BU CS 332 – Theory of Computation

https://forms.gle/BzMCDFv5fFddcu936

Lecture 9:

- Turing Machines

Reading:

Sipser Ch 3.1, 3.3

Mark Bun

February 26, 2024

Turing Machines – Motivation

We've seen finite automata as a restricted model of computation

Finite Automata / Regular Expressions

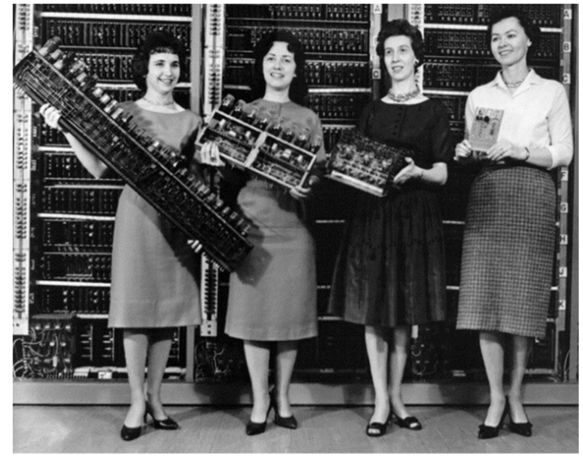
- Can do simple pattern matching (e.g., substrings), check parity, addition
- Can't perform unbounded counting $\{0^n 1^n \mid n \geq 0\}$ is not regular
- Can't recognize palindromes $\{w \mid w = w^R\}$

Somewhat more powerful (not in this course):

Pushdown Automata / Context-Free Grammars

- Can count and compare, parse math expressions Recognizable by PDAs
- Can't recognize $\{a^n b^n c^n \mid n \geq 0\}$

Turing Machines – Motivation



<u>Goal:</u>

Define a model of computation that is

1) General purpose. Captures <u>all</u> algorithms that can be implemented in any programming language.

2) Mathematically simple. We can hope to prove that things are <u>not</u> computable in this model.

A Brief History

1900 – Hilbert's Tenth Problem

Ex. Diophantine Equation

$p(x,y,z) = 3xz - x^2y$

$\exists\ (x,y,z) \in \mathbb{Z}^3$ s.t. $p(x,y,z) = 0$?

$(x,y,z) = (1,3,1)$ satisfies $p(x,y,z) = 0$

Given a Diophantine equation with any number of unknown quantities and with rational integral numerical coefficients: To devise a process according to which it can be determined in a finite number of operations whether the equation is solvable in rational integers.

→ Algorithm

David Hilbert 1862-1943

1928 – The *Entscheidungsproblem*

Wilhelm Ackermann 1896-1962

The "Decision Problem"

Is there an algorithm which takes as input a formula (in first-order logic) and decides whether it's logically valid?

Given: A mathematical statement

Output: Is that statement true or false?

David Hilbert 1862-1943

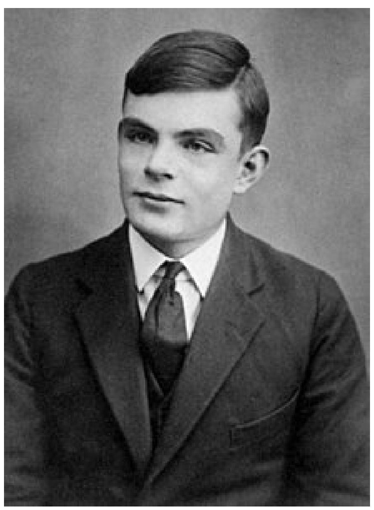
1936 – Solution to the *Entscheidungsproblem*

Alonzo Church 1903-1995

"An unsolvable problem of elementary number theory“

Model of computation: λ-calculus   (CS 320)

≈ regular expressions



Alan Turing 1912-1954

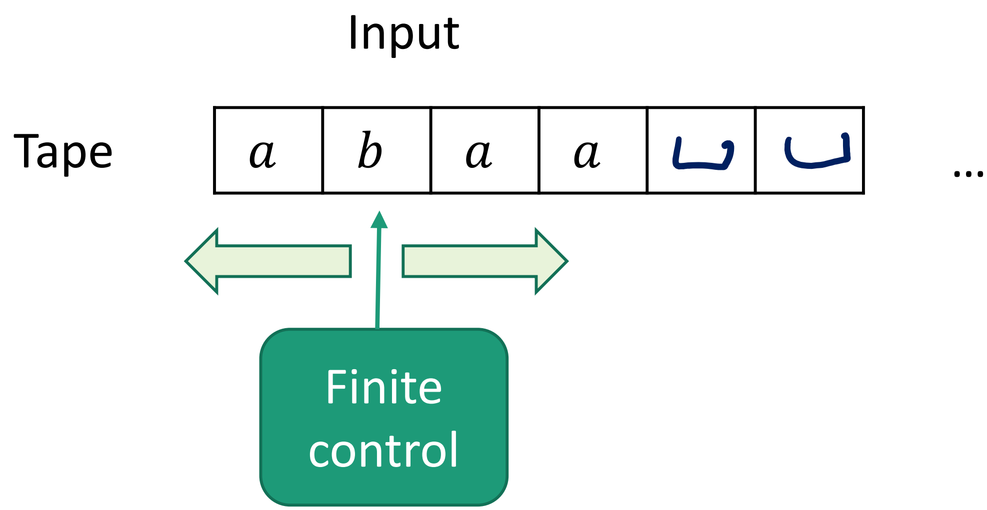
“On computable numbers, with an application to the *Entscheidungsproblem*”

Model of computation: Turing Machine

≈ Finite automata

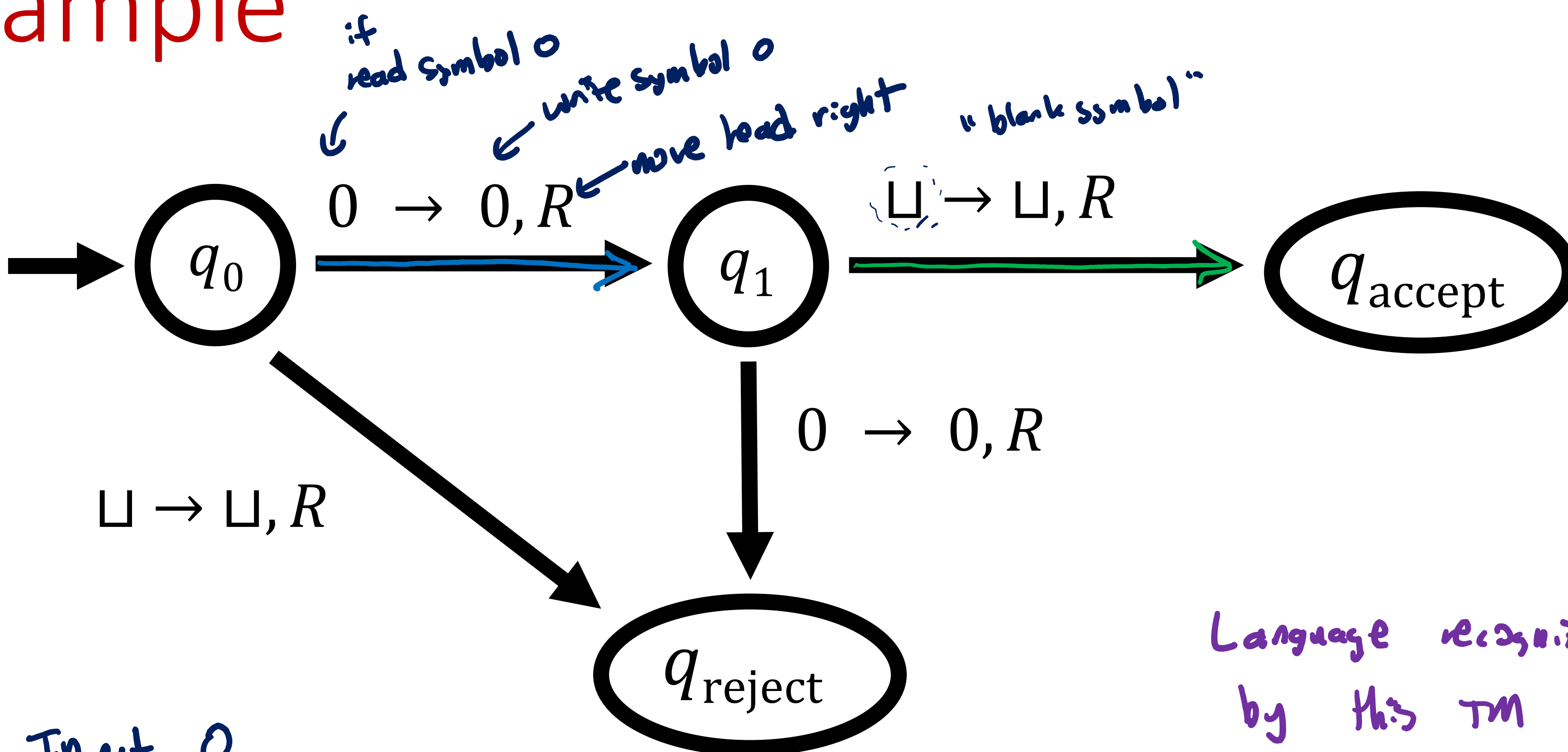
Turing Machines

The Basic Turing Machine (TM)



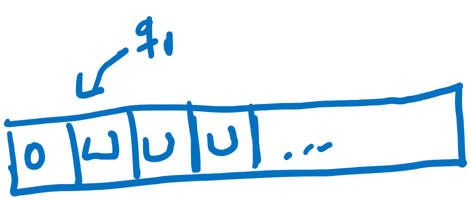
- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts as soon as control reaches "accept" or "reject" state

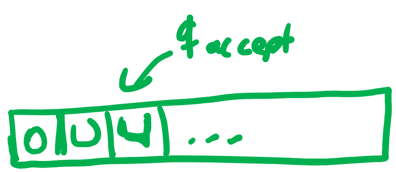
Example



Language recognized by this TM is $\{0\}$

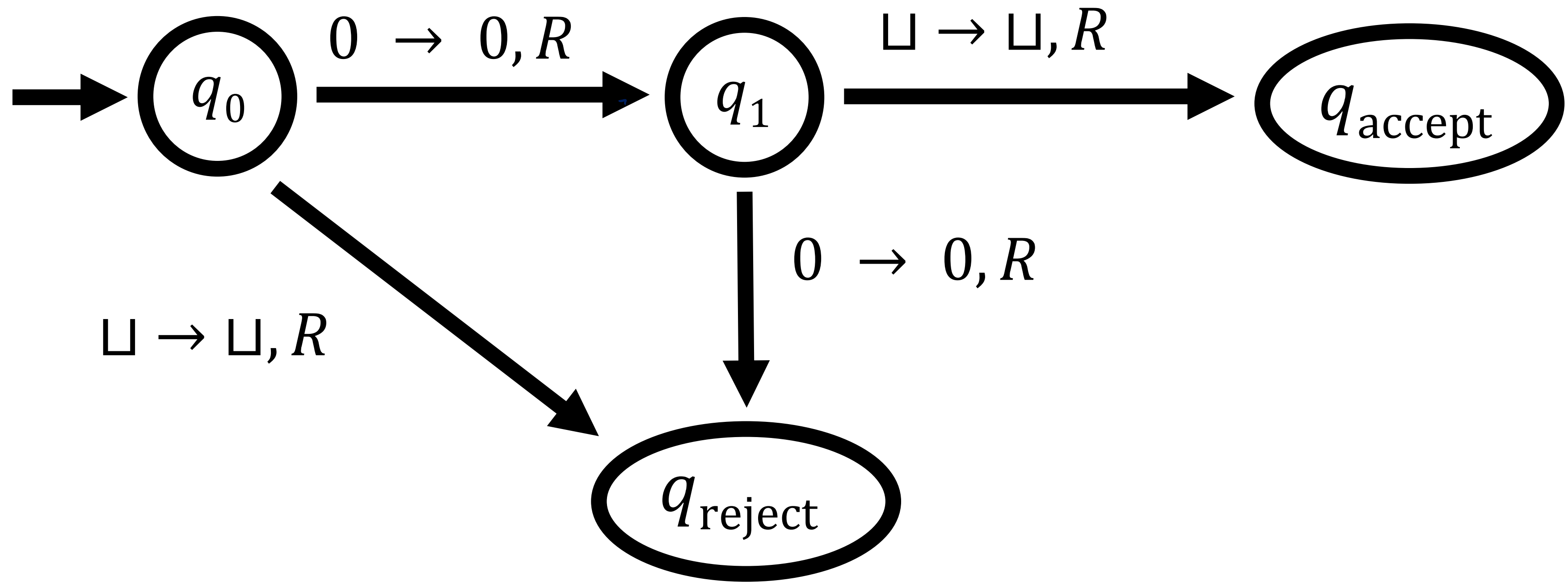
<u>Ex:</u> Input 0





Accepts input 0.

Example



Input 00:

① q_0

0	0	⊔	⊔	...

② q_1

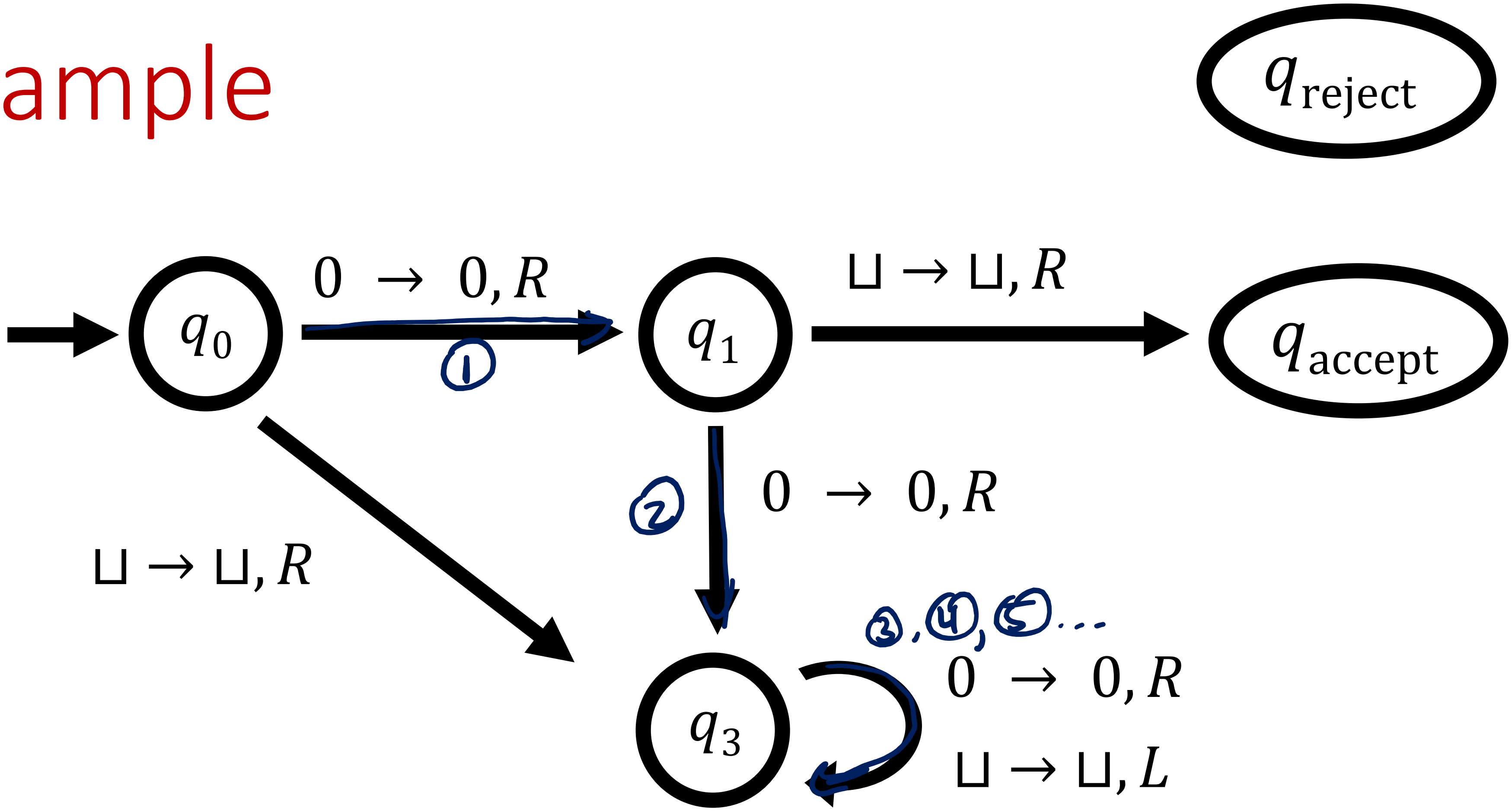
0	0	⊔	⊔	...

③ q_{reject}

0	0	⊔	⊔	...

Rejects input 00

Example



What does this TM do on input 000?

a) Halt and accept
b) Halt and reject
c) Halt in state q_3
d) Loop forever without halting

Three Levels of Abstraction

Programming Analogy

High-Level Description

An algorithm (like CS 330)

Python, Java

Implementation-Level Description

Describe (in English) the instructions for a TM

C, assembly

- How to move the head
- What to write on the tape

Low-Level Description

State diagram or formal specification

Machine code, byte code

Example

Determine if a string $w \in \{0\}^*$ is in the language

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

Given a string of all 0's, is its length a power of 2?

1
2
3
4 → accept

1
2 reject

High-Level Description

Repeat the following forever:

- If there is exactly one 0 in w, accept
- If there is an odd (> 1) number of 0s in w, reject
- Delete half of the 0s in w

Example

Determine if a string $w \in \{0\}^*$ is in the language

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

$0\ 0000 \sqcup \sqcup \sqcup \sqcup \ldots$

Implementation-Level Description

Where is the head?

what is written to tape

1. While moving the tape head left-to-right:
   a) Cross off every other 0 {i.e. replace every other 0 w/ X]
   b) If there is exactly one 0 when we reach the right end of the tape, accept
      = first blank symbol
   c) If there is an odd (> 1) number of 0s when we reach the right end of the tape, reject
2. Return the head to the left end of the tape
3. Go back to step 1

Example

Determine if a string $w \in A = \{0^{2^n} \mid n \geq 0\}$

Low-Level Description

Input $w = 0000$

$q_0 0000 \sqcup \ldots$ ①

$\sqcup q_1 000 \sqcup \ldots$

$\sqcup x q_3 00 \sqcup \ldots$ ②

$\sqcup x 0 q_4 0 \sqcup \ldots$

$\sqcup x 0 x q_3 \sqcup \ldots$

$\sqcup x 0 q_2 x \sqcup \ldots$

$\sqcup x q_2 0 x \sqcup \ldots$

$\sqcup q_2 x 0 x \sqcup \ldots$

$q_2 \sqcup x 0 x \sqcup \ldots$

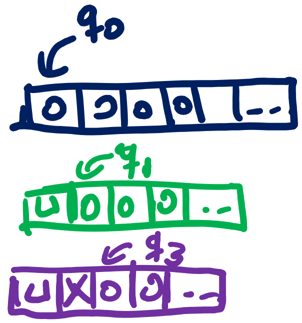
$\sqcup q_1 x 0 x \sqcup \ldots$

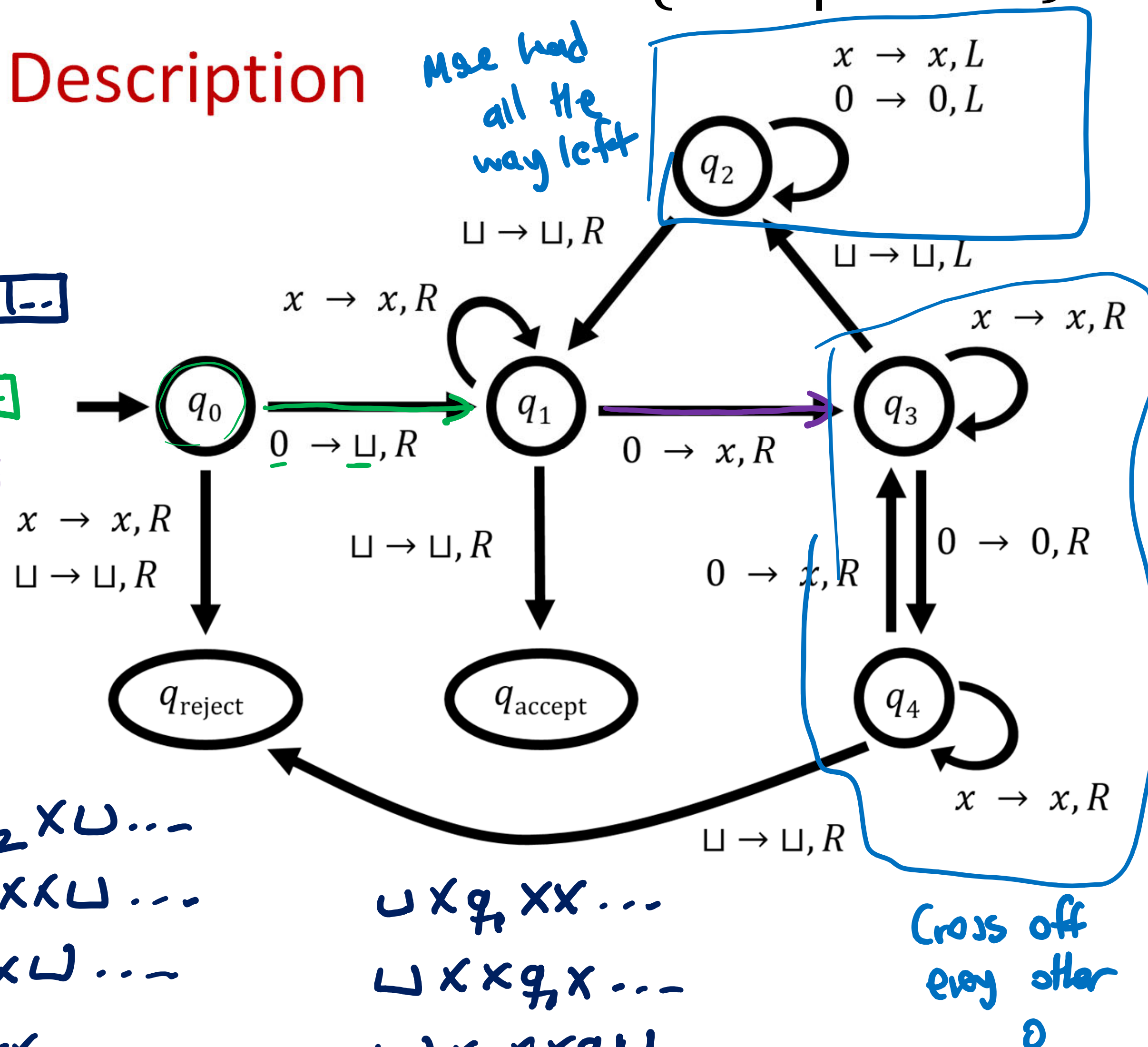
$\sqcup x q_1 0 x \sqcup \ldots$

$\sqcup x x q_3 x \sqcup \ldots$

$\sqcup x x x q_3 \sqcup \ldots$

means





$\sqcup x x q_2 x \sqcup \ldots$

$\sqcup x q_2 x x \sqcup \ldots$

$\sqcup q_2 x x x \sqcup \ldots$

$q_2 \sqcup x x x \ldots$

$\sqcup q_1 x x x \ldots$

$\sqcup x q_1 x x \ldots$

$\sqcup x x q_1 x \ldots$

$\sqcup x x x q_1 \sqcup \ldots$

$\sqcup x x x \sqcup q_{accept} \sqcup \ldots$

Accept!

Differences between TMs and Finite Automata

TMs can write

TMs can move head both left and right

TMs are deterministic (no ε-transitions, branching like NFAs)

TM halts computation when reaching accept or reject state
(vs. FAs that have to read whole input)

TMs can enter infinite loops

TMs alphabet can have more symbols than just input alphabet

Formal Definition of a TM

A TM is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$

- Q is a finite set of states
- Σ is the input alphabet (does **not** include ⊔)
- Γ is the tape alphabet (contains ⊔ and Σ)

$\Sigma \subseteq \Gamma$

$\sqcup \in \Gamma$

- δ is the transition function

 ...more on this later

- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state ($q_{\text{reject}} \neq q_{\text{accept}}$)

TM Transition Function

next state — symbol to write under head — movement instructions

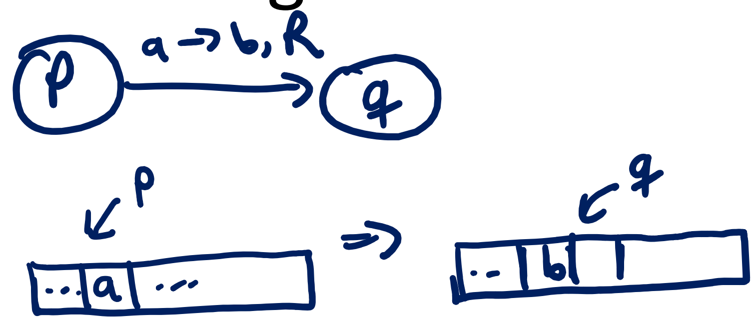
$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

current state — symbol currently under head

L means "move left" and R means "move right"

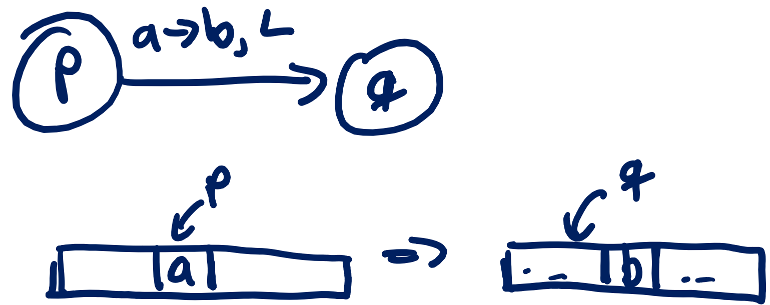
$\delta(p, a) = (q, b, R)$ means:

- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right



$\delta(p, a) = (q, b, L)$ means:

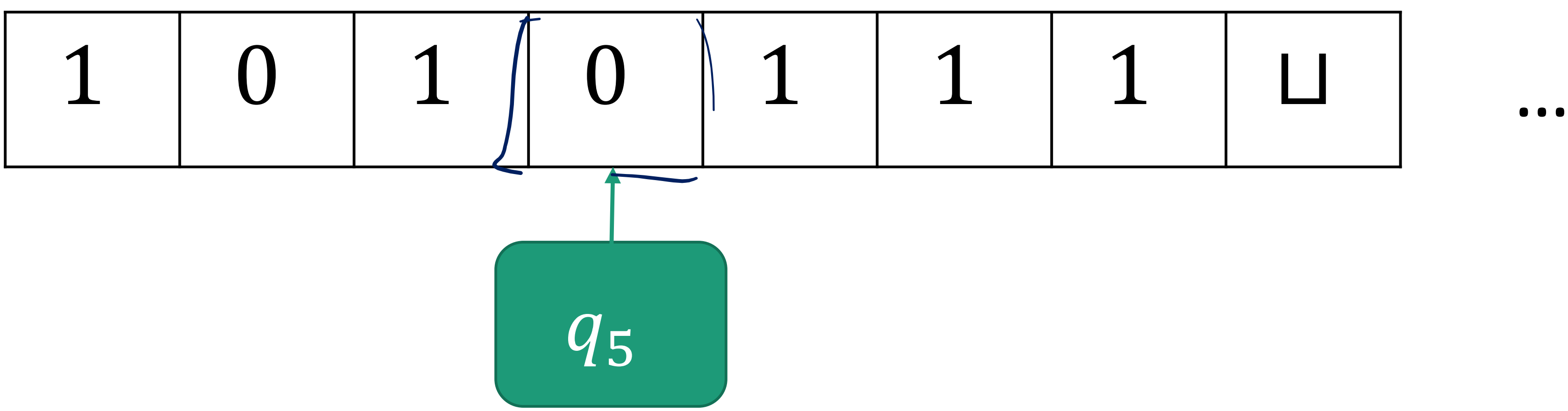
- Replace a with b in current cell
- Transition from state p to state q
- Move tape head left UNLESS we are at left end of tape, in which case don't move



Configuration of a TM

A string that captures the **state** of a TM together with the **contents of the tape**

$1\,0\,1\,q_5\,0\,1\,1\,1$

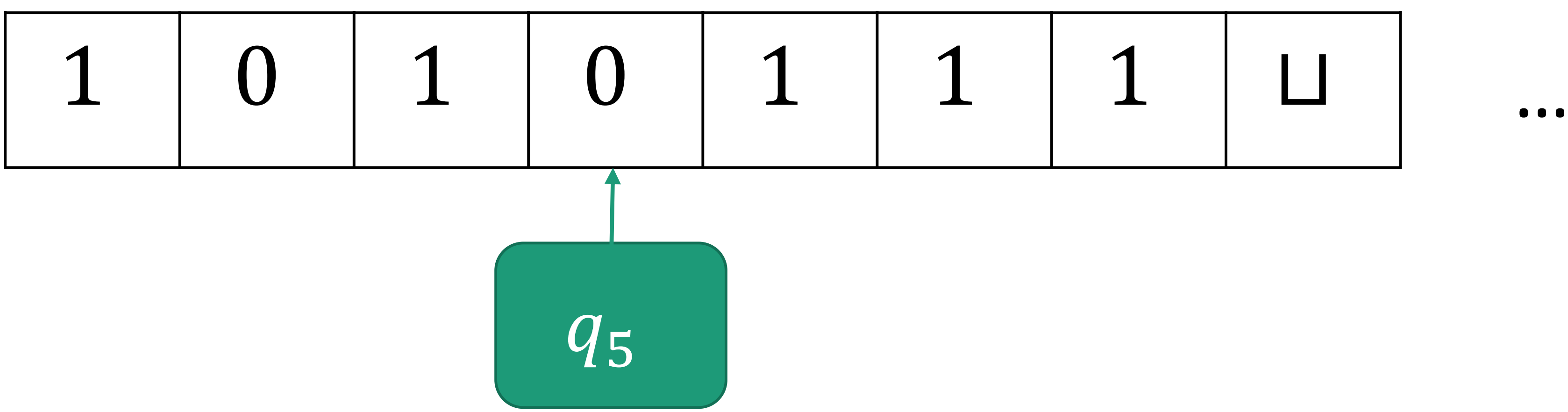


Configuration of a TM: Formally

A configuration is a string uqv where $q \in Q$ and $u, v \in \Gamma^*$

- Tape contents = uv (followed by infinitely many blanks ⊔)
- Current state = q
- Tape head on first symbol of v

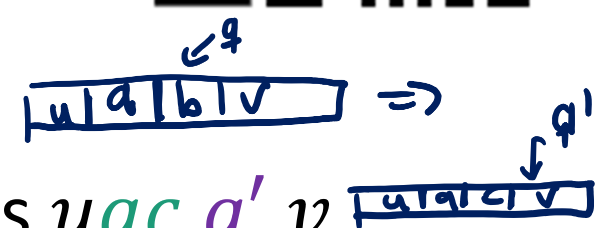
Example: $\overbrace{101}^{u}\overbrace{q_5}^{q}\overbrace{0111}^{v}$



How a TM Computes

Start configuration: $q_0 w$

In one step of computation:



- If $\delta(q, b) = (q', c, R)$, then $ua\ q\ bv$ yields $uac\ q'\ v$
- If $\delta(q, b) = (q', c, L)$, then $ua\ q\ bv$ yields $u\ q'\ acv$
- If we are at the left end of the tape in configuration $q\ bv$, what configuration do we reach if $\delta(q, b) = (q', c, L)$?

a) $cq'v$

b) $q'cv$

c) $q' \sqcup cv$

d) $q'cbv$