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

Somewhat more powerful (not in this course):

Pushdown Automata / Context-Free Grammars

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

Turing Machines – Motivation

Goal:

Define a model of computation that is

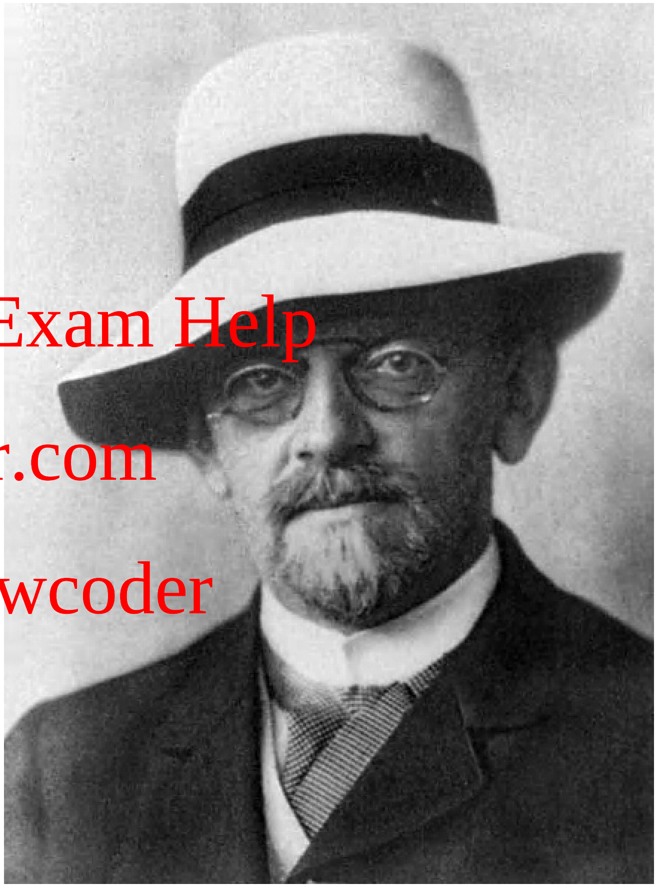
1) General purpose. Captures all algorithms that can be implemented in any programming language.

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*

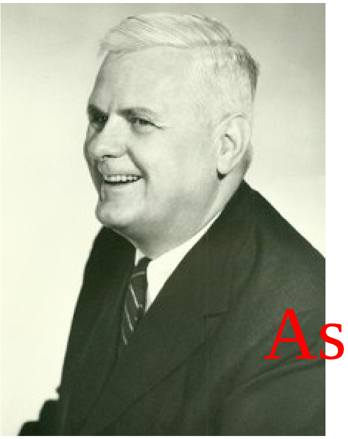
Wilhelm Ackermann 1896-1962

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)

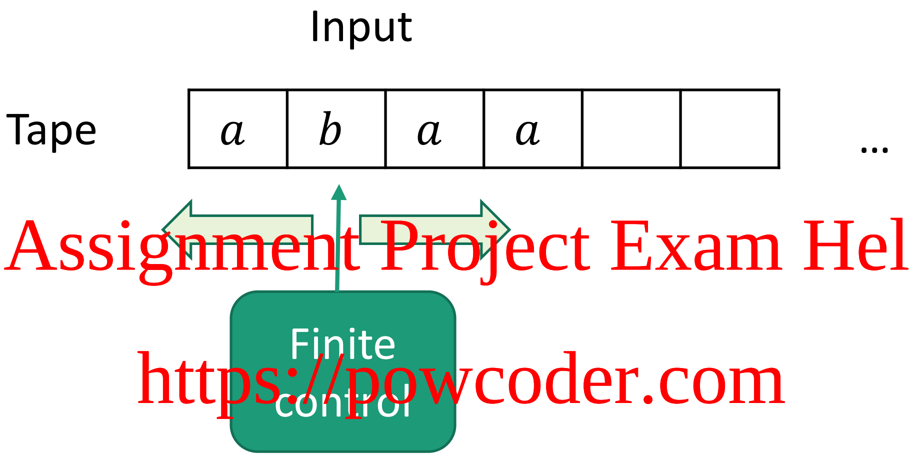
Alan Turing 1912-1954

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

Model of computation: Turing Machine

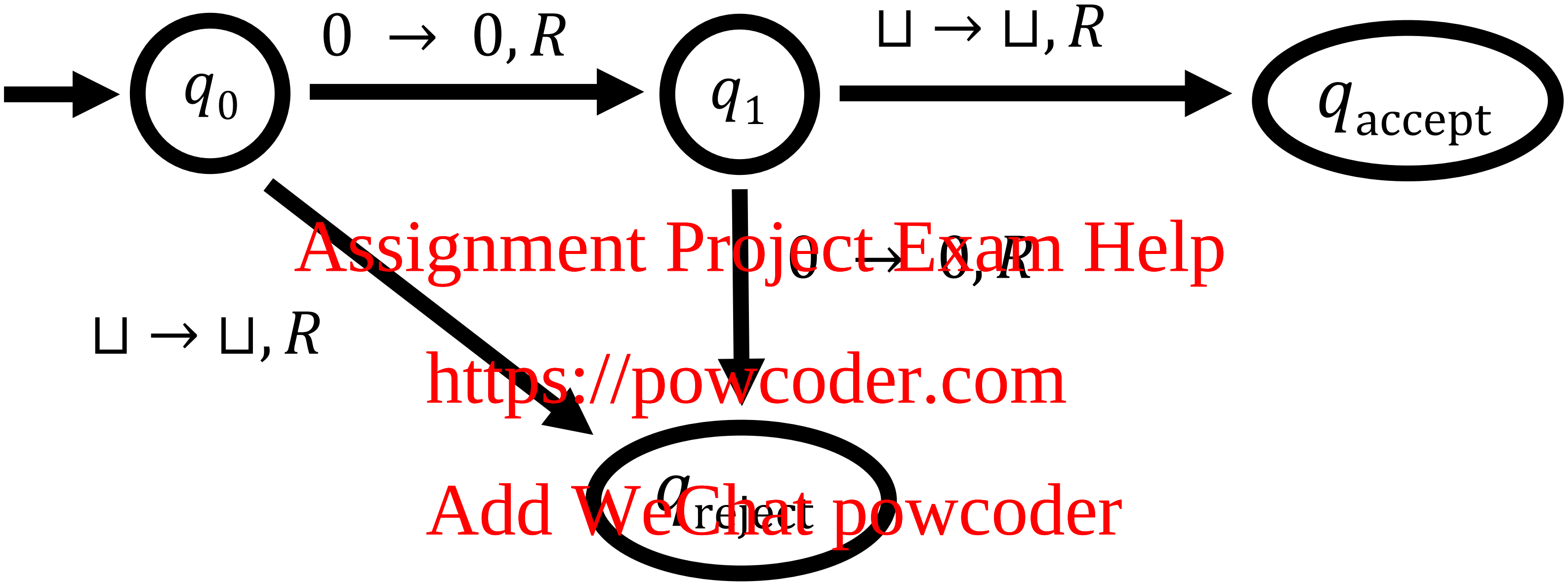
Turing Machines

The Basic Turing Machine (TM)

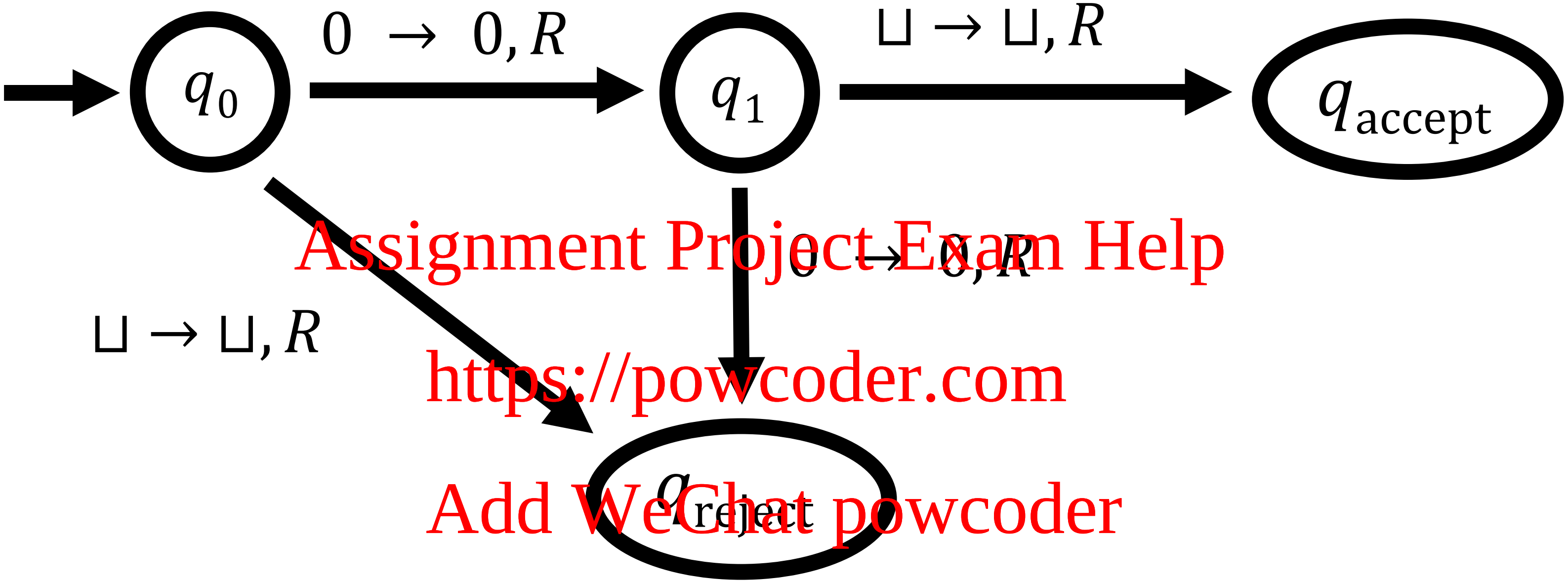


- Input is written on an infinitely long tape
- Head can both read and write, and move in both directions
- Computation halts as soon as control reaches “accept” or “reject” state

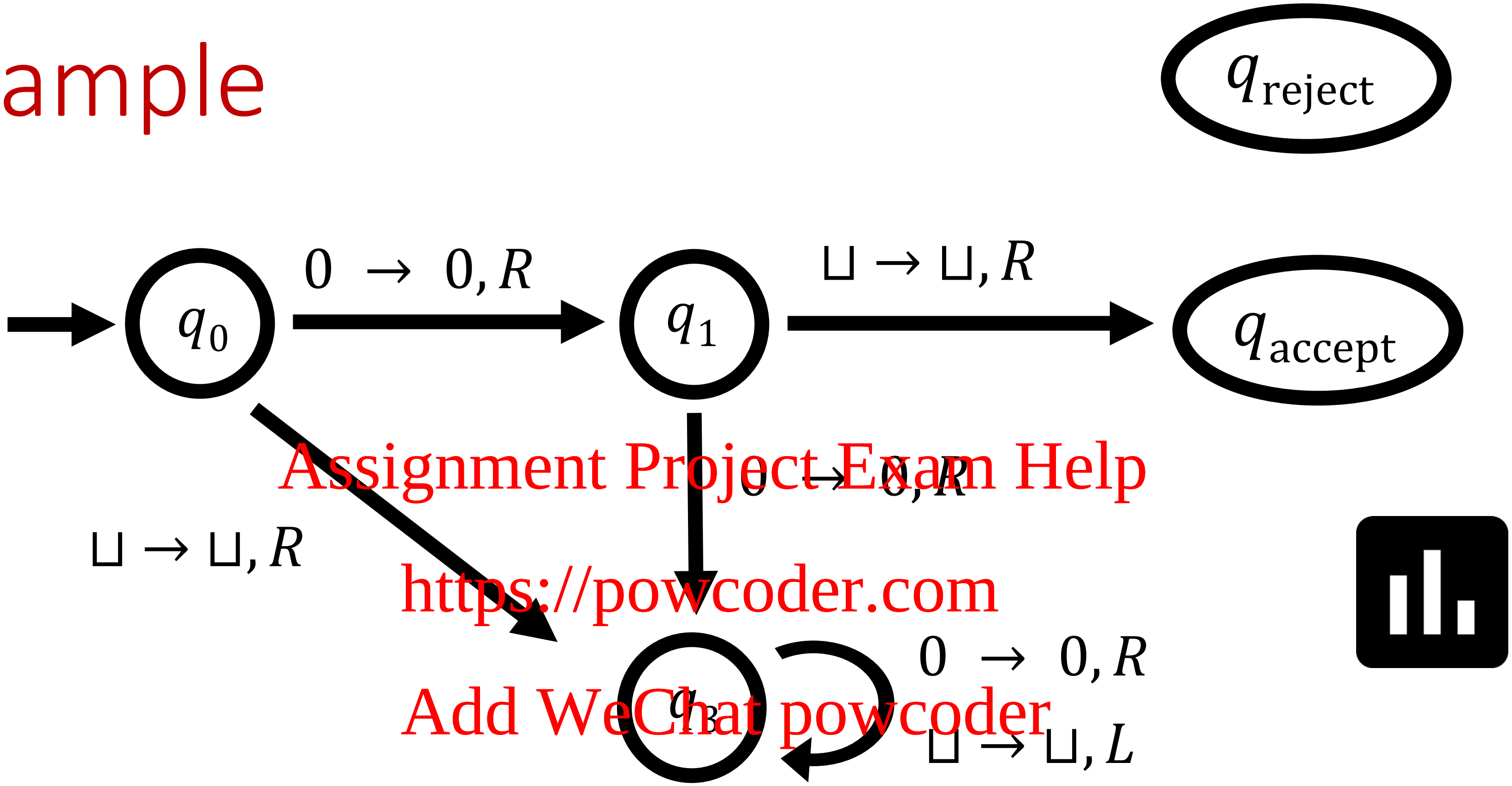
Example


q_0
$0 \rightarrow 0, R$
q_1
$\sqcup \rightarrow \sqcup, R$
q_{accept}
$0 \rightarrow 0, R$
$\sqcup \rightarrow \sqcup, R$
q_{reject}

Example


q_0
$0 \to 0, R$
q_1
$\sqcup \to \sqcup, R$
q_{accept}
$0 \to 0, R$
$\sqcup \to \sqcup, R$
q_{reject}

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

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

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

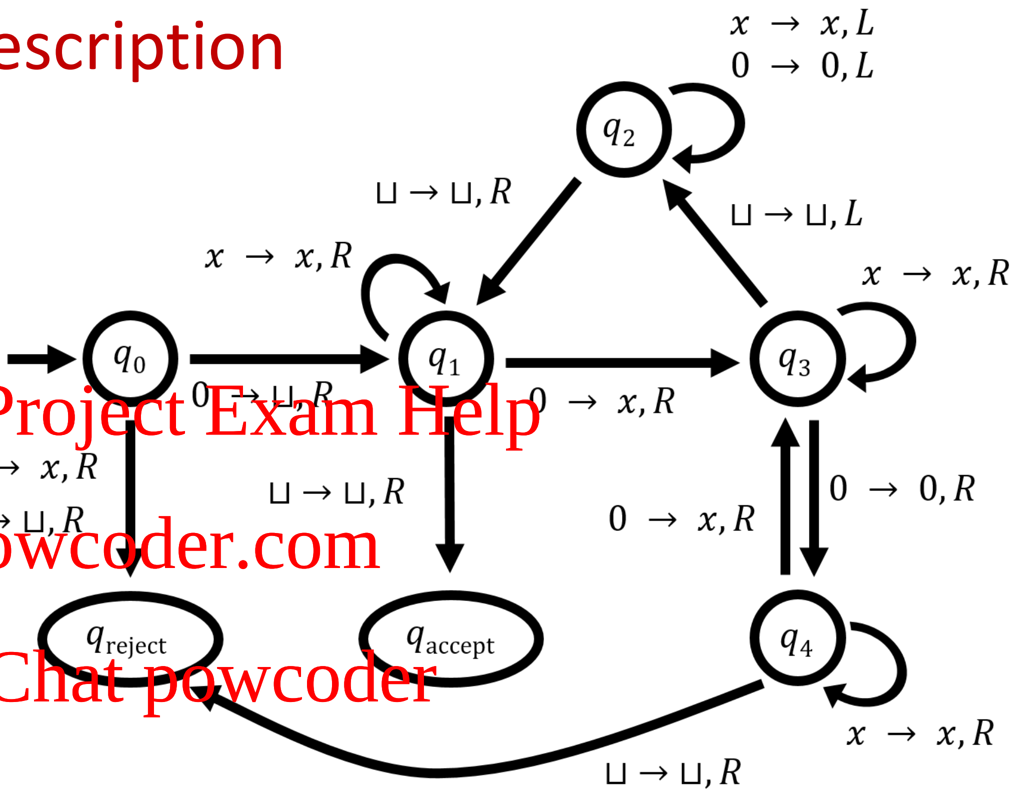
Implementation-Level Description

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

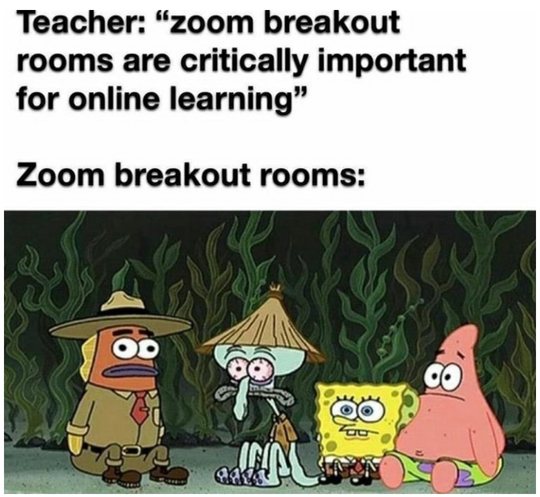
Example

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

Low-Level Description



TMs vs. Finite Automata


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

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

…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:

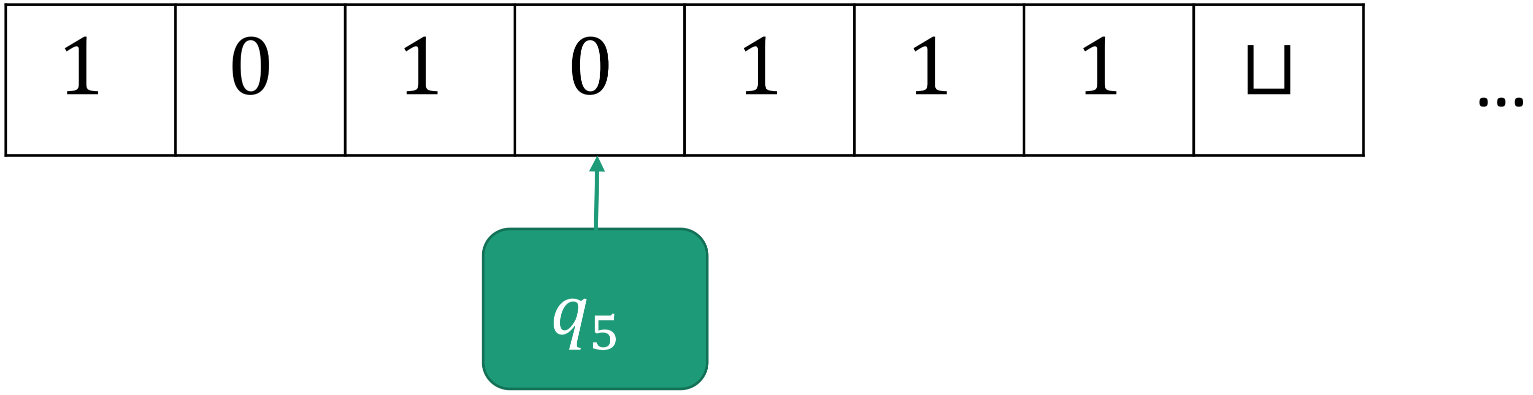
- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right

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

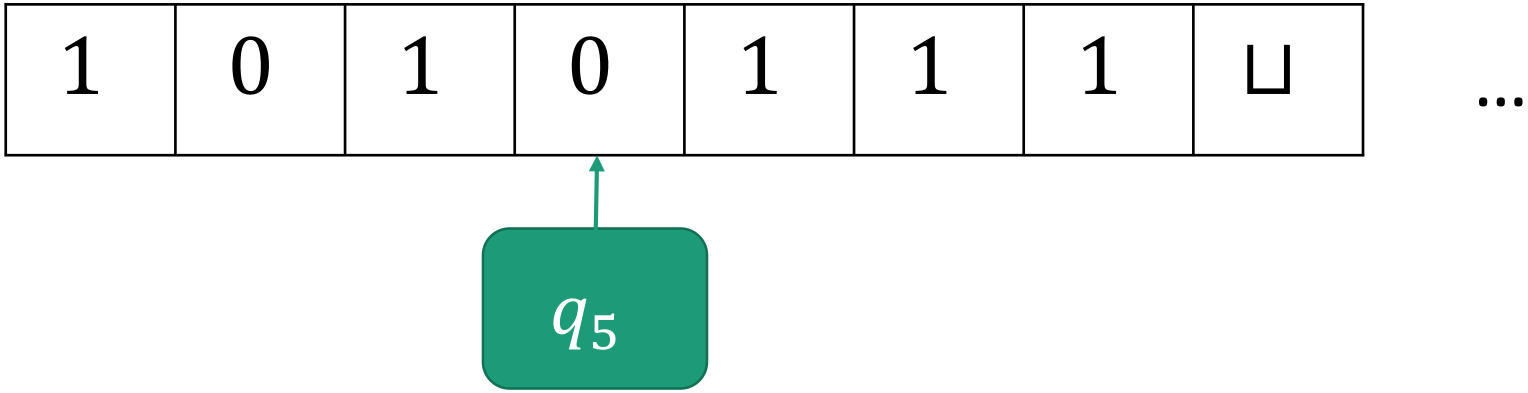


Configuration of a TM: Formally

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

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

Example: $101\, q_5\, 0111$



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 $\delta(q, b) = (q', c, L)$?

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 $u\ q'\ ac\ v$ 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: $q = q_{\text{reject}}$

How a TM Computes

M accepts input w if there is a sequence of configurations $C_1, \ldots, C_k$ 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)