BU CS 332 – Theory of Computation

Lecture 9:

- Turing Machines

Reading:

Sipser Ch 3.1, 3.3

Mark Bun

February 22, 2021

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
- Can't recognize palindromes

$\{a^n b^n \mid n \geq 0\}$

$\{w \mid w = w^R\}$

via distinguishing sets

Somewhat more powerful (not in this course):

Pushdown Automata / Context-Free Grammars

Ch 2 of Sipser

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

Really easy to write a program that solves this problem

Turing Machines – Motivation

Goal:

Define a model of computation that is

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

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

A Brief History

1900 – Hilbert's Tenth Problem

$p(x,y,z) = x^2 - y^2 z + z^3 + 1$

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

$(x,y,z) = (0,0,0)$ is a solution

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.

"an 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 math statement:
Is the 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 expression

Alan Turing 1912-1954

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

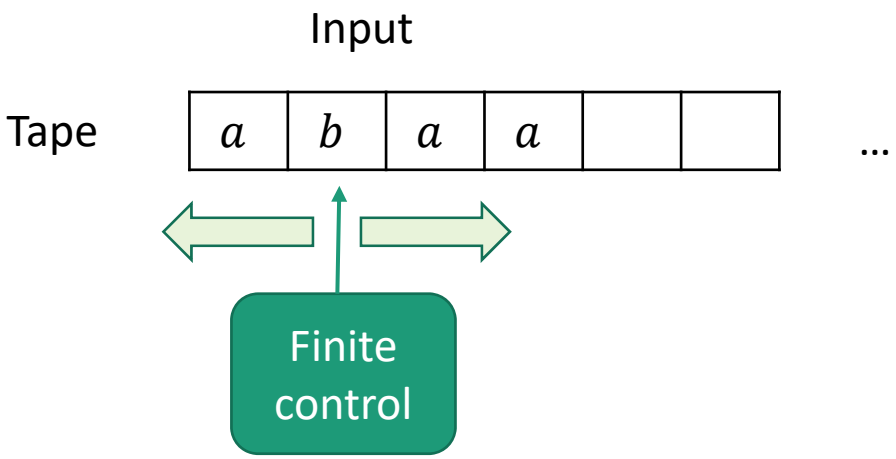
Model of computation: Turing Machine

≈ finite automata

Turing Machines

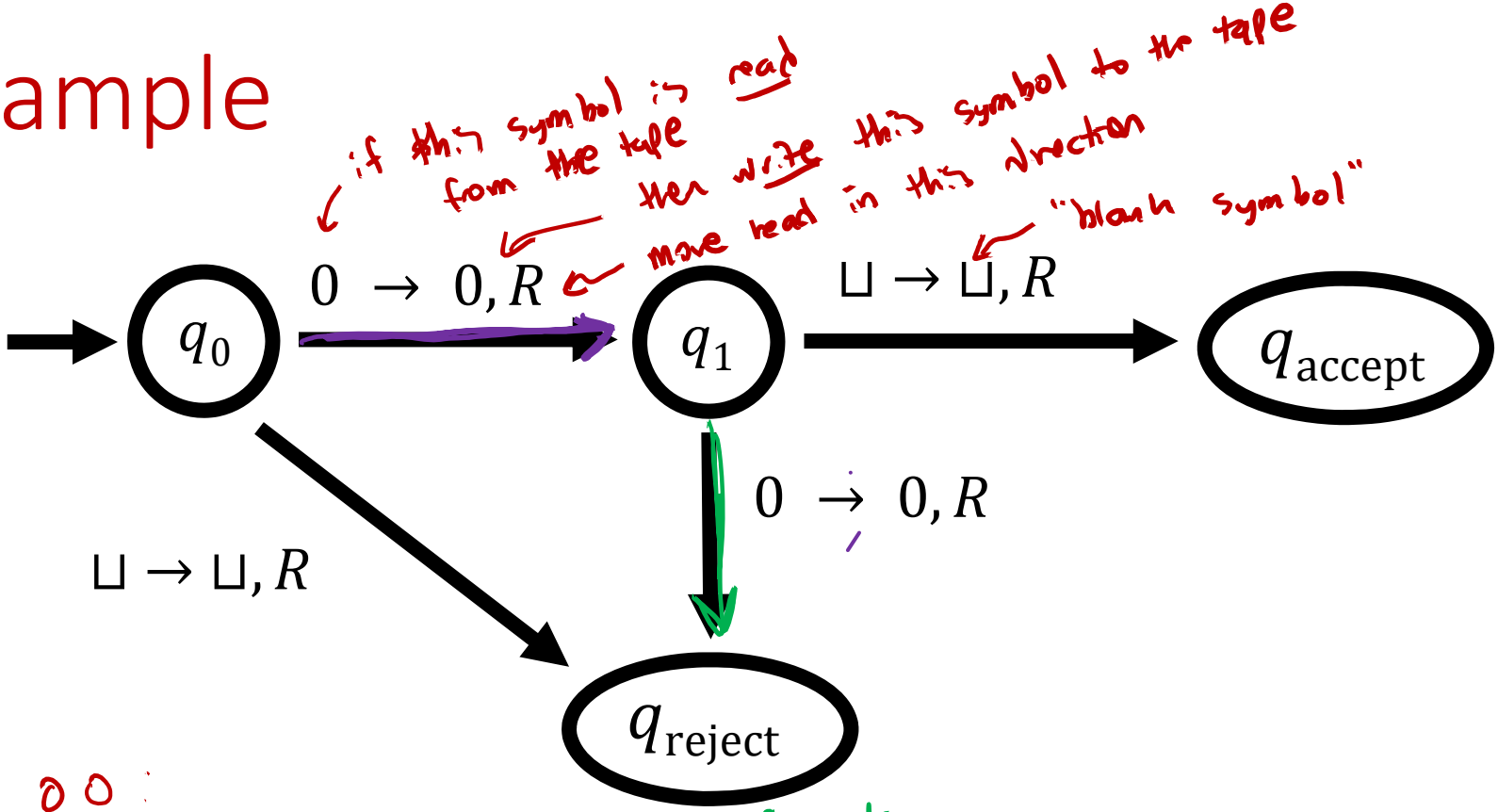
The Basic Turing Machine (TM)

("single tape TM")



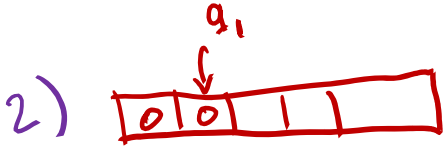
- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts as soon as control reaches "accept" or "reject" state

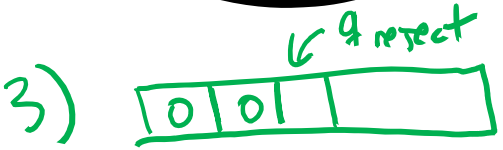
Example



Input 0 0 :

1) 

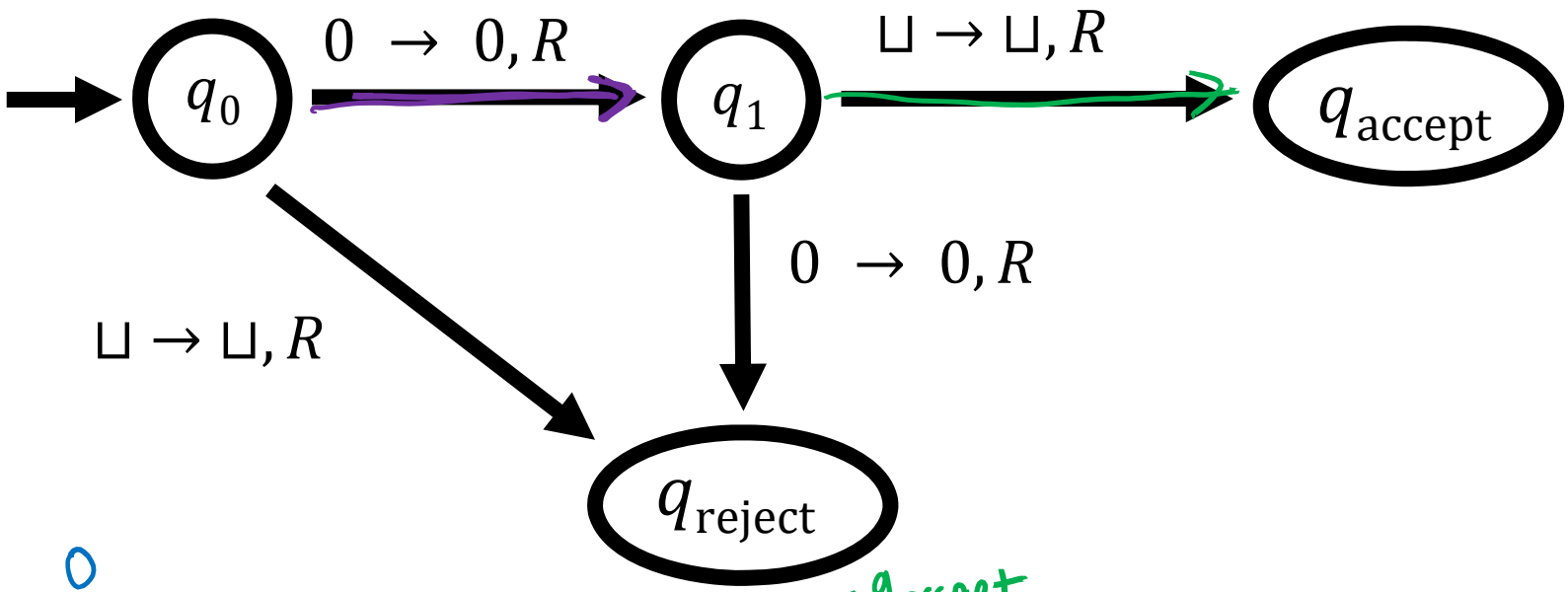
2) 

3) 

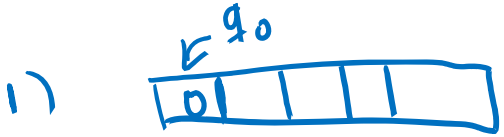
TM rejects this input

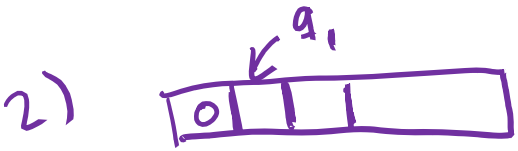
Example

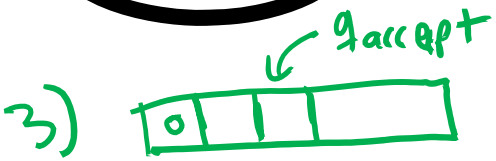
$\Sigma = \{0\}$



Input 0

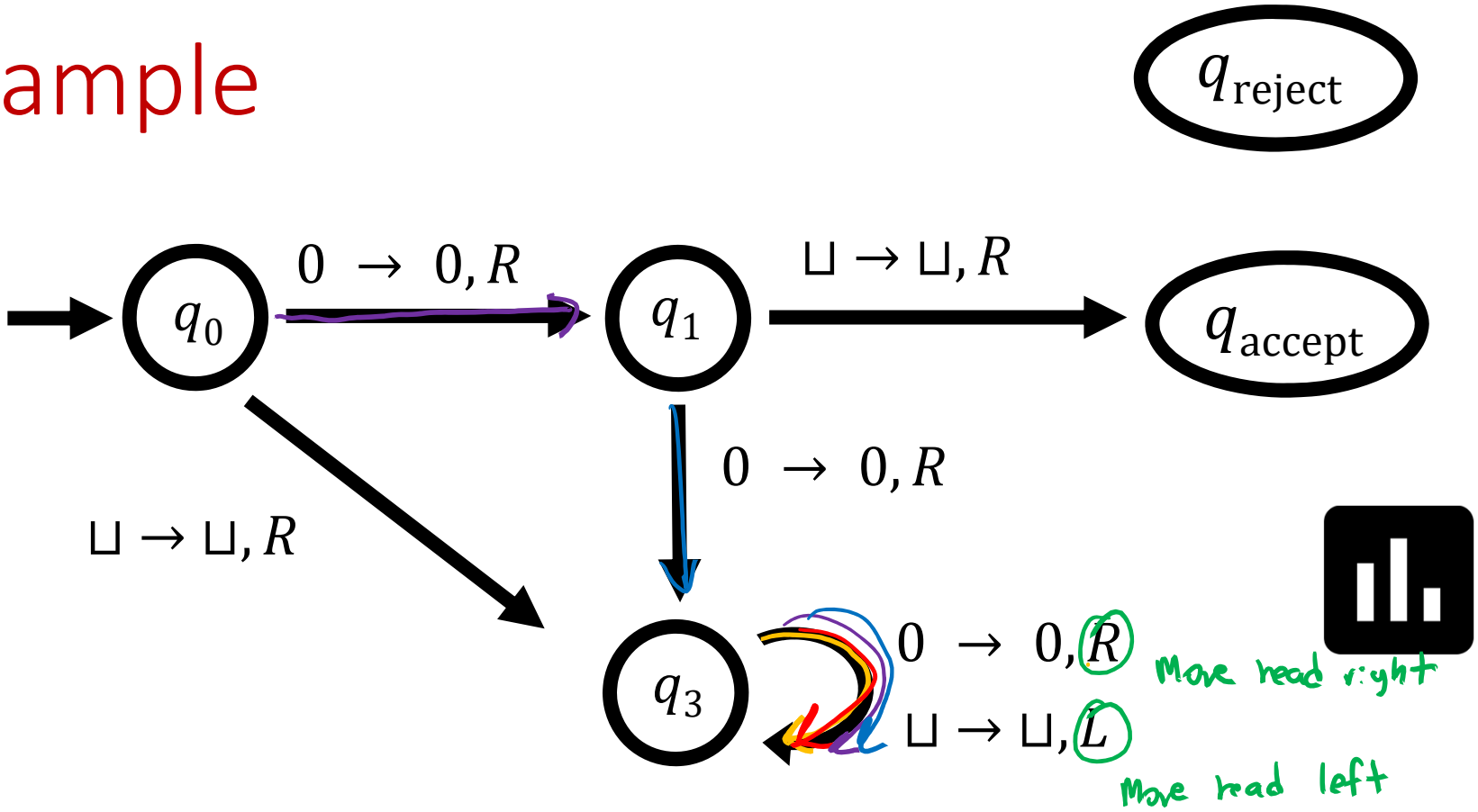
1) 

2) 

3) 

TM accept 0

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

1) $q_0 000$
2) $0 q_1 00$
3) $00 q_2 0$
4) $000 q_3$
5) $00 q_3 0$
6) $00 0 q_3$
7) $00 q_3 0$

Three Levels of Abstraction

High-Level Description

Python, Java code

An algorithm (like CS 330)

Implementation-Level Description

Describe (in English) the instructions for a TM

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

C, assembly lang

Low-Level Description

State diagram or formal specification

Machine code

Example

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

High-Level Description

Repeat the following forever:

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

Example

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

Implementation-Level Description

1. While moving the tape head left-to-right:
   a) Cross off every other 0
   b) If there is exactly one 0 when we reach the right end of the tape, accept
   c) If there is an odd 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: 0000

q_0 0 0 0 0

␣ q_1 0 0 0

␣ x q_3 0 0

␣ x 0 q_4 0

␣ x 0 x q_3

␣ x 0 q_2 x

␣ x q_2 0 x

␣ q_2 x 0 x

q_2 ␣ x 0 x

␣ q_1 x 0 x

␣ x q_1 0 x

␣ x 0 q_1 x

␣ x q_2 x x

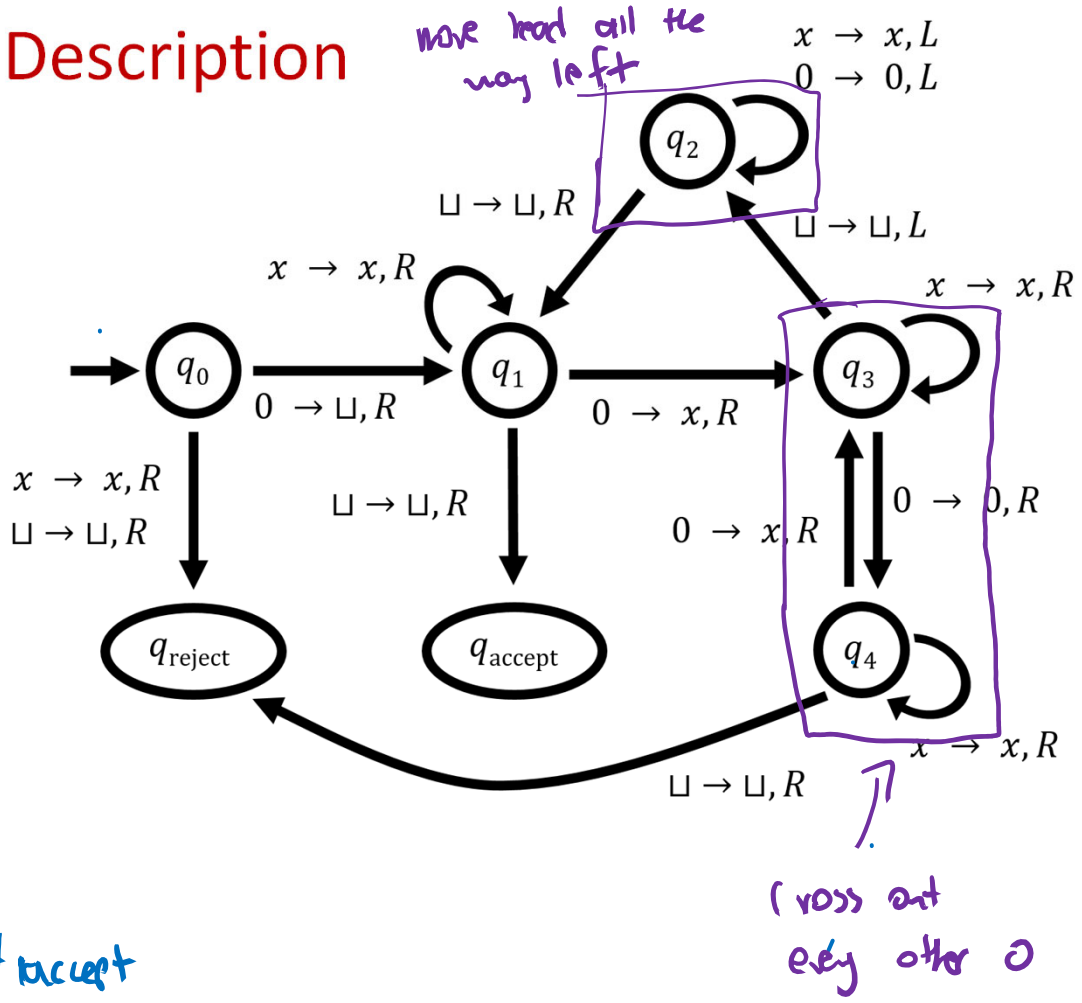
q_2 ␣ x x x

␣ q_1 x x x

␣ x q_1 x x

␣ x x q_1 x

␣ x x x q_{accept}



TMs vs. Finite Automata

Teacher: "zoom breakout rooms are critically important for online learning"

Zoom breakout rooms:

TM has to explicitly manage tape via left/right movements

TMs can write

TMs are actually more powerful (can recog. nonreg. langs)

TMs have blank cells, "tape alphabet" can be bigger than "input alphabet"

TMs can get stuck in loops (Because TM has to explicitly accept or reject to halt)

TMs have infinite memory (infinite tape)

Both TMs and FAs have finite control

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)
- δ is the transition function

e.g. $\Sigma = \{0, 1\}$

e.g. $\Gamma = \{0, x, ⊔\}$

…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

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

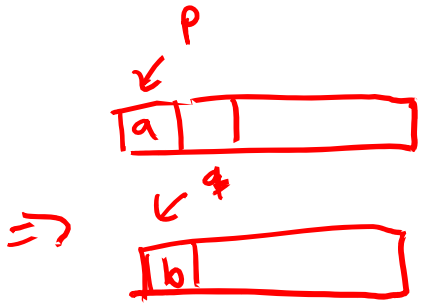
symbol to write; current state; current symbol read; next state; move head left or right

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:

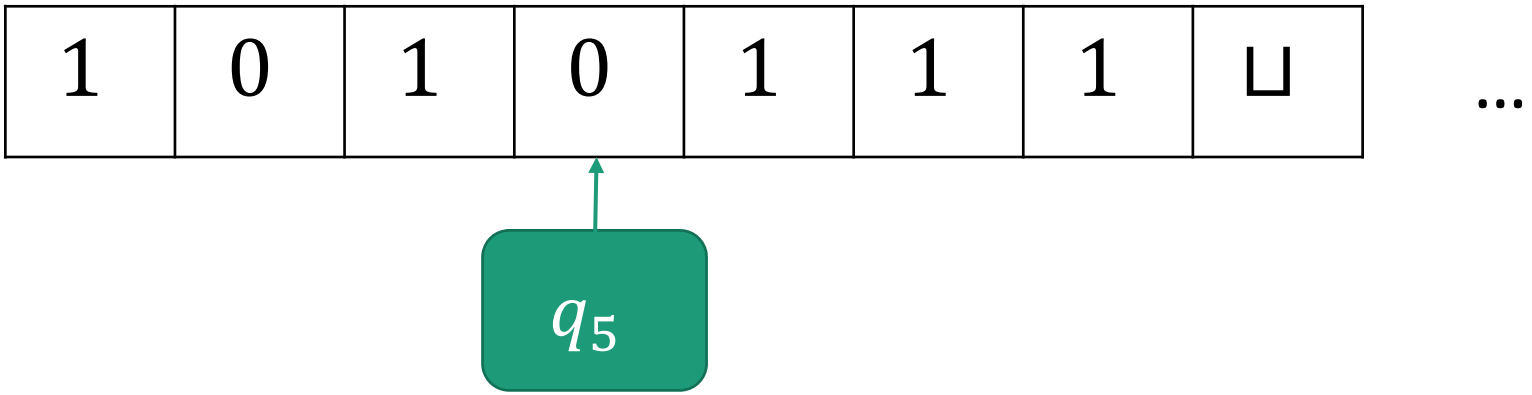


- Replace a with b in current cell
- Transition from state p to state q
- Move tape head left UNLESS we are at left end of tape, in which case don't move

Configuration of a TM

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

1 0 1 q_5 0 1 1 1

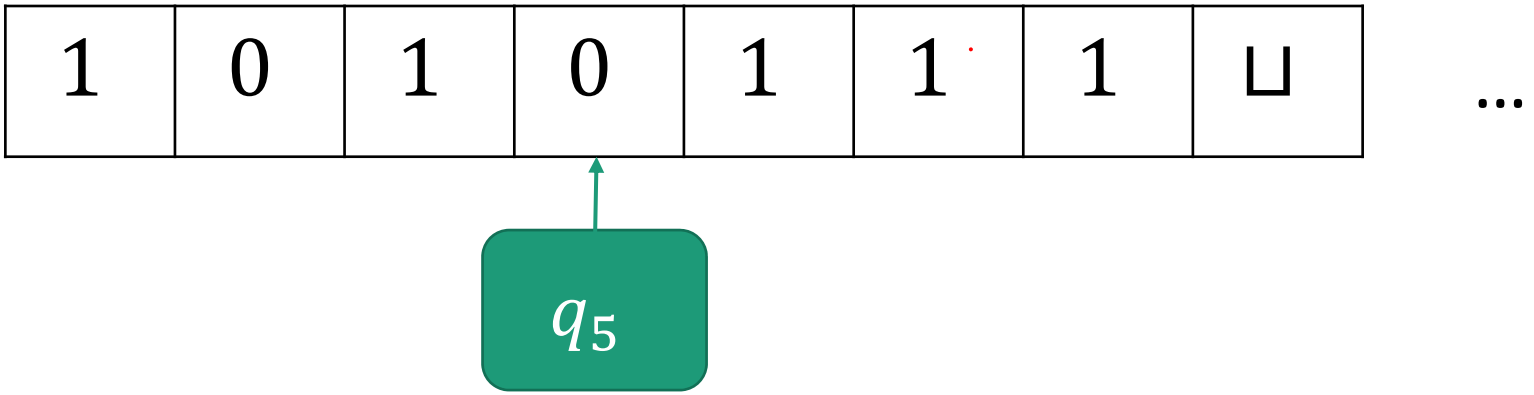


Configuration of a TM: Formally

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

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

Example: $101q_50111$



How a TM Computes

Start configuration: $q_0 w$

One step of computation:

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