BU CS 332 – Theory of Computation

Lecture 10:

- Turing Machines
- TM Variants

Reading:

Sipser Ch 3.1-3.2

Mark Bun

February 26, 2020

Turing Machines – Motivation

So far in this class we've seen several limited models of computation

Finite Automata / Regular Expressions

- Can do simple pattern matching (e.g., substrings), check parity, addition
- Can't recognize palindromes

Pushdown Automata / Context-Free Grammars

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

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

(ch. 3.3 Sipser)

1900 – Hilbert's Tenth Problem

$x^2 + xyz - 3xy^2 = 0$

Determine if $\exists (x,y,z) \in \mathbb{Z}^3$ satisfying this equation

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?

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)

≈ regexes / CFGs

Alan Turing 1912-1954

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

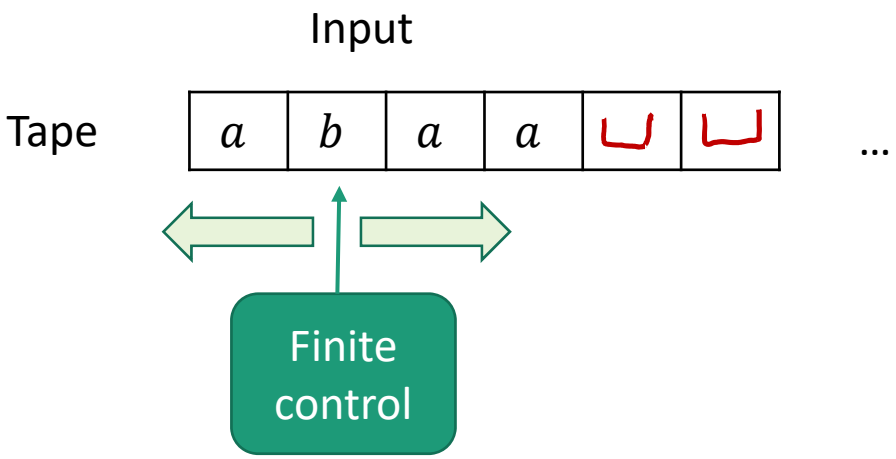
Model of computation: Turing Machine

≈ Automata

Turing Machines

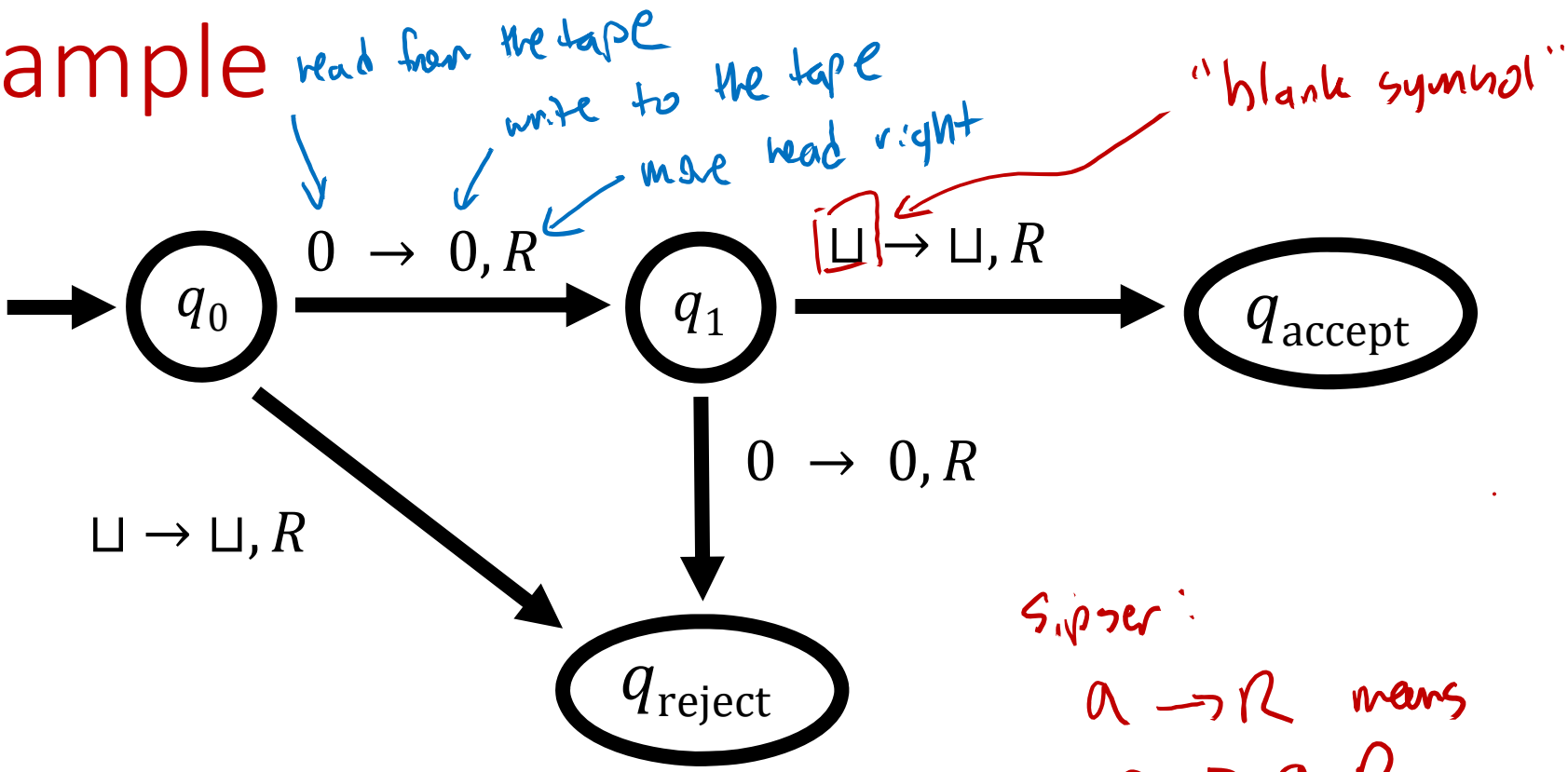
The Basic Turing Machine (TM)

Deterministic



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

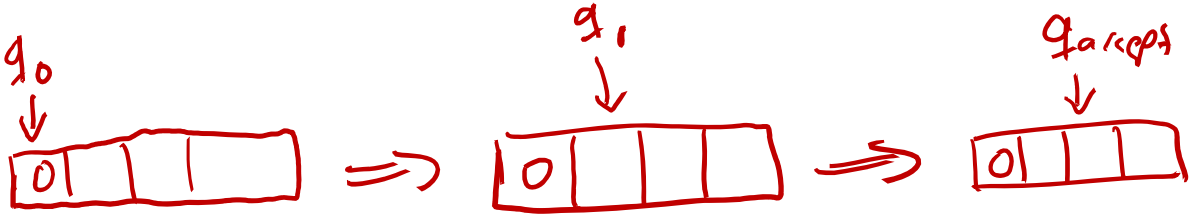
Example



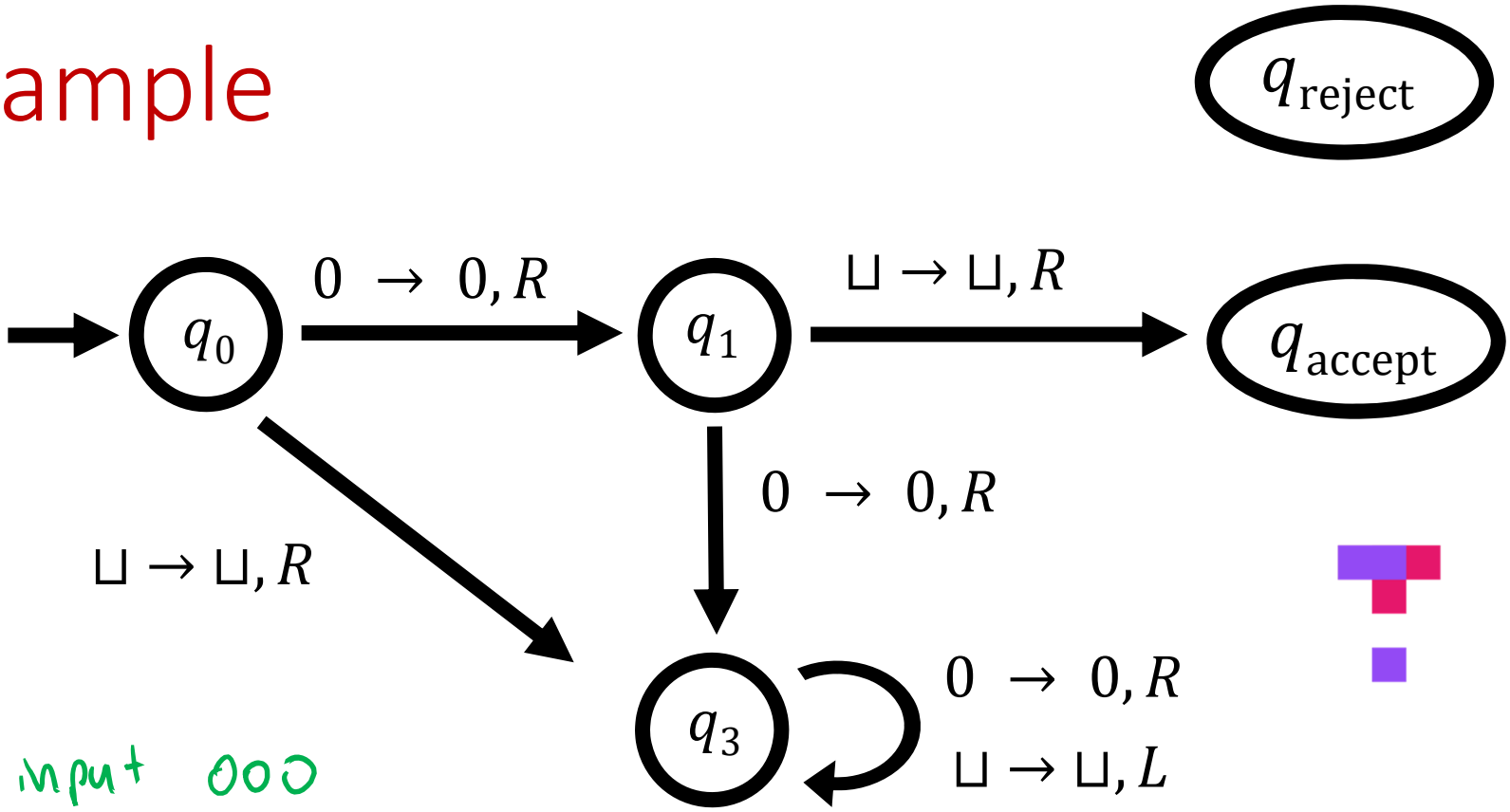
Sipser:

$a \to R$ means $q \to q, R$

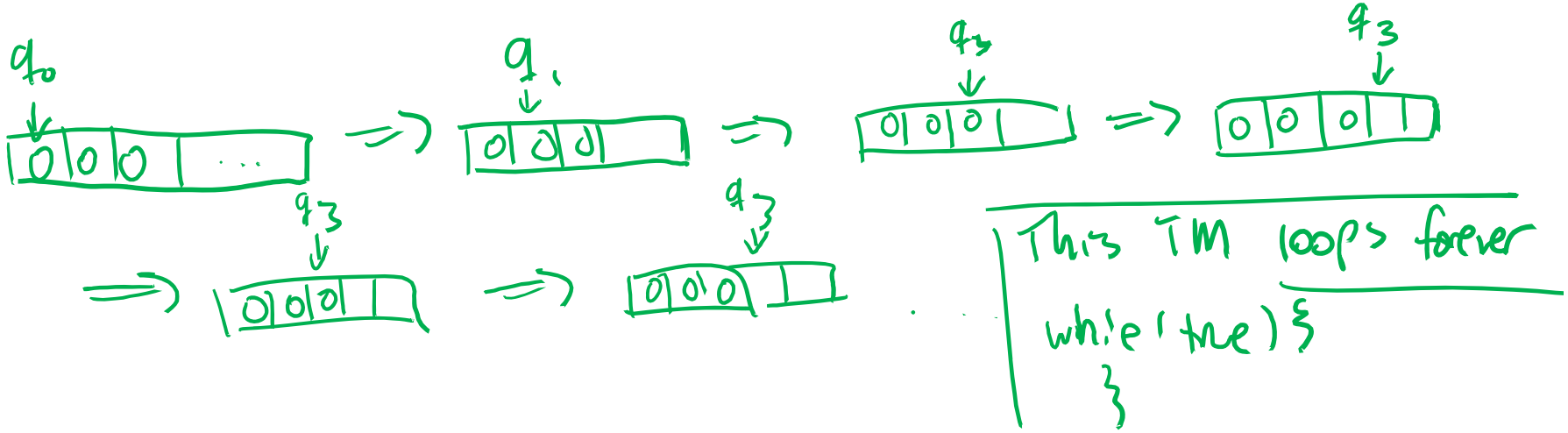
Tape



Example



On input 000



This TM loops forever

```
while(true){
}
```

TMs vs. Finite / Pushdown Automata

Head moves both ways

Unbounded, unrestricted memory

Deterministic

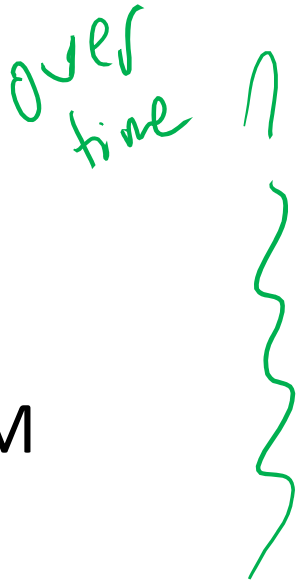
Must explicitly reach accept or reject
(could loop forever)

Halts as soon as it hits accept or reject

Three Levels of Abstraction

High-Level Description

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

Low-Level Description

State diagram or formal specification

Example

Decide if $w \in A = \{0^{2^n} \mid n \geq 0\}$

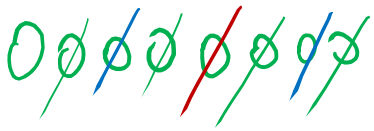
(not a CFL, but can recognize w/ TM)

High-Level Description

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

Repeat the following:

- 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



First step
second step
Third step
Accept

Example

Decide if $w \in A = \{0^{2^n} \mid n \geq 0\}$

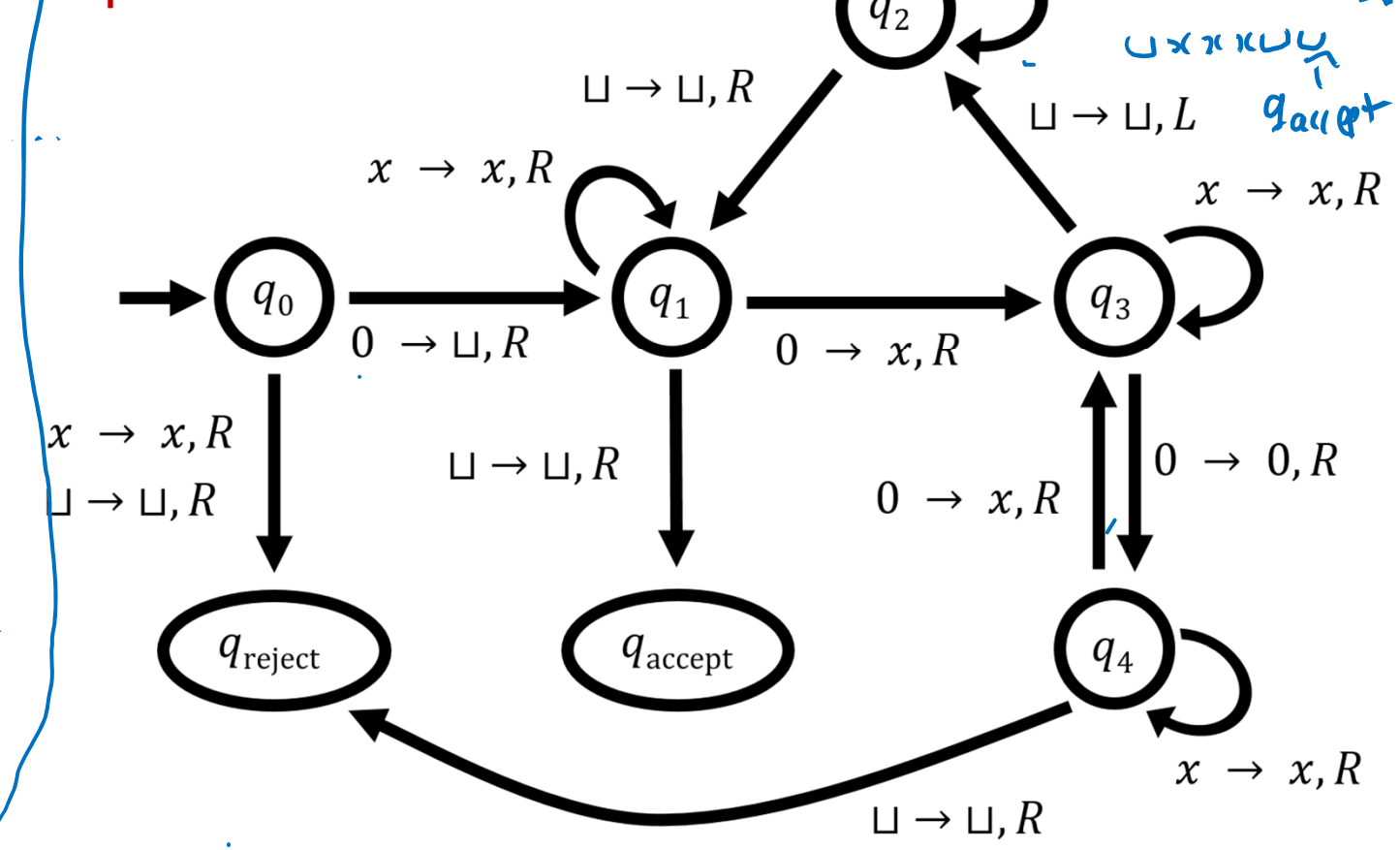
Implementation-Level Description

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

Decide if $w \in A = \{0^{2^n} \mid n \geq 0\}$

Low-Level Description



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 $= \{q_0, q_{\text{accept}}, q_{\text{reject}}, q_1, q_2 \ldots\}$
- Σ is the input alphabet (does **not** include ⊔)
- Γ is the tape alphabet (contains ⊔ and Σ)
- 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

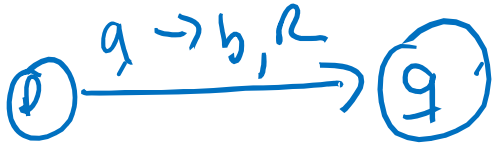
$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

next state, next tape cell, current state, current symbol on tape cell, move tape head left or right

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

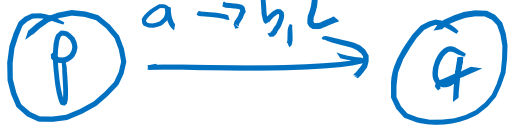
$\delta(p, a) = (q, b, R)$ means:

- 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:

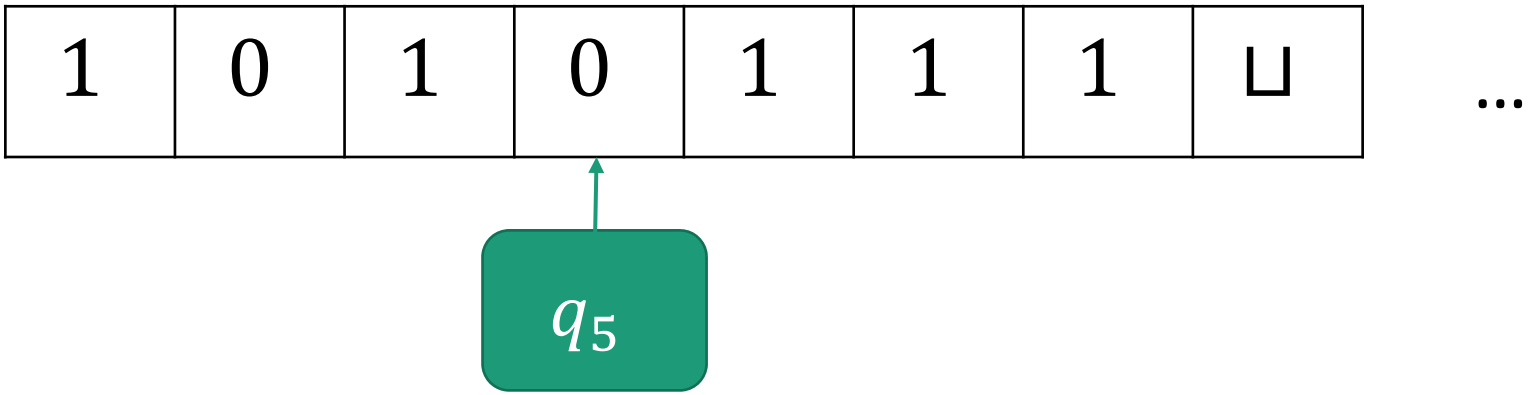
- Replace a with b in current cell
- Transition from state p to state q
- Move tape head left UNLESS we are at left end of tape, in which case don't move



Configuration of a TM

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

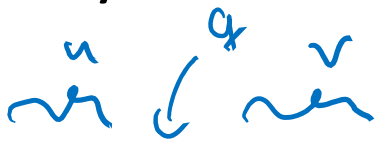
$1\ 0\ 1\ q_5\ 0\ 1\ 1\ 1$



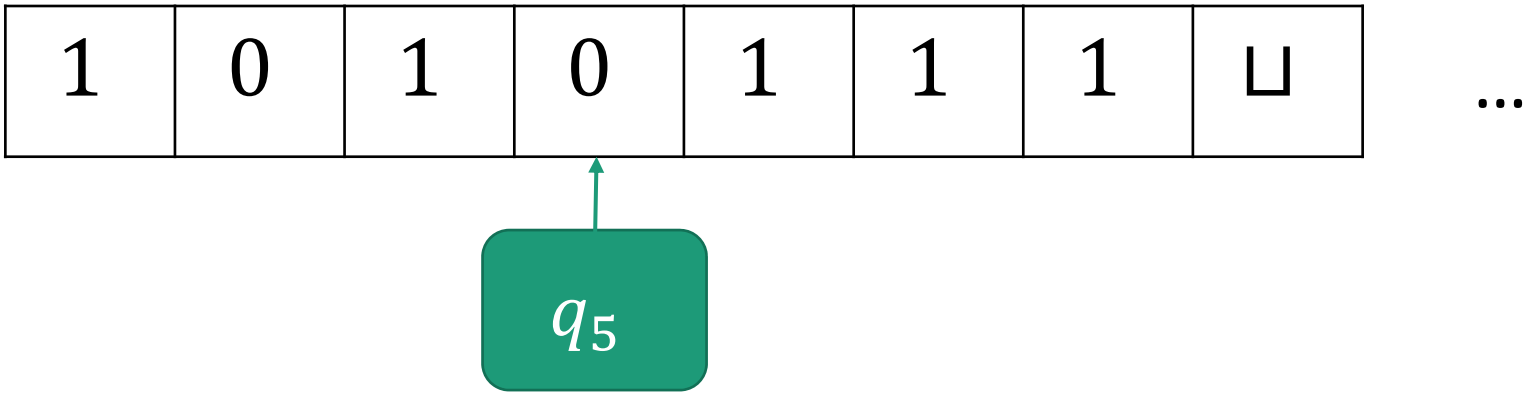
Configuration of a TM: Formally

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

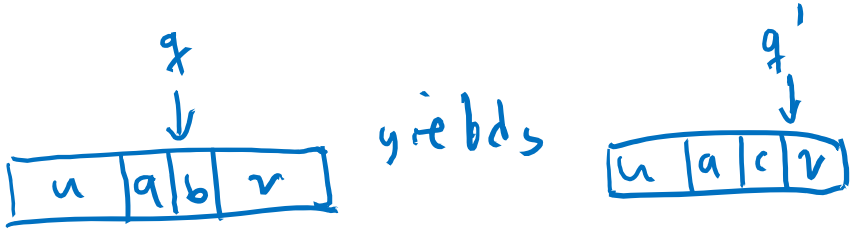


Example: $101q_50111$



How a TM Computes on input w

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 $\delta(q, b) = (q', c, L)$
- $q\ bv$ yields $q'\ cv$ if $\delta(q, b) = (q', c, L)$

Accepting configuration: $q = q_{\text{accept}}$

Rejecting configuration: $q = q_{\text{reject}}$