BU CS 332 – Theory of Computation

https://forms.gle/XDLCbA5mrSRjNvZX6

Lecture 8:

- More on non-regularity
- Turing Machines

Reading:

"Myhill-Nerode" note

Sipser Ch 3.1, 3.3

Mark Bun

February 18, 2025

Midterm Monday 2/24

Permanent office Hour rooms

HW4 due tonight

Last Time: Distinguishing Set Method

Definition: Strings x and y are **distinguishable** by L if there exists a "distinguishing extension" $z \in \Sigma^*$ such that exactly one of xz or yz is in L.

Definition: A set of strings S is **pairwise distinguishable** by L if every pair of distinct strings $x, y \in S$ is distinguishable by L.

Theorem: If S is pairwise distinguishable by L, then every DFA recognizing L needs at least $|S|$ states.

Corollary: If language L has an infinite pairwise distinguishable set, then L is not regular.

Reusing a Proof

Finding a distinguishing set can take some work…

Let's try to reuse that work!

How might we show that

$$BALANCED = \{w \mid \underline{w \text{ has an equal \# of 0s and 1s}}\}$$

is not regular?

$$\overbrace{\{0^n1^n \mid n \geq 0\}}^{\text{not regular}} = BALANCED \cap \overbrace{\{w \mid \text{all 0s in } w \text{ appear before all 1s}\}}^{\text{regular: } = L(0^*1^*)}$$

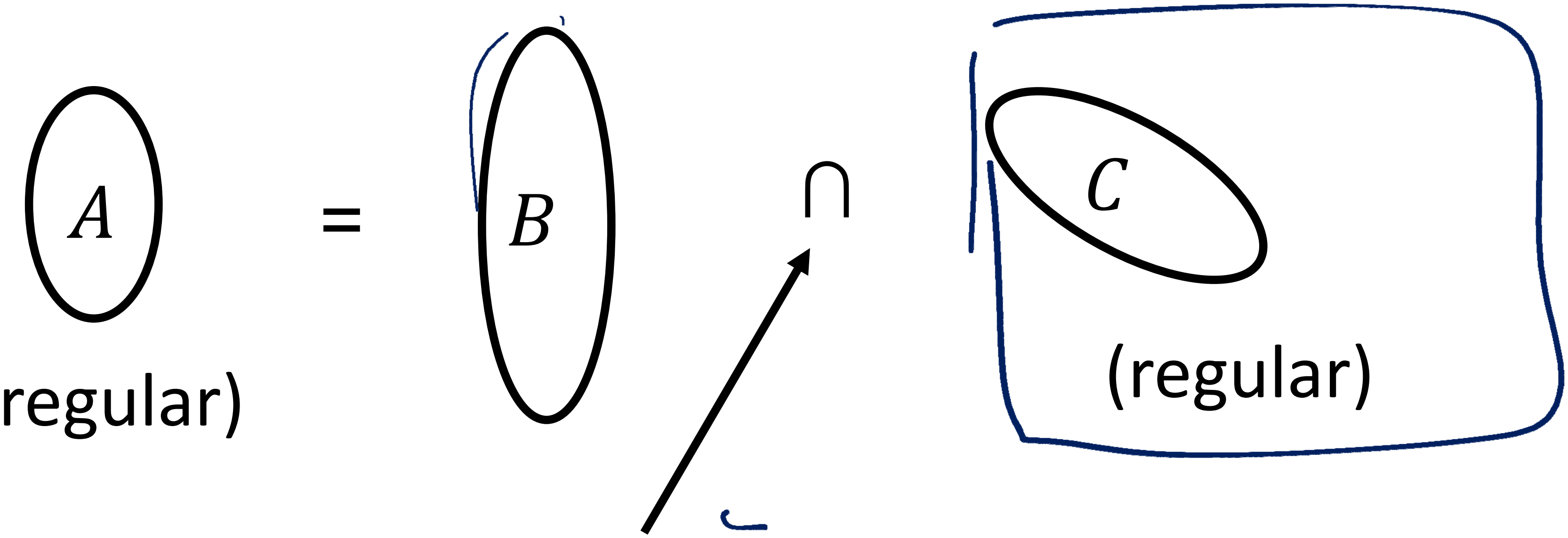
Assume FTSOC BALANCED were regular:

Then $\underbrace{BALANCED \cap L(0^*1^*)}$ is regular (closure under intersection)

$\| \; \{0^n1^n \mid n \geq 0\}$ non-regular $\times$

Using Closure Properties

If A is not regular, we can show a related language B is not regular



A (not regular) $=$ B $\cap$ C (regular)

any of $\{\circ, \cup, \cap\}$ or, for one language, $\{\neg, {}^R, *\}$

By contradiction: If B is regular, then $B \ \cap \ C \ (= \ A)$ is regular.

But A is not regular so neither is B!

Example

$\bar{B}$ = All strings either of the form $0^n 1^n$, $n \geq 0$, or not of the form $0^i 1^j$

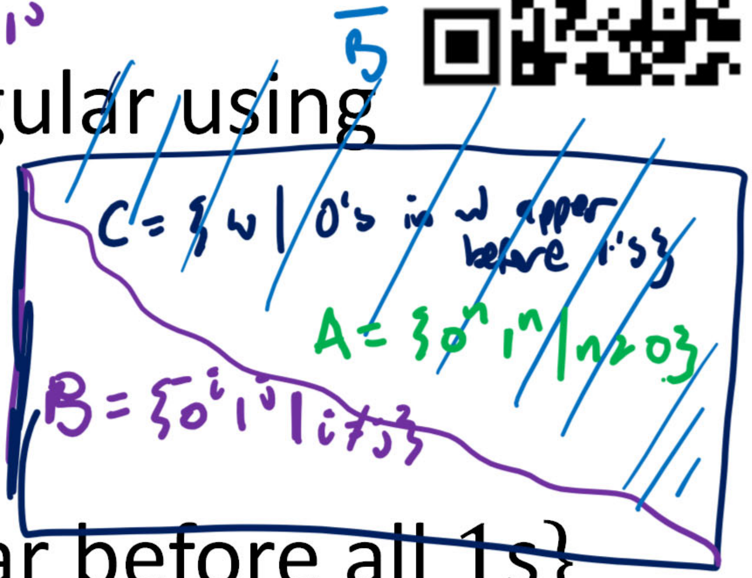
Prove $B = \{0^i 1^j | i \neq j\}$ is not regular using

- Nonregular language

  $A = \{0^n 1^n | n \geq 0\}$ and

- Regular language

  $C = \{w \mid \text{all 0s in } w \text{ appear before all 1s}\}$



Which of the following expresses A in terms of B and C?

a) $A = B \cap C$

b) $A = \bar{B} \cap C = C \setminus B$

c) $A = B \cup C$

d) $A = \bar{B} \cup C$

Proof that B is nonregular

Assume for the sake of contradiction that B is regular

We know: $\underset{\text{non-regular}}{\underline{A}} = \bar{B} \cap \underset{\text{regular}}{\underline{C}}$

Then $\bar{B}$ is regular (closure under complement)

$\Rightarrow$ $\bar{B} \cap C$ is regular (closure under intersection)

$\Rightarrow$ A is regular ($A = \bar{B} \cap C$)

$\times$ since A non-regular

$\Rightarrow$ B must be non-regular.

!DANGER!

Let $B = \{0^i 1^j | i \neq j\}$ and write $B = A \cup C$ where

- Nonregular language

  $A = \{0^i 1^j | i > j \geq 0\}$ and

- Nonregular language

  $C = \{0^i 1^j | j > i \geq 0\}$ and

Does this let us conclude B is nonregular?

No Non-regular languages are **not** closed under union,

i.e. the union of two non-reg. langs. might be regular

e.g. $L_1 = \{0^i 1^j \mid i \neq j,\ i,j \geq 0\}$ $L_1 \cup L_2 = L(0^* 1^*)$ regular

$L_2 = \{0^n 1^n \mid n \geq 0\}$ non-regular

Non-regular languages <u>are</u> closed under complement.

<u>Proof.</u> Let A be non-regular.

Assume FTSOC that $\overline{A}$ were regular

$A = \overline{(\overline{A})}$ ⇒ A is regular (since regular langs. closed under complement)

⨳

⇒ $\overline{A}$ is non-regular

Turing Machines

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

Somewhat more powerful (not in this course):

Pushdown Automata / Context-Free Grammars

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

$\{0^n 1^n \mid n \geq 0\}$

is context-free

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

Diophantine equation:

Given a multivariate polynomial $p(x_1, \ldots, x_n)$

e.g. $p(x_1, x_2) = x_1^2 x_2 + 3x_2 + x_1^3$

w/ integer coefficients

Determine whether $\exists\ x_1, x_2, \ldots, x_n \in \mathbb{Z}$ s.t. $p(x_1, \ldots, x_n) = 0$

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, 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

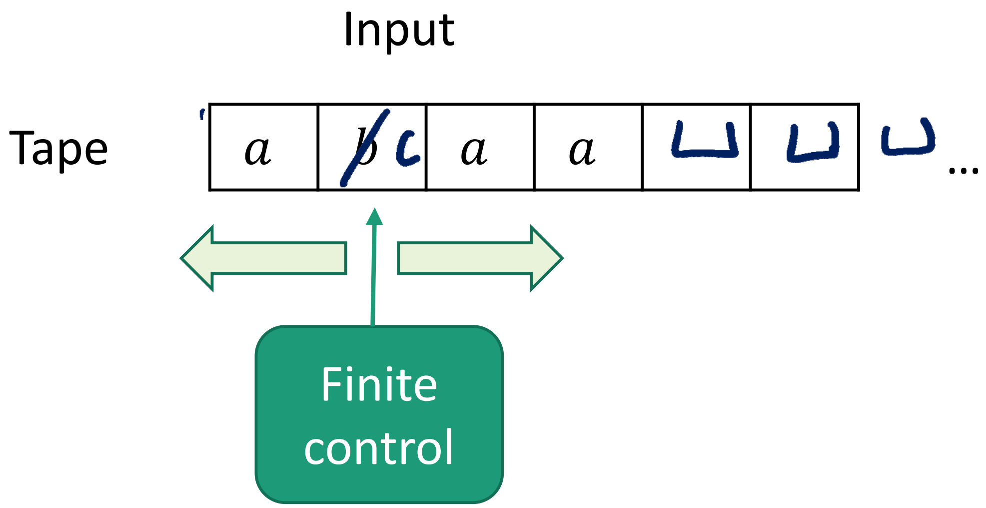
“On computable numbers, with an application to the *Entscheidungsproblem*”

Model of computation: Turing Machine

≈ Finite automata

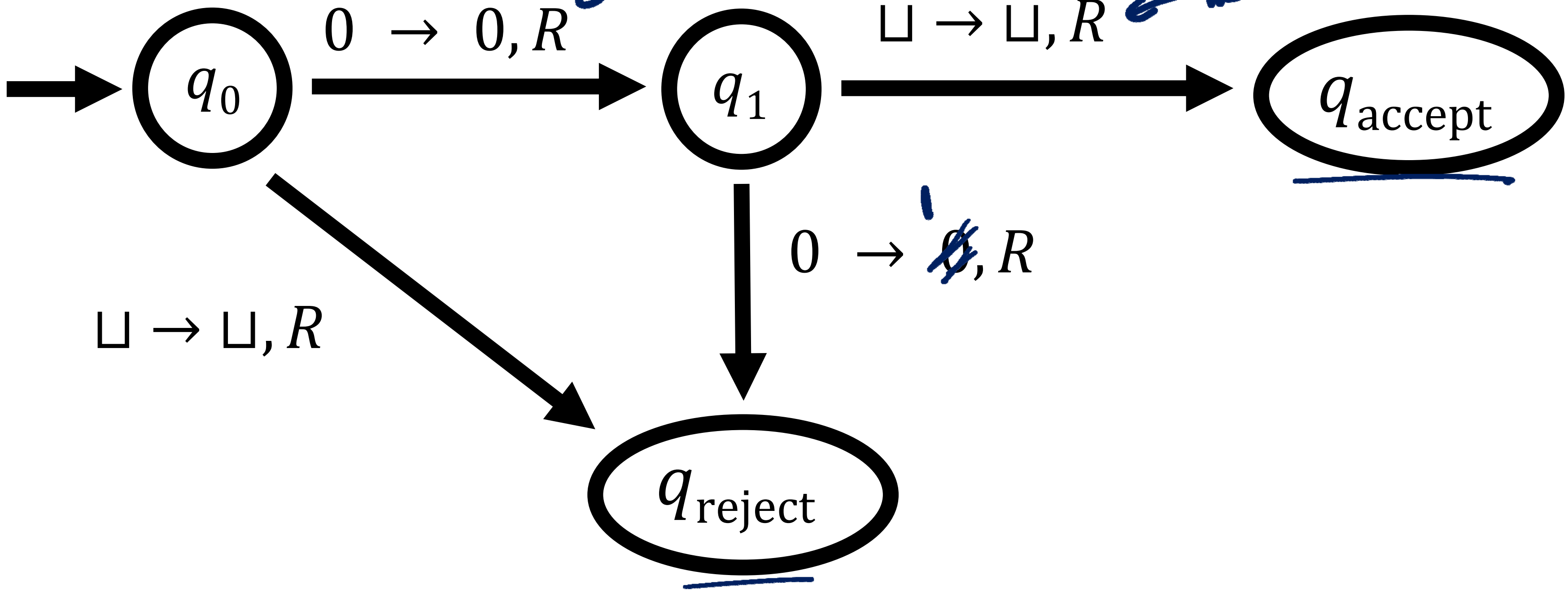
The Turing Machine Model

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



if a 0 is read

write a 0

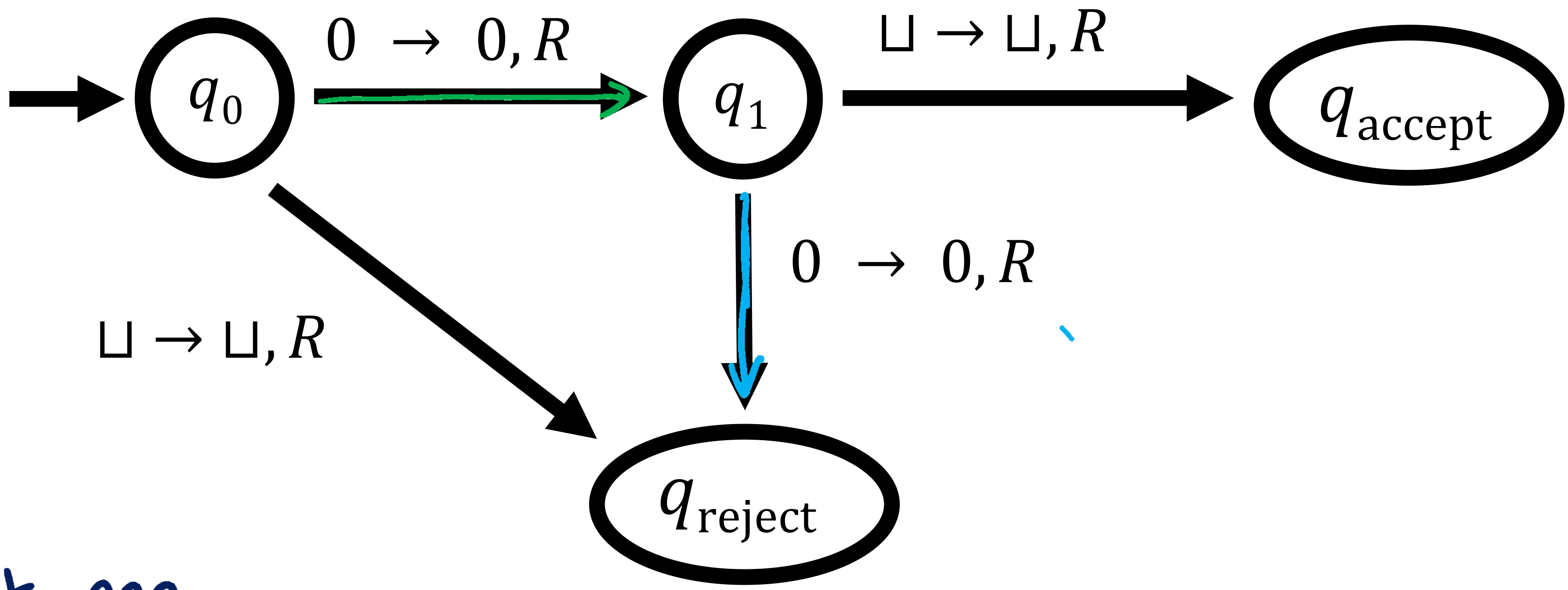
move head one cell right

if a blank is read

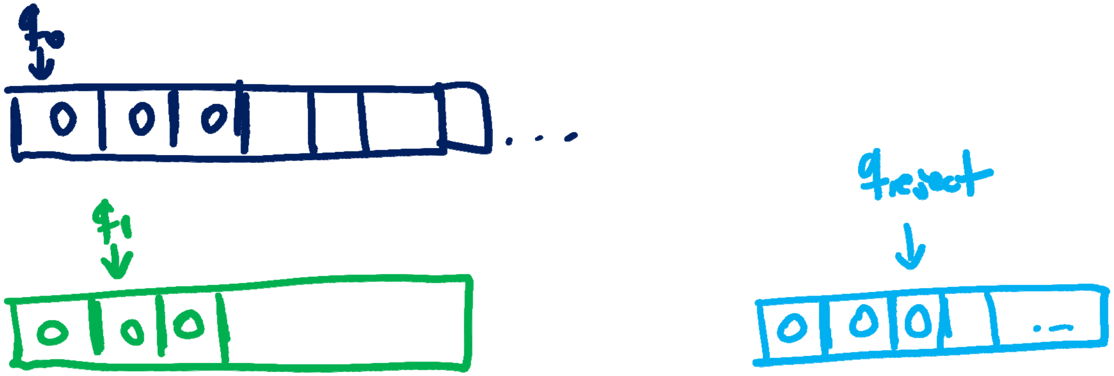
write a blank

move head right

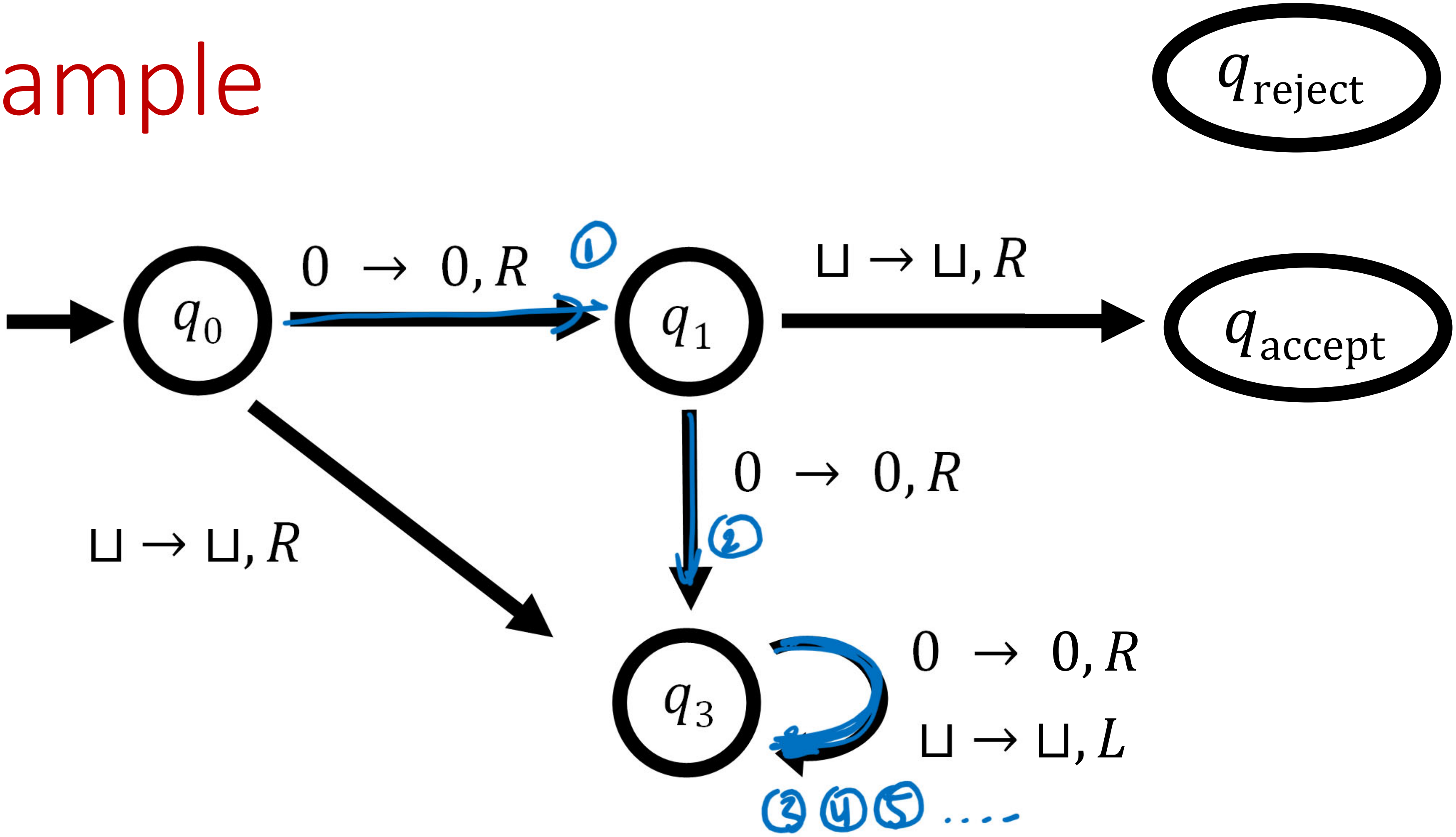
Example



Input 000



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

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

Byte code,
Machine code

Example

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

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

X is an allowable alphabet symbol

Input: $0\cancel{0}\cancel{0}\cancel{0}\cancel{0}\cancel{0}\cancel{0}\cancel{0}$

iteration 1

iteration 2

iteration 3

↳ accept

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

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

Implementation-Level Description

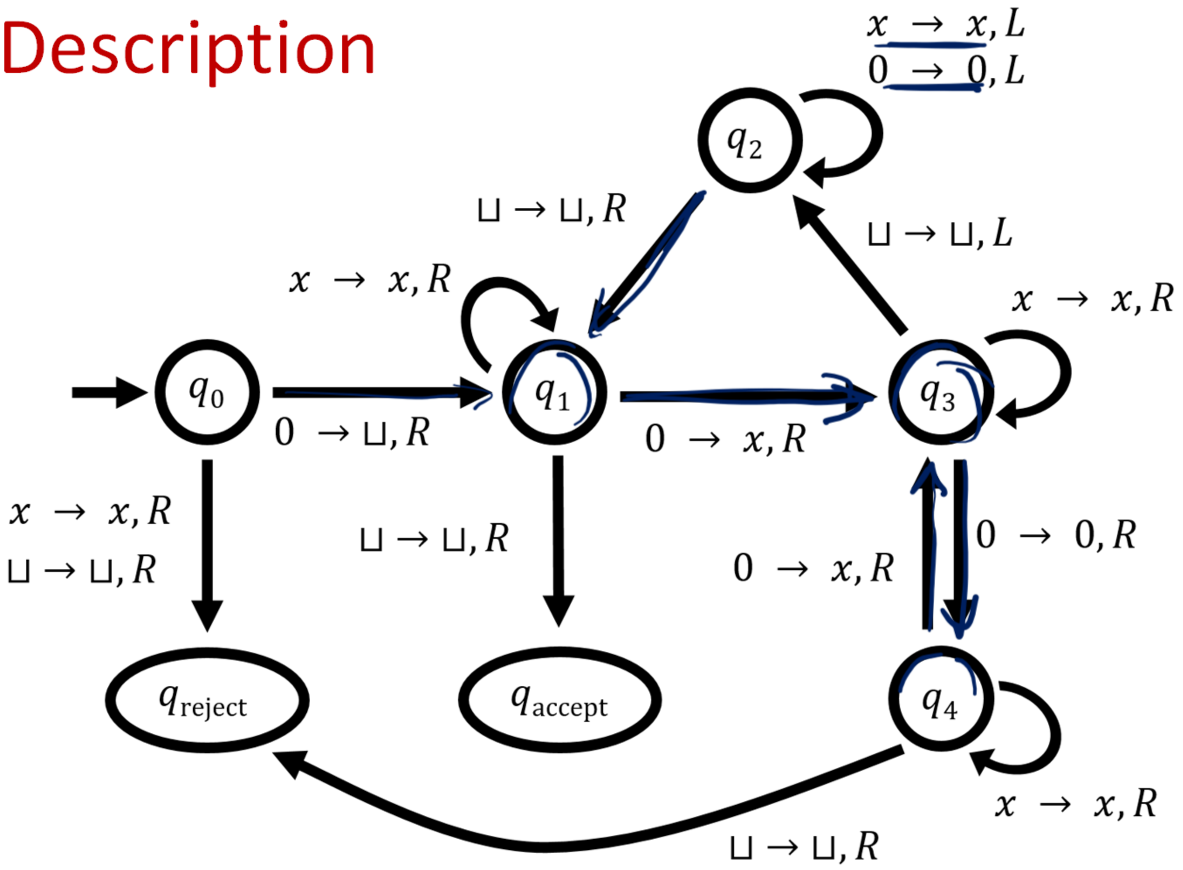
Head location

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

Example

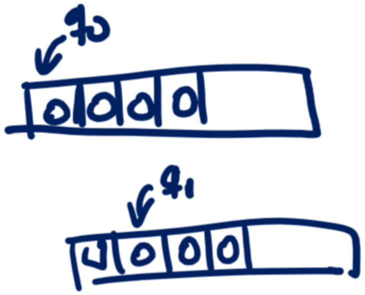
Determine if a string $w \in A = \{0^{2^n} \mid n \geq 0\}$

Low-Level Description



Ex: Input 0000

$q_0\,0000$ means



$\sqcup\, q_1\, 000$

$\sqcup\, x\, q_3\, 00$

$\sqcup\, x\, 0\, q_4\, 0$

$\sqcup\, x\, 0\, x\, q_3$

$\sqcup\, x\, 0\, q_2\, x$

$\sqcup\, x\, q_2\, 0\, x$

$\sqcup\, q_2\, x\, 0\, x$

$q_2\, \sqcup\, x\, 0\, x$

$\sqcup\, q_1\, x\, 0\, x$

$\sqcup\, x\, q_1\, 0\, x$

$\sqcup\, x\, x\, q_3\, x$

$\sqcup\, x\, x\, x\, q_3$

$\sqcup\, x\, x\, q_2\, x$

$\sqcup\, x\, q_2\, x\, x$

$\sqcup\, q_2\, x\, x\, x$

$q_2\, \sqcup\, x\, x\, x$

$\sqcup\, q_1\, x\, x\, x$

$\sqcup\, x\, q_1\, x\, x$

$\sqcup\, x\, x\, q_1\, x$

$\sqcup\, x\, x\, x\, q_1$

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

Accept!