

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

1900 – Hilbert’s Tenth Problem

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.



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)



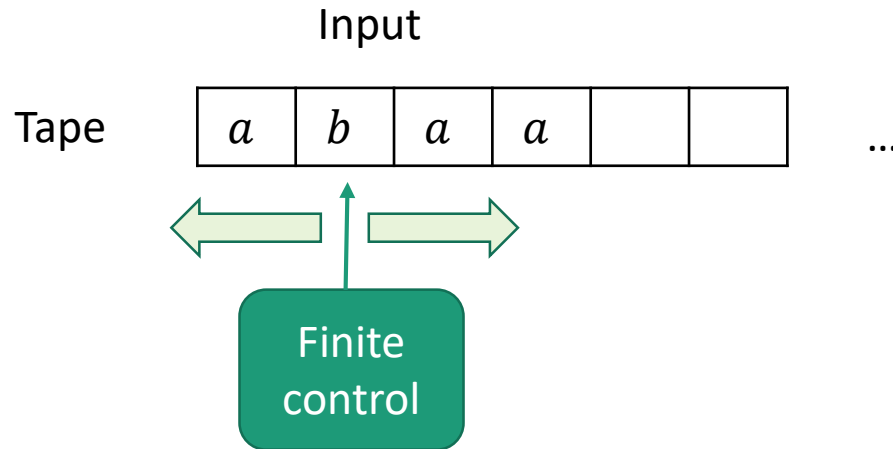
Alan Turing 1912-1954

"On computable numbers, with an application to the *Entscheidungsproblem*"

Model of computation: Turing Machine

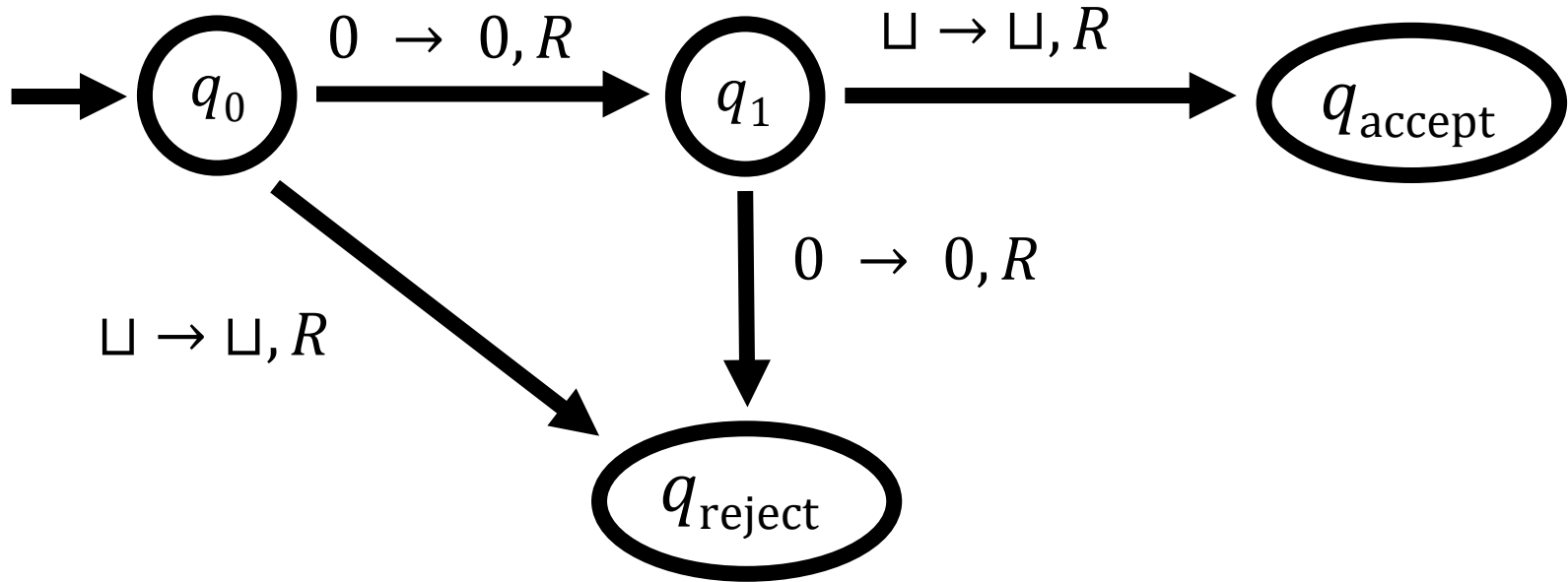
Turing Machines

The Basic Turing Machine (TM)

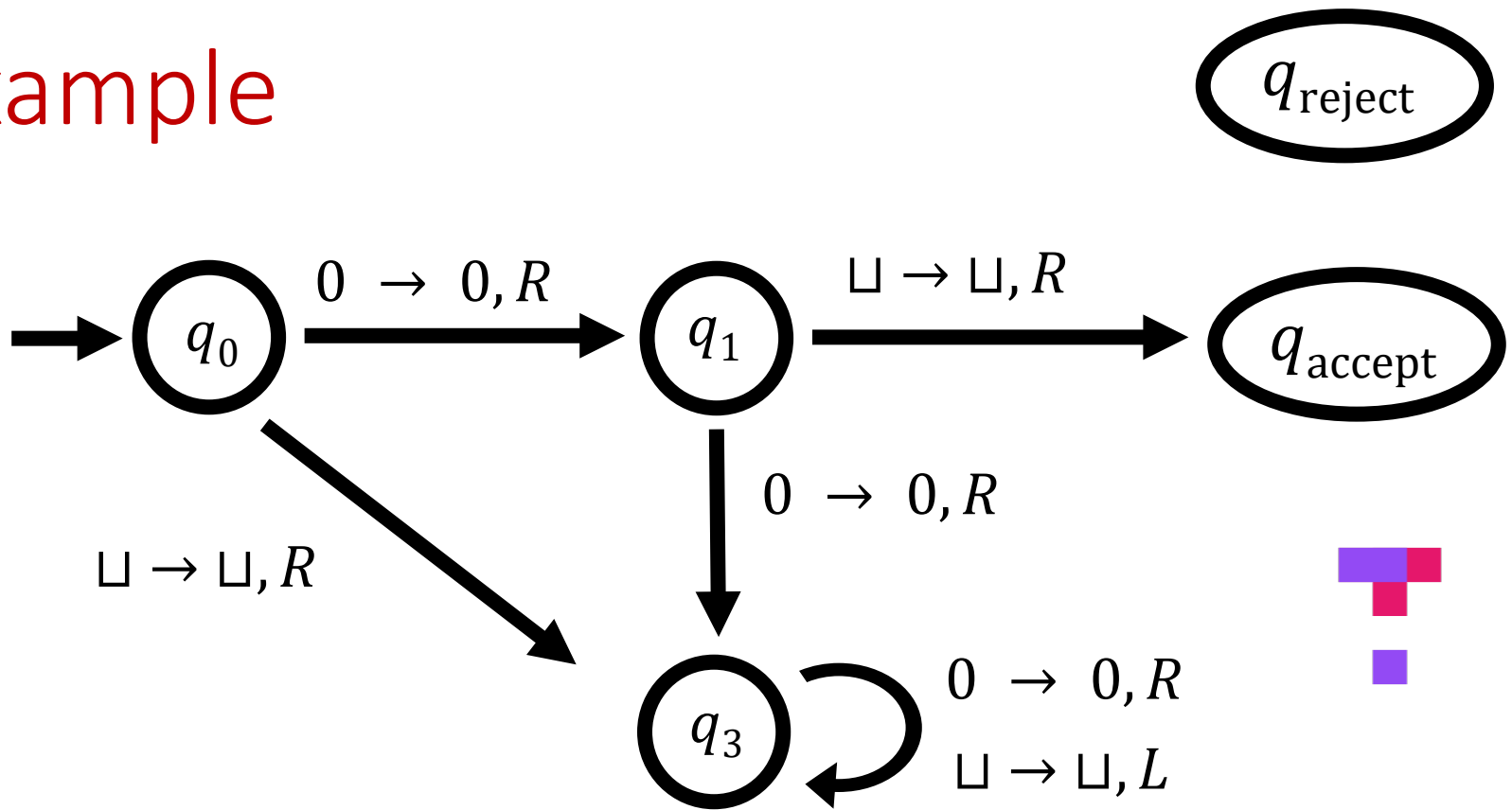


- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts when control reaches “accept” or “reject” state

Example



Example



TMs vs. Finite / Pushdown Automata



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

High-Level Description

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

Example

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

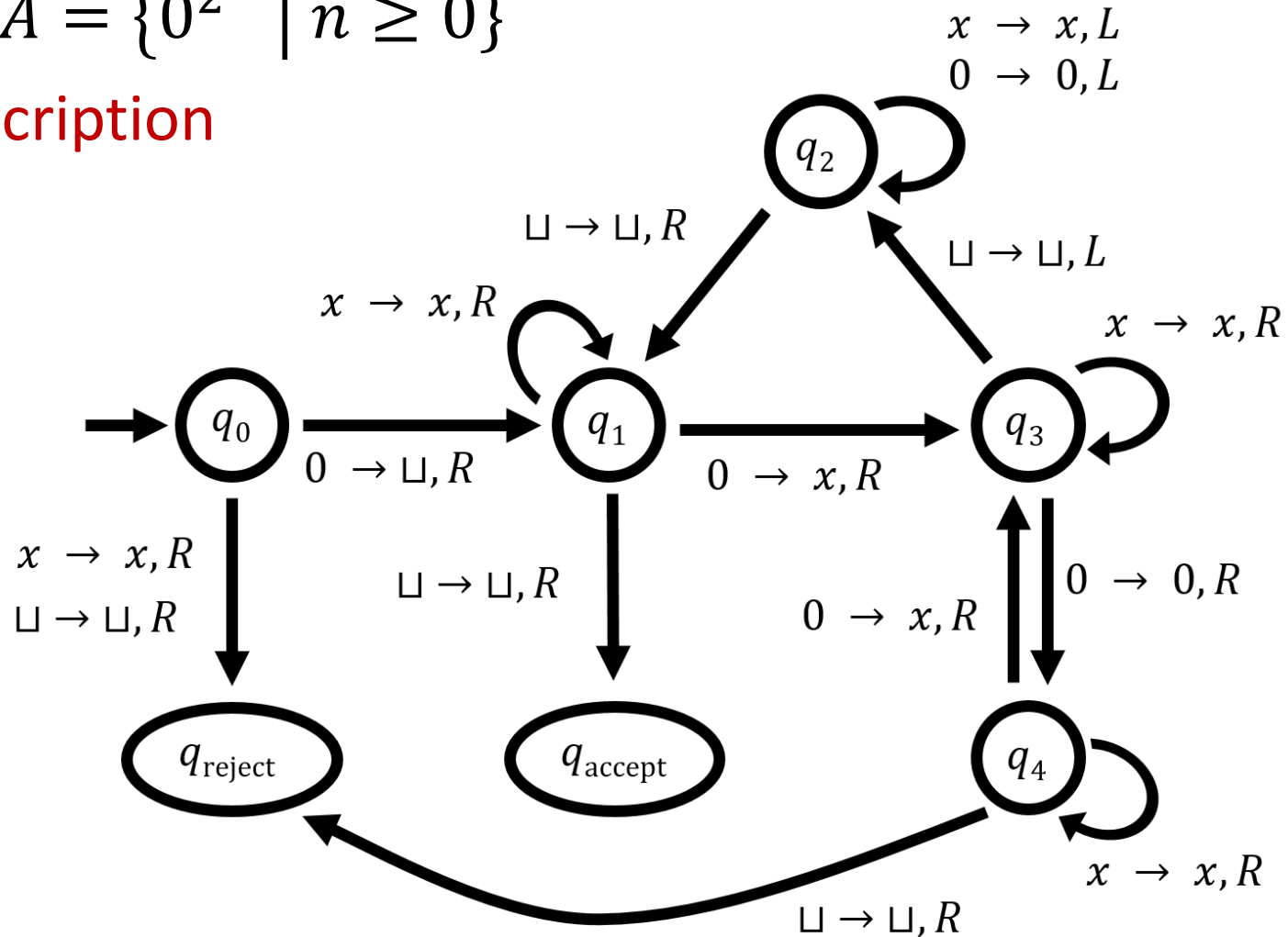
Implementation-Level Description

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
- Σ is the input alphabet (does **not** include \sqcup)
- Γ is the tape alphabet (contains \sqcup 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

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

L means “move left” and R means “move right”

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

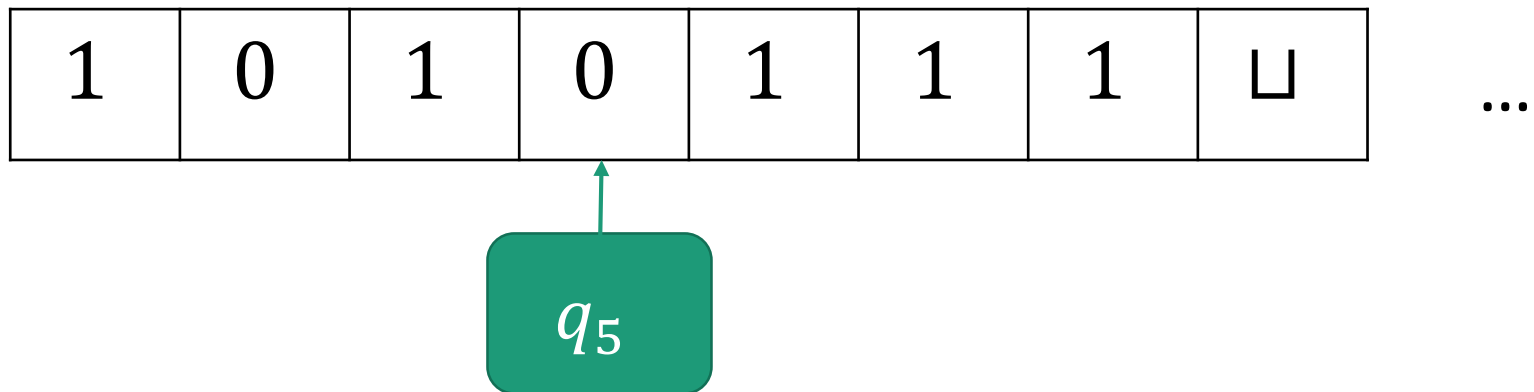
- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right

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

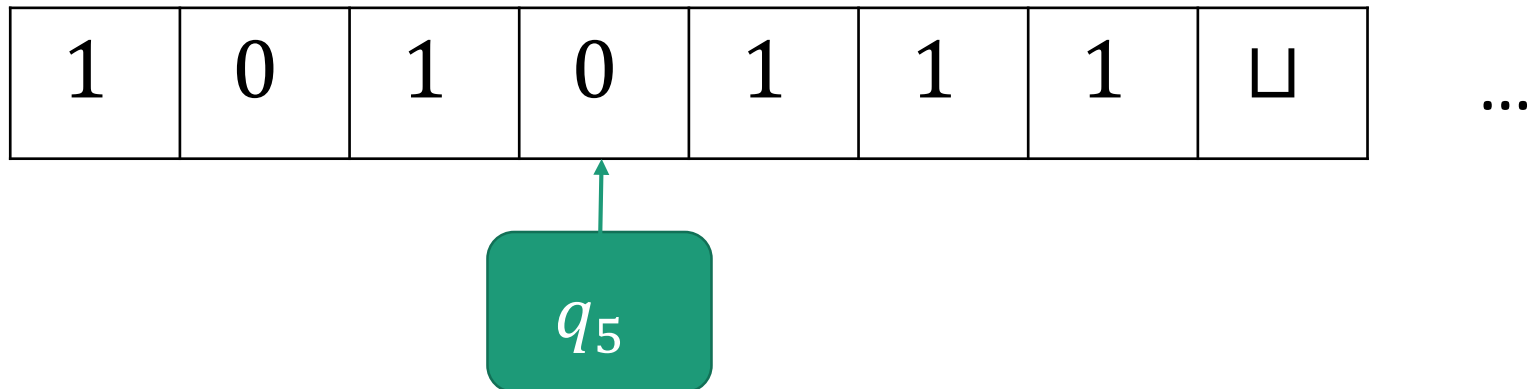


Configuration of a TM: Formally

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

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

Example: $101q_50111$



How a TM Computes

Start configuration: q_0w

One step of computation:

- $uaqbv$ yields $uacq'v$ if $\delta(q, b) = (q', c, R)$
- $uaqbv$ yields $uq'acv$ if $\delta(q, b) = (q', c, L)$
- qbv yields $q'cv$ if $\delta(q, b) = (q', c, L)$

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

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



How a TM Computes

M **accepts** input w if there is a sequence of configurations C_1, \dots, C_k such that:

- $C_1 = q_0 w$
- C_i yields C_{i+1} for every i
- C_k is an accepting configuration

$L(M)$ = the set of all strings w which M accepts

A is **Turing-recognizable** if $A = L(M)$ for some TM M :

- $w \in A \implies M$ halts on w in state q_{accept}
- $w \notin A \implies M$ halts on w in state q_{reject} **OR**
 M runs forever

Recognizers vs. Deciders

$L(M)$ = the set of all strings w which M accepts

A is **Turing-recognizable** if $A = L(M)$ for some TM M :

- $w \in A \implies M$ halts on w in state q_{accept}
- $w \notin A \implies M$ halts on w in state q_{reject} **OR**
 M runs forever

A is **(Turing-)decidable** if $A = L(M)$ for some TM M

which halts on every input

- $w \in A \implies M$ halts on w in state q_{accept}
- $w \notin A \implies M$ halts on w in state q_{reject}

Back to Hilbert's Tenth Problem

Computational Problem: Given a Diophantine equation, does it have a solution over the integers?

$L =$

- L is Turing-recognizable

- L is **not** decidable (1949-70)



TM Variants

How Robust is the TM Model?

Does changing the model result in different languages being recognizable / decidable?

So far we've seen...

- We can require that FAs/PDAs have a single accept state
- (CFGs can always be put in Chomsky Normal Form)
- Adding nondeterminism does not change the languages recognized by finite automata

Turing machines have an **astonishing** level of robustness

Extensions that do not increase the power of the TM model

- TMs that are allowed to “stay put” instead of moving left or right

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

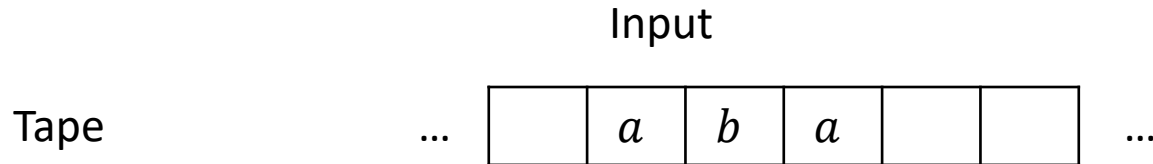
Proof that TMs with “stay put” are no more powerful:

Simulation: Convert any TM M with “stay put” into an equivalent TM M' without

Replace every “stay put” instruction in M with a move right instruction, followed by a move left instruction in M'

Extensions that do not increase the power of the TM model

- TMs with a 2-way infinite tape, unbounded left to right



Proof that TMs with 2-way infinite tapes are no more powerful:

Simulation: Convert any TM M with 2-way infinite tape into a 1-way infinite TM M' with a “two-track tape”

TMs are equivalent to...

- TMs with “stay put”
- TMs with 2-way infinite tapes
- Multi-tape TMs
- Nondeterministic TMs
- Enumerators
- Finite automata with access to an unbounded queue = 2-stack PDAs
- Primitive recursive functions
- Cellular automata
- ...

