BU CS 332 – Theory of Computation

https://forms.gle/wk3dp8JPCnJ1cYRN7

Lecture 10:

- Turing Machines
- TM Variants and Closure Properties

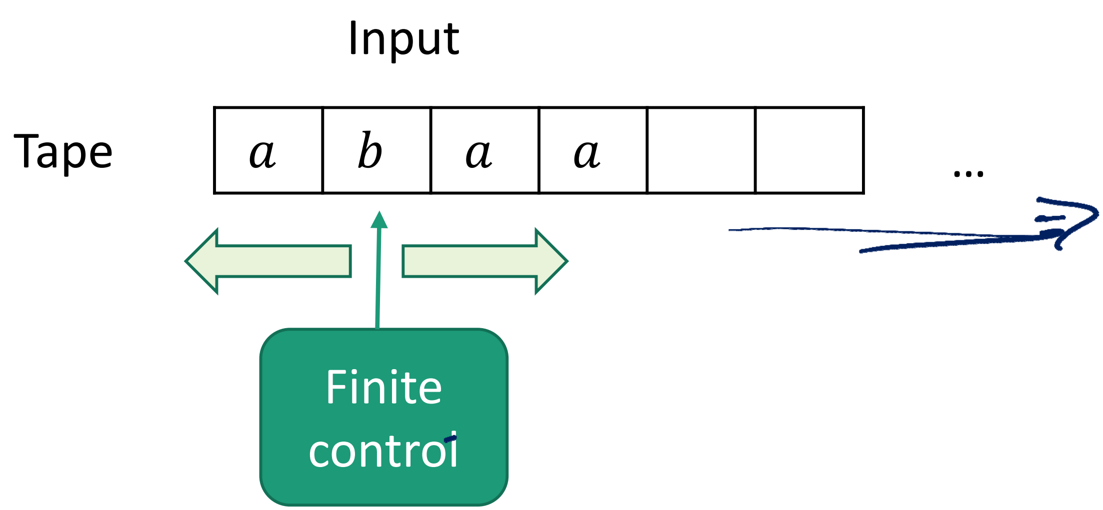
Reading:

Sipser Ch 3.1-3.3

Mark Bun

February 26, 2025

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

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

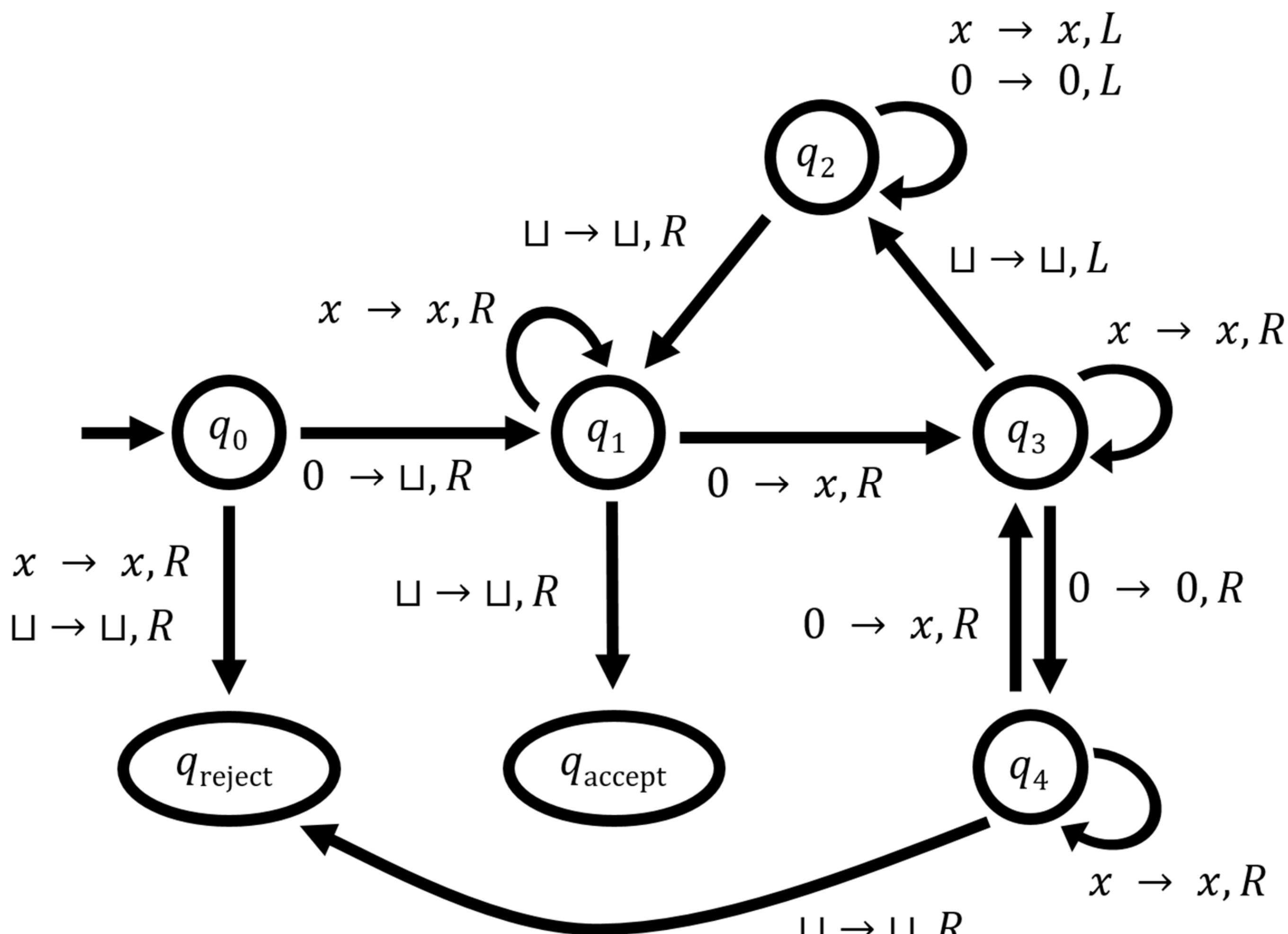
Head location

1. While moving the tape head left-to-right:
   a) Cross off every other 0 ie. 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



Differences between TMs and Finite Automata

- Finite automata can only use "states" for memory
  vs. TM has both states <u>and</u> tape for memory

- TMs can move both left and right
  vs. FA can only move right

  - TMs can solve problems FA cannot solve, e.g. involving counting

- TMs can read/write additional chars beyond chars in input <u>alphabet</u>
  e.g. ⊔, X

  - TMs as we've defined them are deterministic

- TMs halt immediately when reaching accept or reject state
  vs. FA which halt @ right end of input string

- FA can only read, vs. TMs can read and write

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 $\sqcup$) e.g. $\Sigma = \{0\}$ or $\Sigma = \{0, 1\}$
- Γ is the tape alphabet (contains $\sqcup$ and Σ) e.g. $\Gamma = \{0, 1, \sqcup, X\}$
- 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

new symbol to write under head

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

curr. state

curr. symbol under head

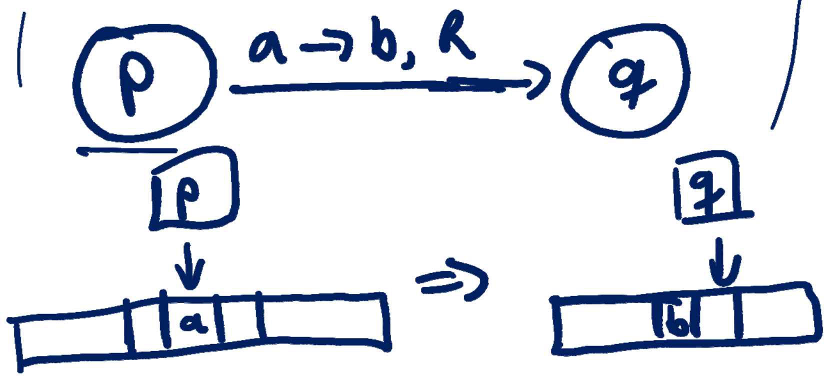
next state

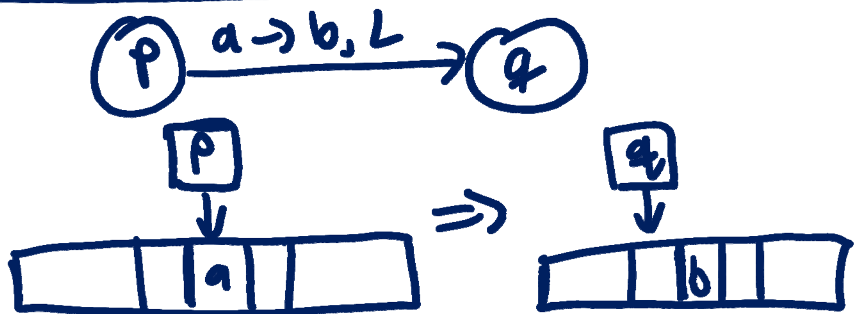
movement instruction (L = left, R = right)

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

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

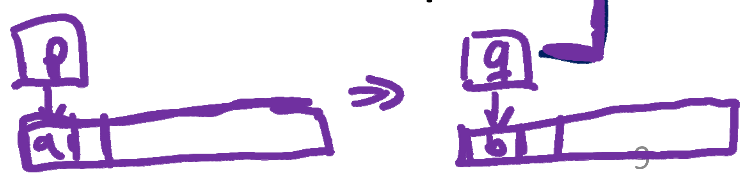
- Replace a with b in current cell
- Transition from state p to state q
- Move tape head right





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

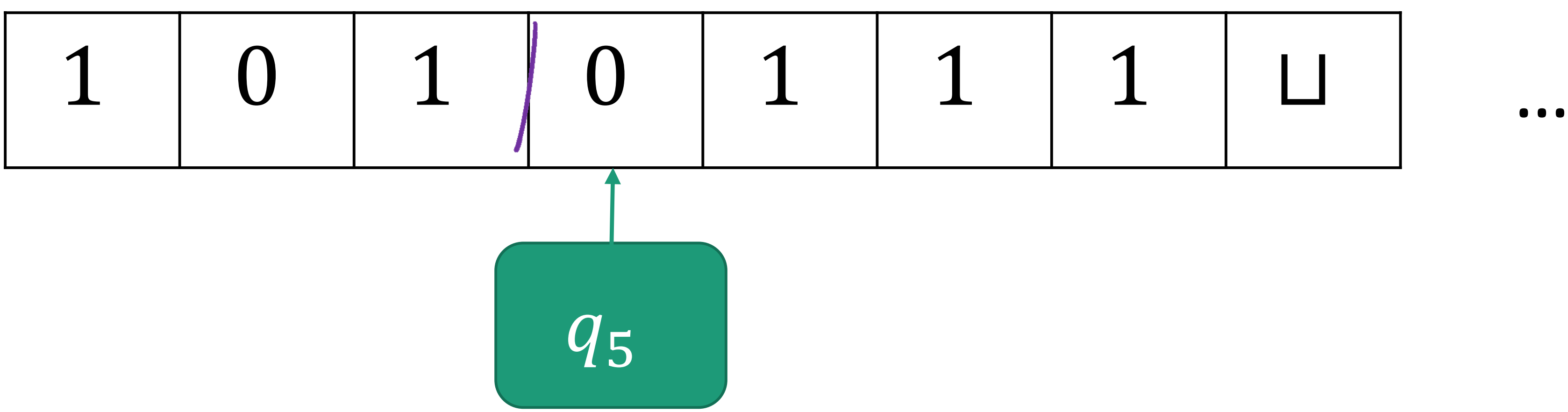


Configuration of a TM: Formally

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

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

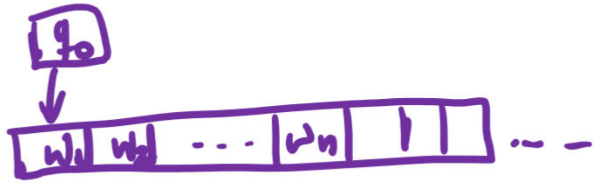
Example: $101q_5 0111$



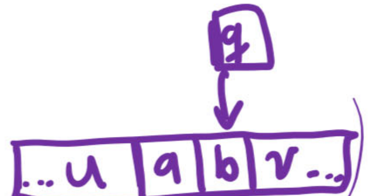
How a TM Computes

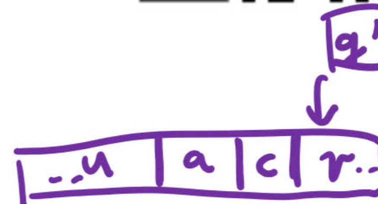
On input $w \in \Sigma^*$

Start configuration: $q_0 w$



In one step of computation:





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

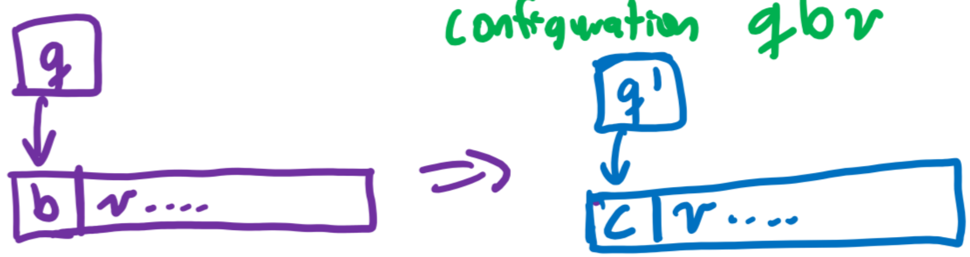
a) $cq'v$

b) $q'cv$

c) $q' \sqcup cv$

d) $q'cbv$

configuration qbv



How a TM Computes

Start configuration: $q_0 w$

In one step of computation:

- If $\delta(q, b) = (q', c, R)$, then $ua\ q\ bv$ yields $uac\ q'\ v$
- If $\delta(q, b) = (q', c, L)$, then $ua\ q\ bv$ yields $u\ q'\ acv$
- If $\delta(q, b) = (q', c, L)$, then $q\ bv$ yields $q'\ cv$

Any config. for which

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

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

How a TM Computes

M accepts input w if there exists a sequence of configurations $C_1, \ldots, C_k$ such that:

- $C_1 = q_0 w$ start configuration
- C_i yields C_{i+1} for every i TM transitions from C_i to C_{i+1} in one step of computation
- 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 on w

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 on w

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}

Recognizers vs. Deciders

If A is decidable, then $A = L(M)$ for some TM M that always halts

$\Rightarrow A = L(M)$ for some TM M

$\Rightarrow A$ is recognizable

Which of the following is true about the relationship between decidable and recognizable languages?

$\{\text{Decidable languages}\} \subsetneq \{\text{Turing recognizable languages}\}$

a) The decidable languages are a subset of the recognizable languages

b) The recognizable languages are a subset of the decidable languages

c) They are incomparable: There might be decidable languages which are not recognizable and vice versa

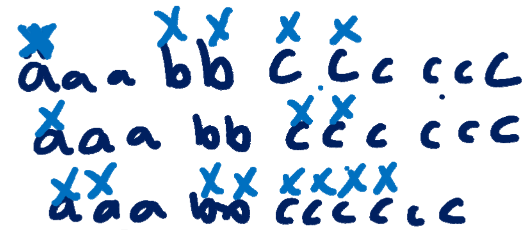
Example: Arithmetic on a TM

The following TM decides $\mathrm{MULT} = \{a^i b^j c^k \mid i \times j = k\}$:

On input string w:

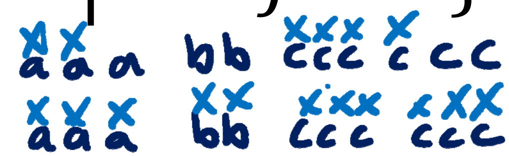
1. Check w is formatted correctly ie. check $w = a^i b^j c^k$ for some i, j, k
2. For each a appearing in w:
3. For each b appearing in w:
4. Attempt to cross off a c. If none exist, reject.
5. If all c's are crossed off, accept. Else, reject.

Example: Arithmetic on a TM



Implementation-level Description

The following TM decides $\text{MULT} = \{a^i b^j c^k \mid i \times j = k\}$:



On input string w:

1. Scan the input from left to right to determine whether it is a member of $L(a^* b^* c^*)$

   ↳ accept

2. Return head to left end of tape

   Else, go to step 5.

3. Cross off an a if one exists. Scan right until a b occurs. Shuttle between b's and c's crossing off one of each until all b's are gone. Reject if all c's are gone but some b's remain.

   crossed off

4. Restore crossed off b's. If any a's remain, repeat step 3.
5. If all c's are crossed off, accept. Else, reject.

Back to Hilbert's Tenth Problem

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

$L = \{ p(x_1, \ldots, x_n) \mid p$ is an integer polynomial, $\exists\, z_1, \ldots, z_n \in \mathbb{Z}$ s.t. $p(z_1, \ldots, z_n) = 0 \}$

- L is Turing-recognizable

Special case where $n = 2$, i.e. $p(x, y)$ e.g. $p(x,y) = x^2 y - 2xy + x^3$

<u>Idea</u>: Try evaluating p on all possible pairs $(x, y) \in \mathbb{Z}^2$

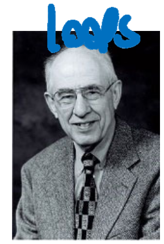
$y \backslash x$	0	-1	+1	-2	+2	....
0	(1)	(2)	(4)			
-1	(3)	(5)				
+1	(6)					
-2						
+2						
⋮						

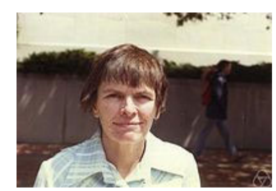
1) Evaluate $p(0,0)$. If $= 0$, accept, else
2) Evaluate $p(-1,0)$. If $= 0$ accept, else
3) Eval. $p(0,-1)$. ...

If $p(x,y)$ has a solution, TM <u>accepts</u>

Else, TM loops forever

- 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…

- Adding nondeterminism does not change the languages recognized by finite automata
- We can require that NFAs have a single accept state

Other modifications possible too: E.g., allowing DFAs to have multiple passes over their input does not increase their power

Turing machines have an astonishing level of robustness

TMs are equivalent to…

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

…

Equivalent TM models

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

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

TMs with stay put are *at least* as powerful as basic TMs

(Every basic TM is a TM with stay put that never stays put)

How would you show that TMs with stay put are *no more* powerful than basic TMs? Everything a TM w/ stay put can do, a basic TM can do

a) Convert any basic TM into an equivalent TM with stay put

b) Convert any TM with stay put into an equivalent basic TM

c) Construct a language that is recognizable by a TM with stay put, but not by any basic TM

d) Construct a language that is recognizable by a basic TM, but not by any TM with stay put