0 \in Q$ is the start state
6. $q_{accept} \in Q$ is the accept state, and
7. $q_{reject} \in Q$ is the reject state, where $q_{accept} \neq q_{reject}$

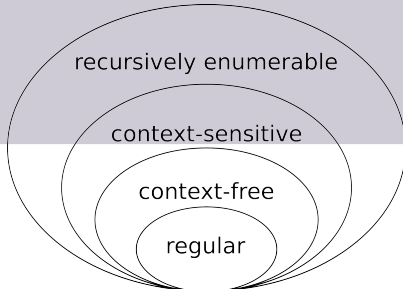
Can we define computation with this?



Define machine configuration

Problem 1

List all the differences between finite state machines and computers we have today with the concept of Turing machines.

FSM	TM	PC
Hypothetical	Hypothetical	Physical
States and transitions only (No memory)	Infinite memory (tape)	Finite memory
transition rules $\delta : \{ S \} \times \{ A \} \rightarrow \{ S \}$ (Read \rightarrow Head: R)	$\delta : \{ S \} \times \{ A \}_{\text{TAPE}} \rightarrow \{ S \} \times \{ A \}_{\text{TAPE}} \times \{ L, R \}$ (Read \rightarrow Write \rightarrow Head: L/R)	Read/write
Limited run time	Unlimited run time	Limited run time?
Describes the class of regular languages	Accepts a much larger class of languages  <div> $L = \{ w w \text{ describes a terminating Turing machine} \}$ $L = \{ a^n b^n c^n n > 0 \}$ $L = \{ a^n b^n n > 0 \}$ $L = \{ a^n n \geq 0 \}$ </div>	Any n, as long as it does not run out of memory?

Problem 2

Write down the formal description for the Turing machine that decides the language $AA = \{w\#w \mid w \in \{0,1\}^*\}$.
You may use a transition table or diagram to describe the transition rules for the machine.

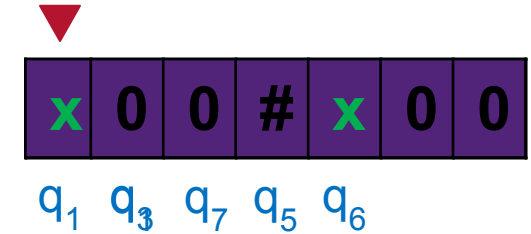
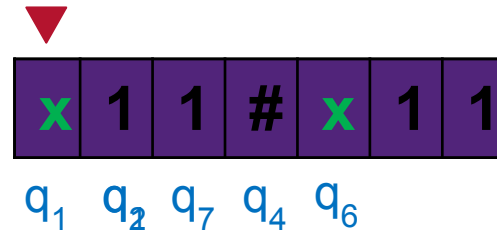
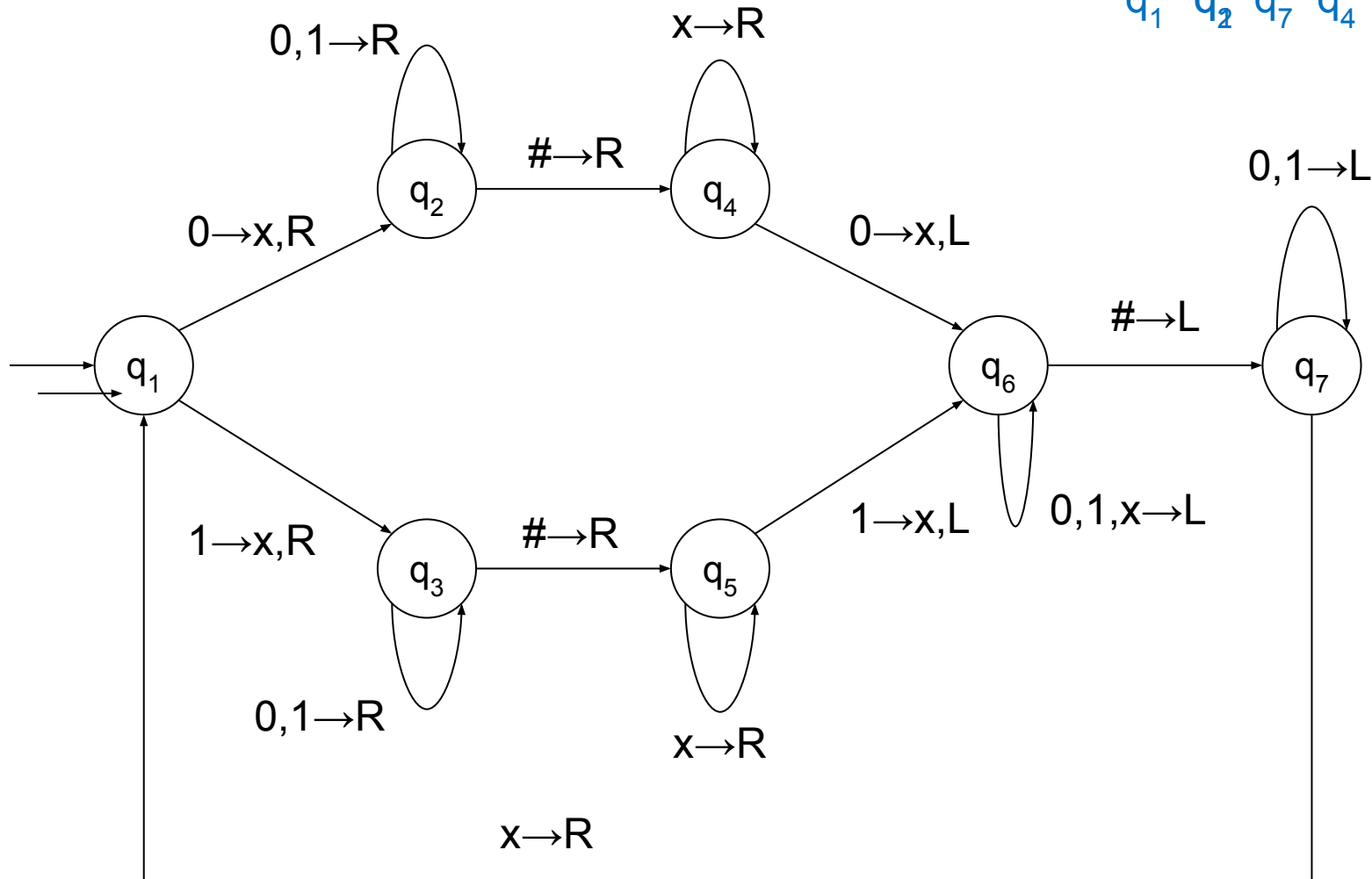
* is like a 'closure' symbol

$w \in \{0,1\}^* \rightarrow$ Space of finite strings in the alphabet $\{0,1\}$ including the empty string

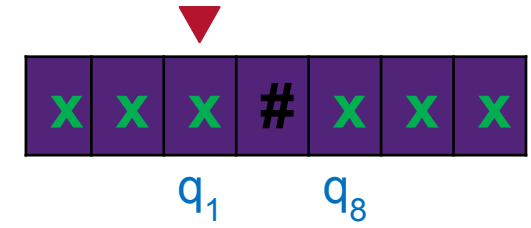
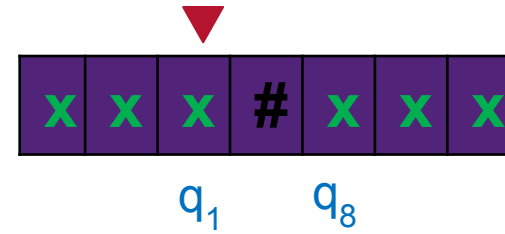
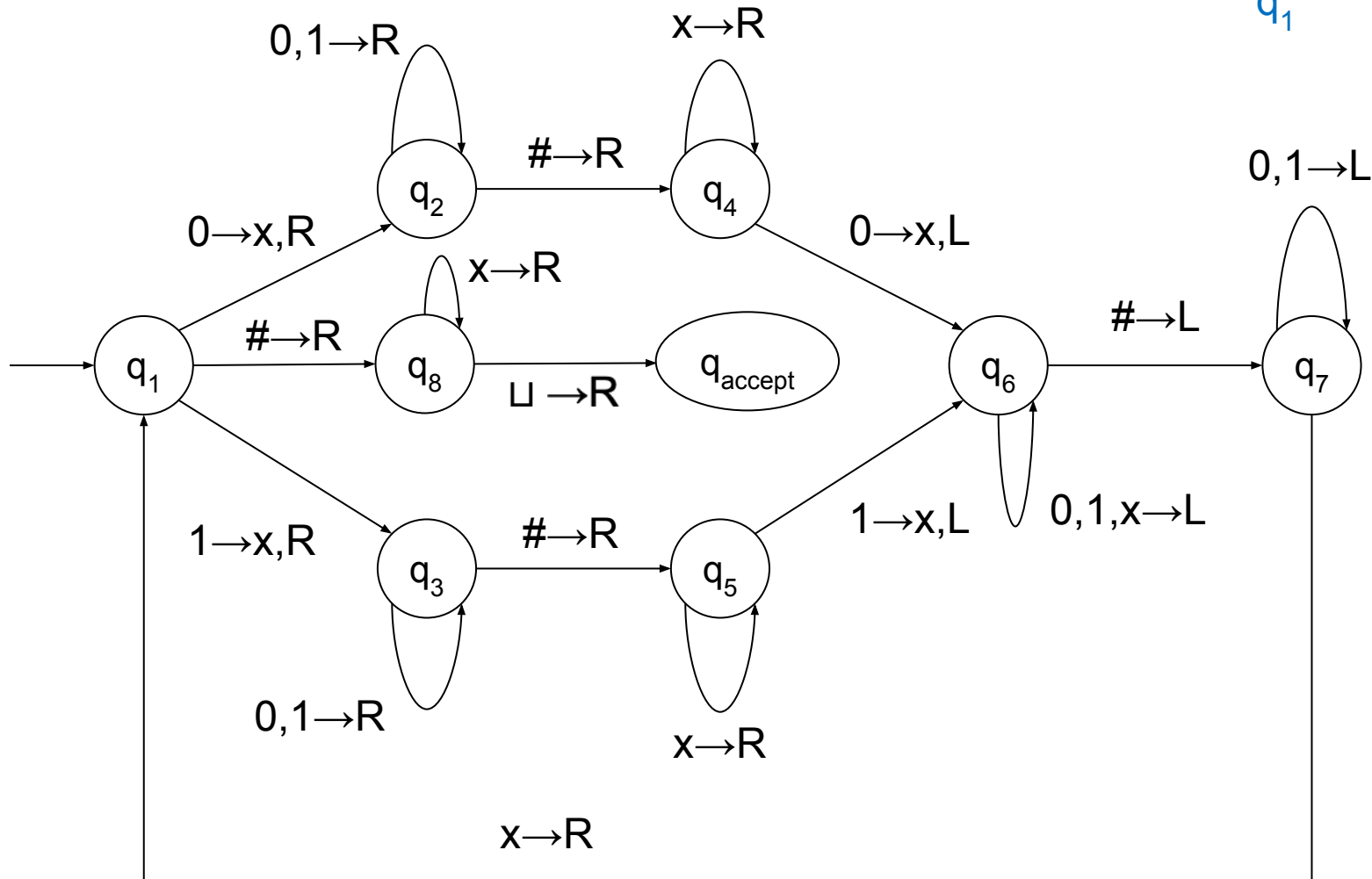
$\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 100, \dots\}$

(0 or 1 repeated 0 or more times)

State diagram



State diagram



Formal definition

$$AA = \{w\#w \mid w \in \{0,1\}^*\}$$

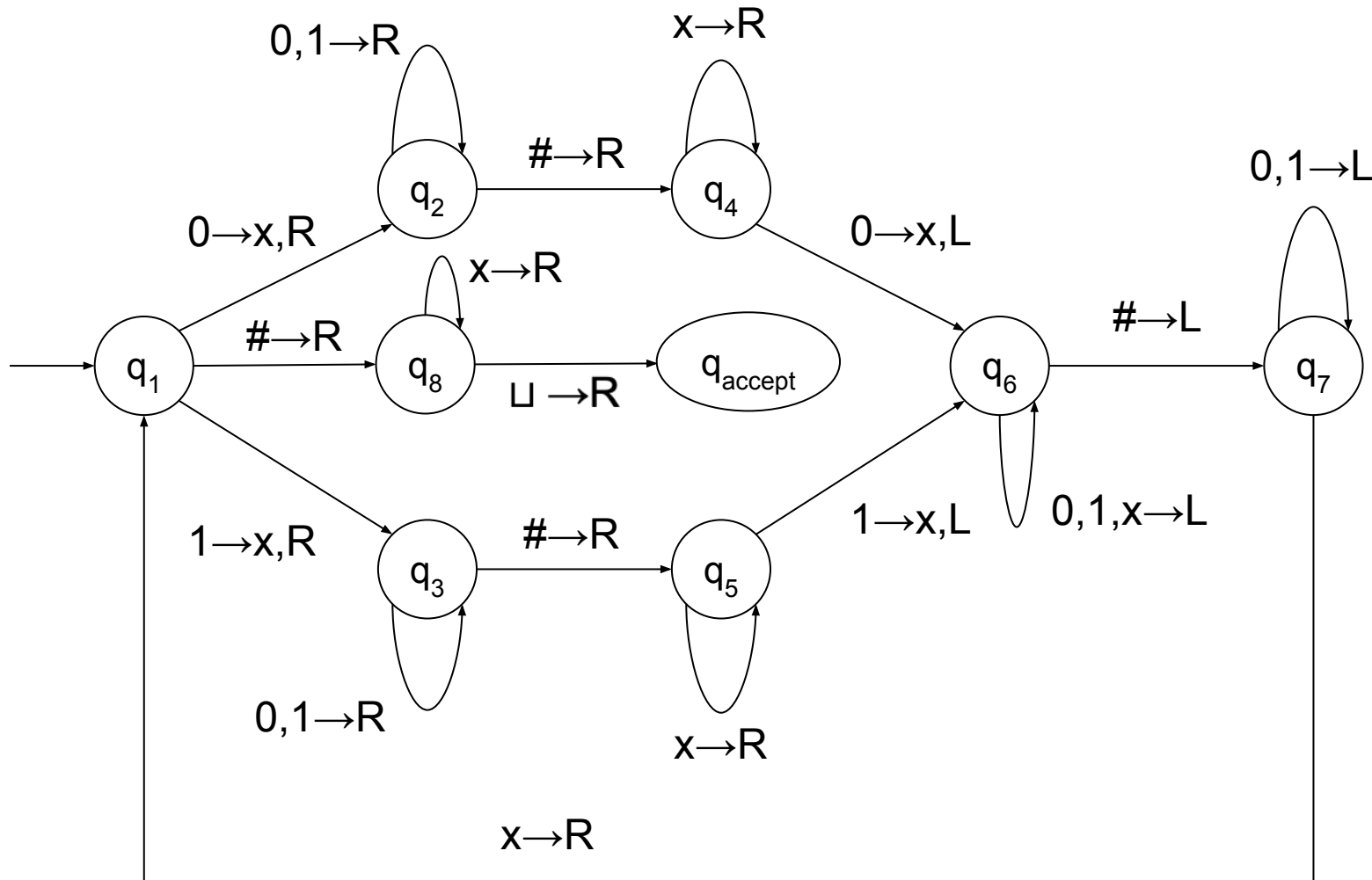
$$Q = \{q_1, q_2, \dots, q_8, q_{\text{accept}}, q_{\text{reject}}\}$$

$$\Sigma = \{0, 1, \#\}$$

$$\Gamma = \{0, 1, \#, x, \sqcup\}$$

δ is a state diagram

q_0 is q_1



Transition table

State \ Symbol	0	1	x	#	␣
q1	(q2,x,R)	(q3,x,R)	-	(q8, ,R)	-
q2	(q2, ,R)	(q2, ,R)	-	(q4, ,R)	-
q3					
...					
q8	-	-	(q8, ,R)	-	(qaccept, ,)

Problem 3

Design and formally describe a Turing machine that decides $B = \{0^{2^n} \mid n \geq 0\}$, the language consisting of all strings of 0s whose length is a power of 2. Include

- a) a high-level description of its algorithm,
- b) a formal description of the Turing machine,
- c) a transition/state diagram of the Turing machine,
- d) a sample run of the machine on the string 0000 noting its configuration at each step.

Example strings:

$$0^{2^0} = 0$$

$$0^{2^1} = 00$$

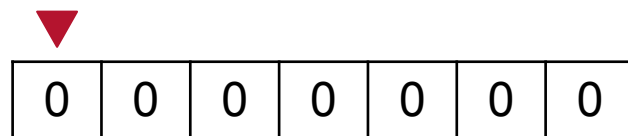
$$0^{2^2} = 0000$$

$$0^{2^3} = 00000000$$

a) a high-level description of its algorithm

On input string w :

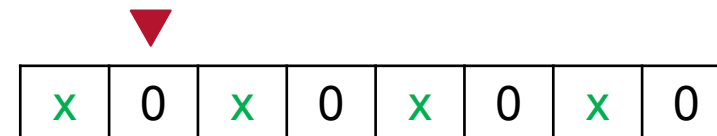
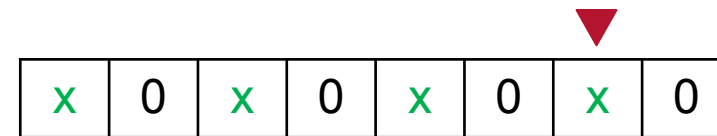
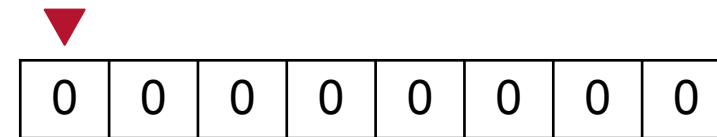
1. Sweep left to right across the tape, crossing off every other 0 (Divide by 2)
2. If the tape contained a single 0 \rightarrow accept
3. Return the head to the left-hand end of the tape, repeat from 2
4. If all inputs are crossed off (all x) or last=0 (only remaining zero is the last) \rightarrow accept
5. If the tape contained more than a single 0 and the number of 0s was odd \rightarrow reject



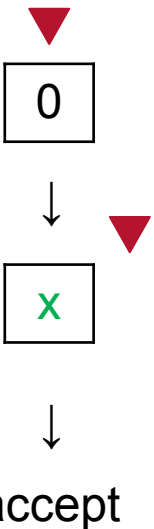
\rightarrow



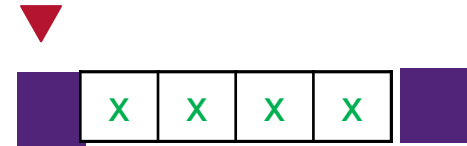
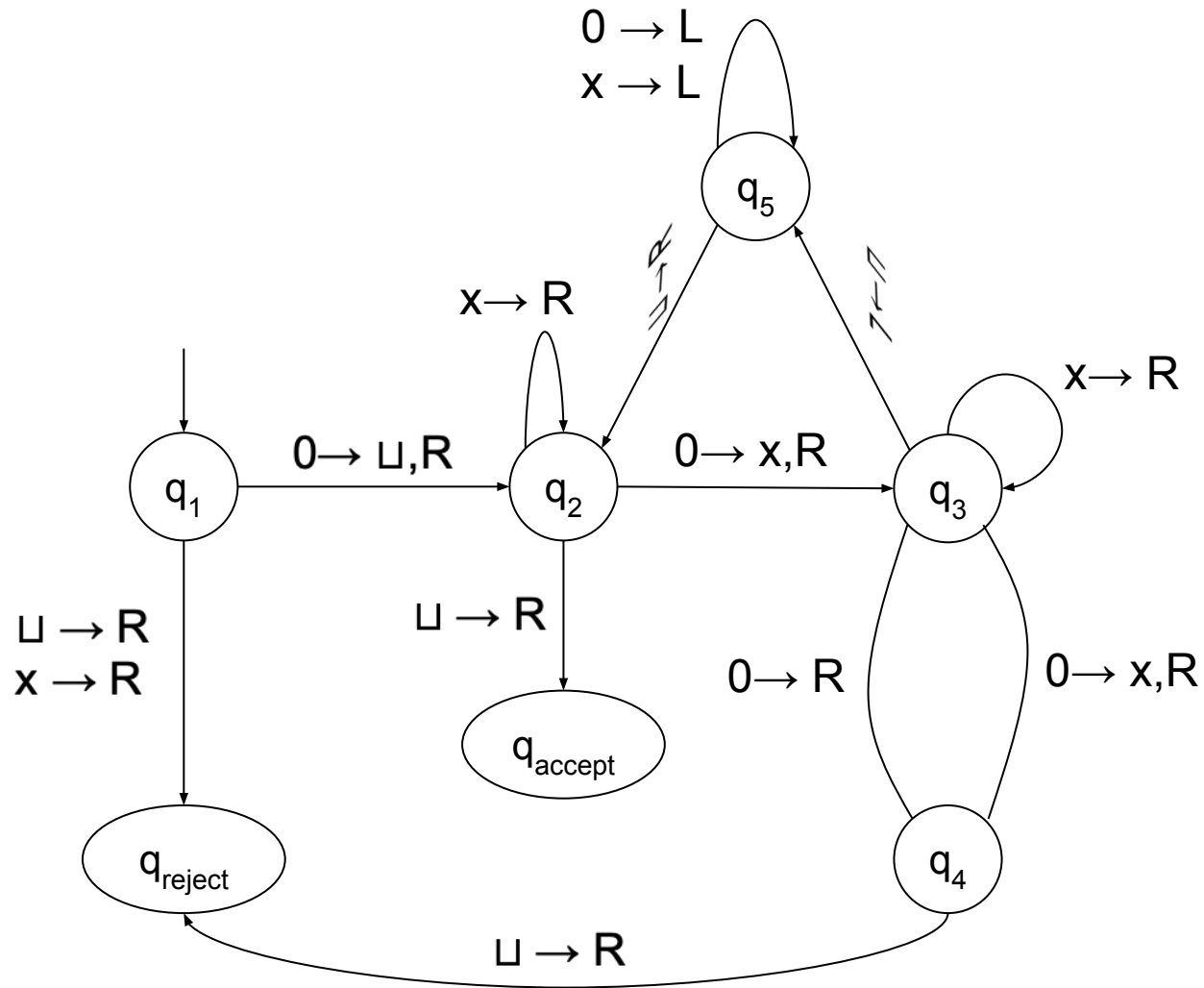
\rightarrow reject



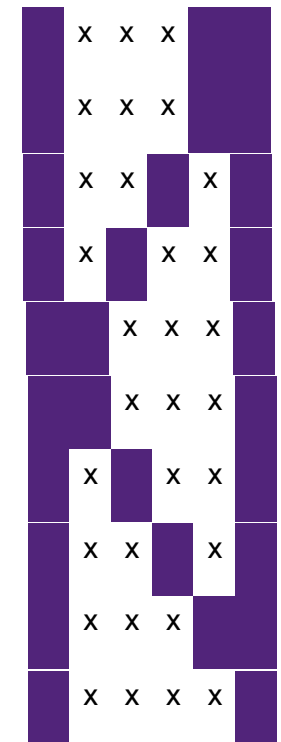
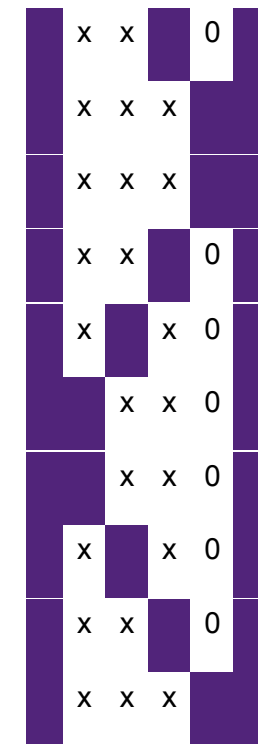
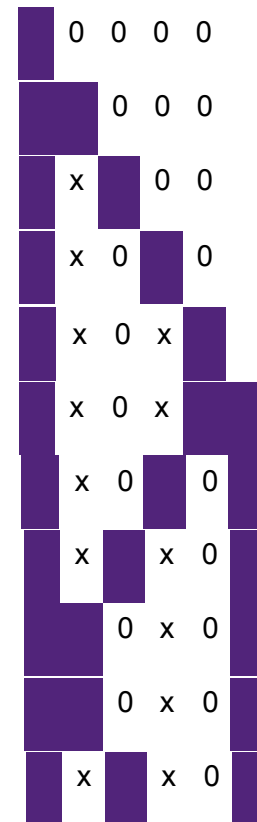
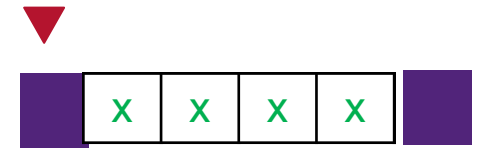
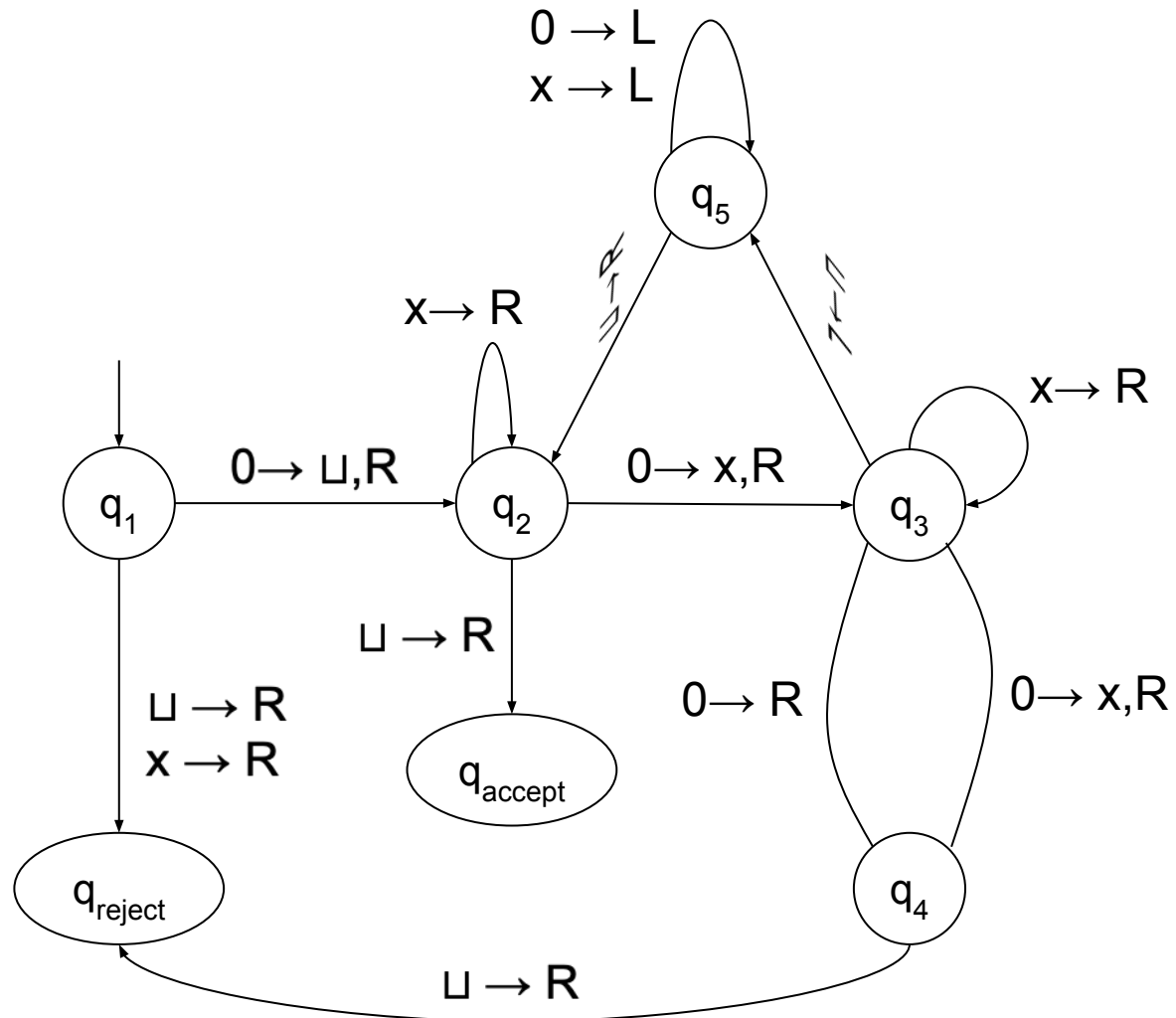
\rightarrow accept



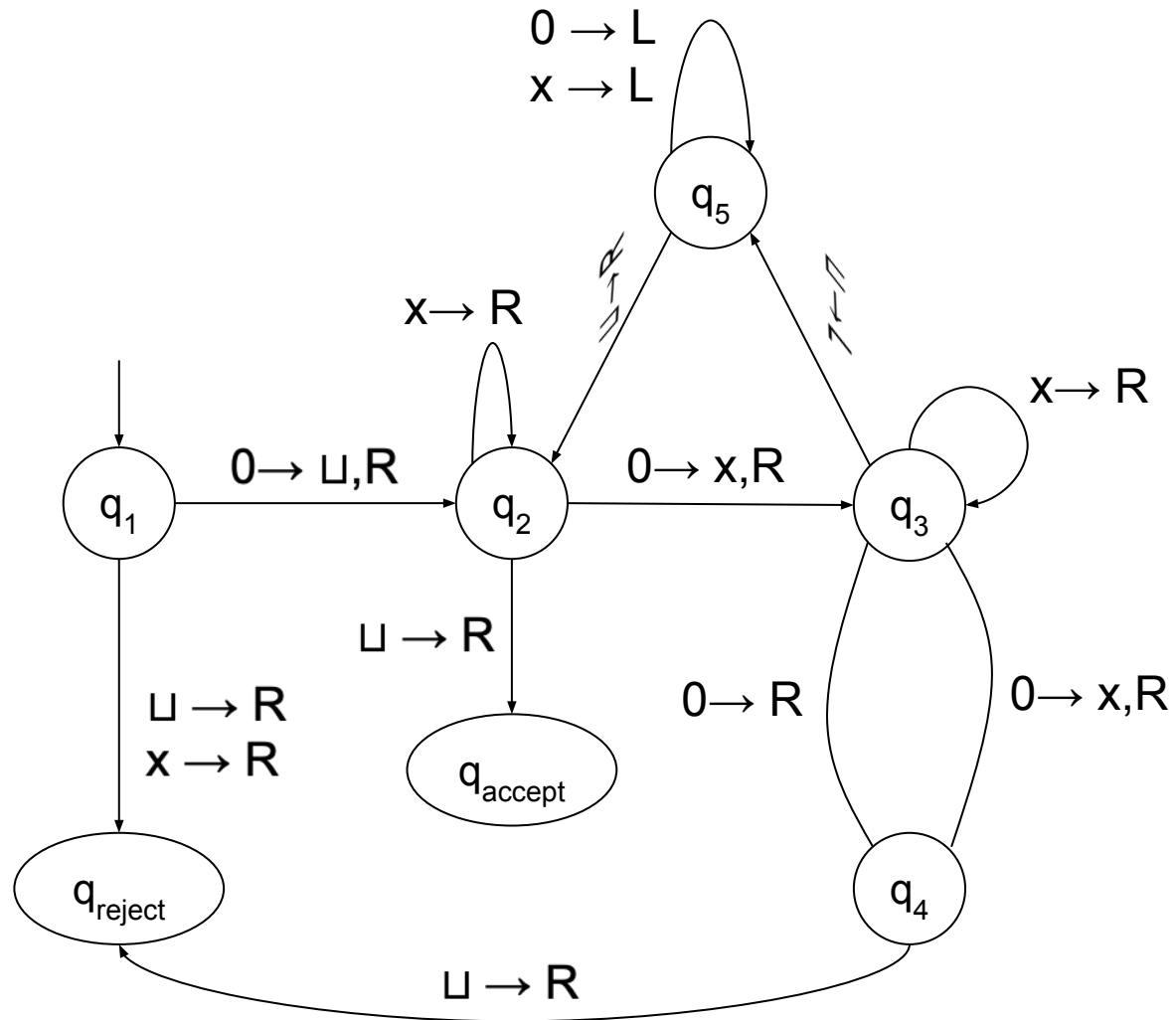
c) a transition/state diagram of the Turing machine



d) a sample run of the machine on the string 0000 noting its configuration at each step



b) a formal description of the Turing machine



$$B = \{0^{2^n} \mid n \geq 0\}$$

$$Q = \{q_1, q_2, q_3, q_4, q_5, q_{\text{accept}}, q_{\text{reject}}\}$$

$$\Sigma = \{0\}$$

$$\Gamma = \{0, x, \square\}$$

δ is a state diagram

q_0 is q_1

Problem 4

Show that the language $C = \{a^i b^j c^k \mid i \times j = k \text{ and } i, j, k \geq 1\}$ cannot be recognisable by a finite state machine by designing a Turing machine that decides it.

eg: For $i = 2, j = 3 \rightarrow k = 6$

Input $\rightarrow aabbcccccc$

- Need to keep track of i, j, k
- Have unbounded inputs \rightarrow FSM has finite number of states
- For some large sufficiently large input \rightarrow not enough states to process

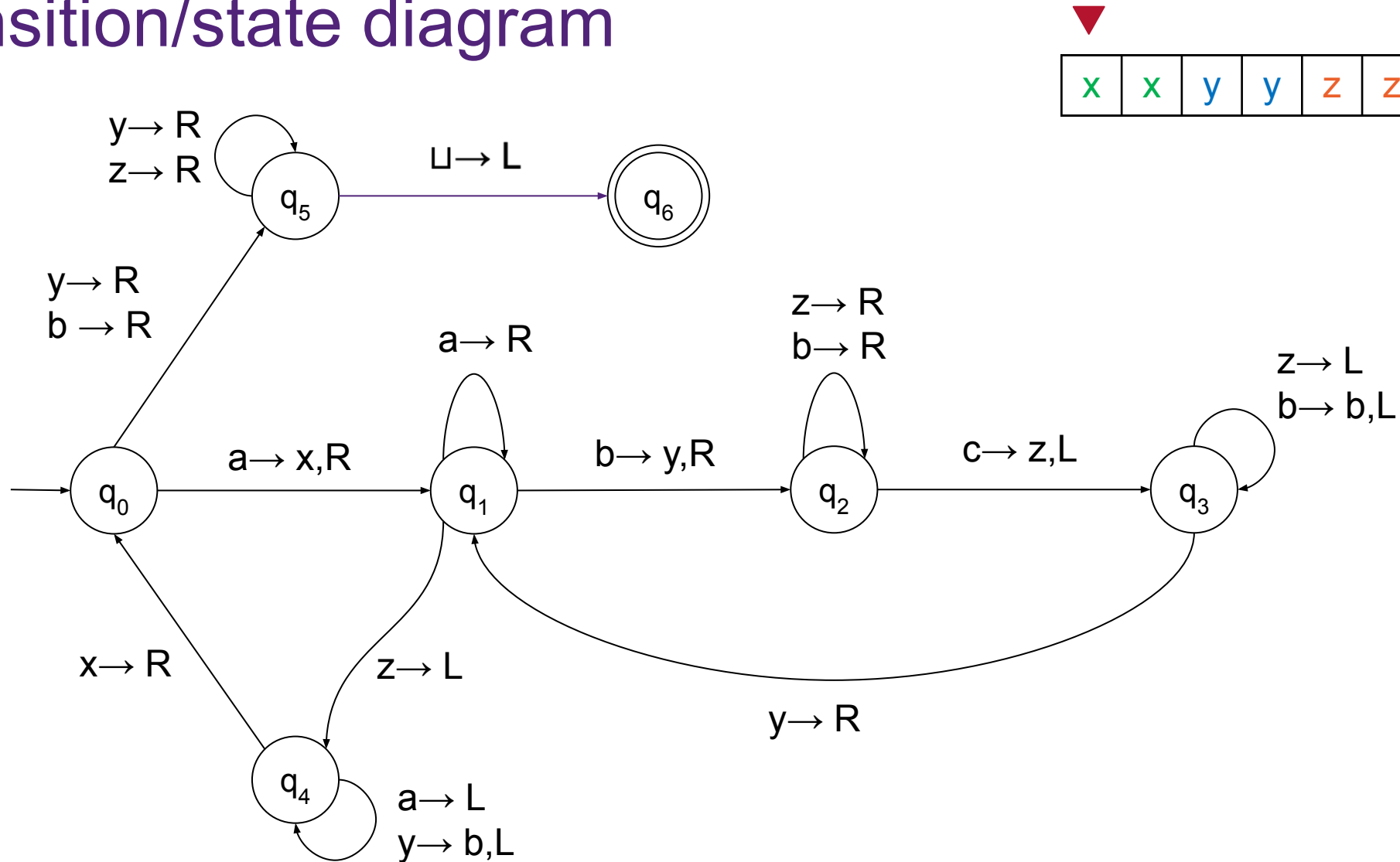
Pseudocode

1. If the input doesn't match $a*b*c^*$ after a full initial sweep \rightarrow reject
2. Move the head back to the leftmost symbol.
3. Cross off an a with an x , scan to the right until b
4. Cross off a b with a y , scan to the right until c
5. Cross off c with a z , move to the left-most uncrossed b
6. Repeat steps 4 & 5, until all b 's are crossed off.
 - If all c 's get crossed off while doing this (uncrossed a 's remaining) \rightarrow reject
7. Move left until left-most uncrossed a , while uncrossing b 's
8. If there's another a left, then repeat from stage 3.
9. If all a 's are crossed out
 - Check if all c 's are crossed off
If yes \rightarrow accept, else \rightarrow reject



Basically, need to multiply a , b and see if it matches c .
(multiplication = repeated addition)

Transition/state diagram



Problem 5

Given the differences between finite state machines and Turing machines, and the definition of a Turing machine. In no more than one page (12pt, normal 2.54 cm margins), argue either for or against why a Turing machine can or cannot obtain artificial intelligence (AI).

Discuss what needs to be added or removed to ensure that AI can result or can't result.

If an AI can result, what restrictions or safeguards do you need in place to protect human-kind from becoming obsolete?

If an AI can't result, how will computers and advancements in computing help human-kind in the future?