BU CS 332 – Theory of Computation

Lecture 10:

- Turing Machines
- TM Variants and Closure Properties

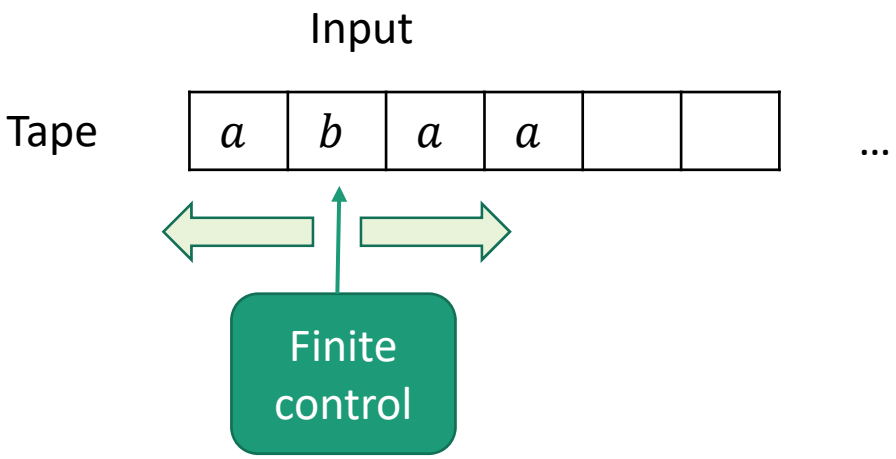
Reading:

Sipser Ch 3.1-3.3

Mark Bun

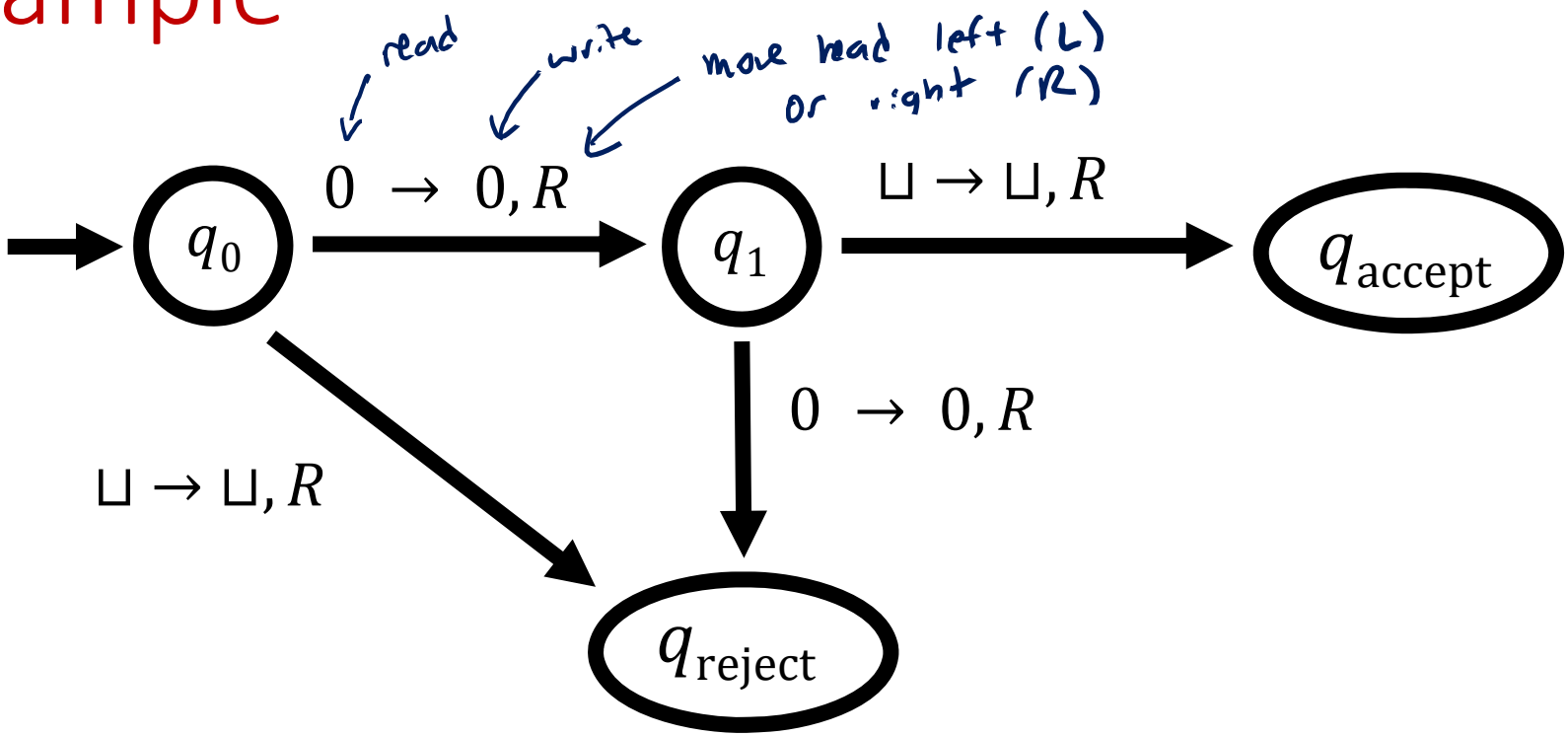
February 24, 2021

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


read
write
move head left (L)
or right (R)
$0 \to 0, R$
$\sqcup \to \sqcup, R$
q_0
q_1
q_{accept}
$0 \to 0, R$
$\sqcup \to \sqcup, R$
q_{reject}

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)
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ is the transition function

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

current state, current symbol (read), next state, next symbol (write), movement (left or right)

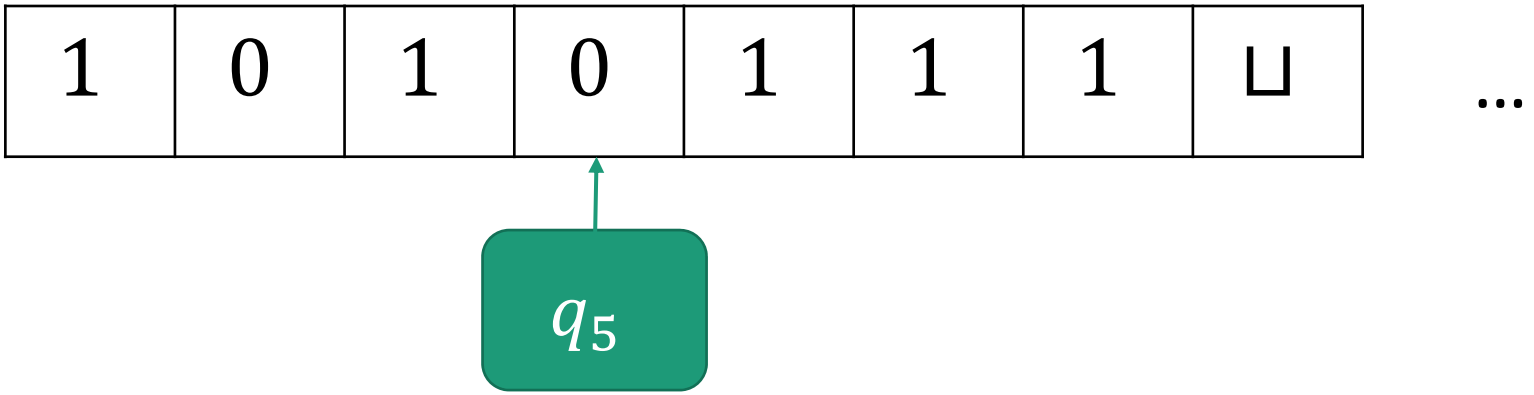
- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state ($q_{\text{reject}} \neq q_{\text{accept}}$)

Configuration of a TM: Formally

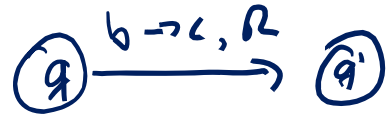
A configuration is a string uqv where $q \in Q$ and $u, v \in \Gamma^*$

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

Example: $101q_5 0111$

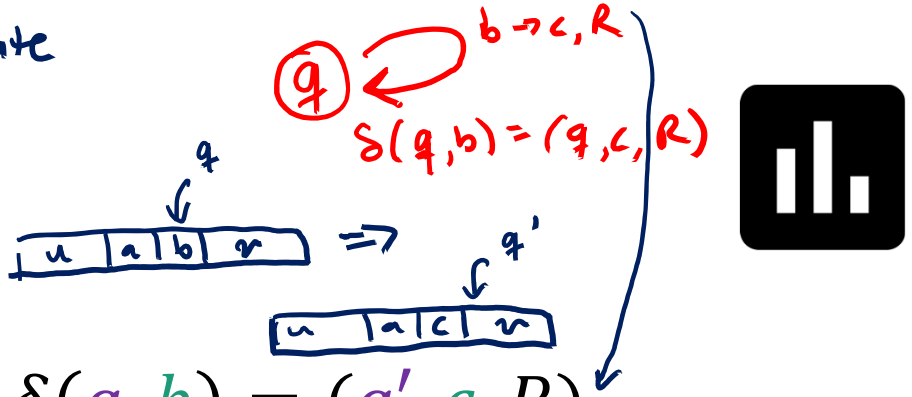


How a TM Computes



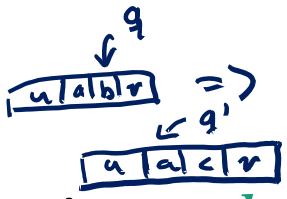
On input w

Start configuration: $q_0 w$ (q_0: start state)

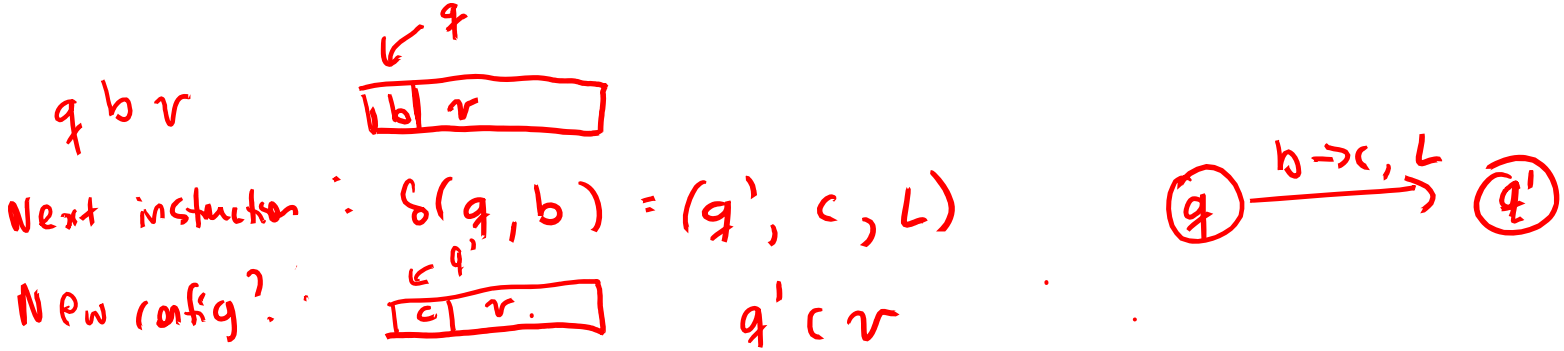


One step of computation:

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



- If we are at the left end of the tape in configuration qbv, 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:

- $uaqbv$ 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 $C_1, \ldots, C_k$ such that: C_i as snapshot of TM @ time i

- $C_1 = q_0 w$ TM starts in start config
- C_i yields C_{i+1} for every i Transition takes TM from C_i to C_{i+1}
- C_k is an accepting configuration TM halts and accepts

$L(M)$ = 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} (no infinite looping)

Recognizers vs. Deciders

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

Decidable langs. $\subseteq$ Recognizable langs.

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

On input string w:

1. Check w is formatted correctly
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

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 $L(a^*b^*c^*)$
2. Return head to left end of tape
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.
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

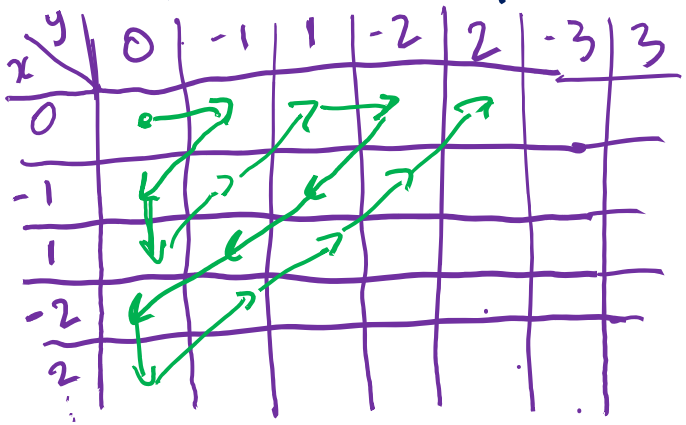
Computational Problem: Given a Diophantine equation, does it have a solution over the integers?

$L = \{ p(z_1, \ldots, z_m) \mid p$ is an integer polynomial $\exists z_1, \ldots, z_m \in \mathbb{Z} \quad p(z_1, \ldots, z_m) = 0 \}$

- L is Turing-recognizable

Simple case: $L_2 = \{ p(x,y) \mid \exists x,y \quad p(x,y) = 0 \}$

Try out (in a slightly clever way) all poss. $(x,y) \in \mathbb{Z}^2$ and see if $p(x,y) = 0$ "dovetailing"



Try $p(0,0)$. If $=0$, accept
Try $p(0,-1)$. If $=0$, accept
" $p(-1,0)$. "
$p(1,0)$.

$p \in L_2 \Rightarrow$ Machine accepts
$p \notin L_2 \Rightarrow$ Machine 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…

- We can require that NFAs have a single accept state
- Adding nondeterminism does not change the languages recognized by finite automata Subset construction
- Bonus problem on test: 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

…

analog of recog. lang. from λ-calculus

Extensions that do not increase the power of the TM model

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

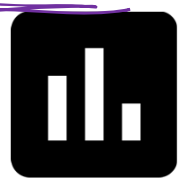
$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$$

"stay put"

How would you show that TMs with stay put are no more powerful than ordinary TMs?

"No more powerful" : Ordinary TMs are at least as powerful

⇐ Anything a TM w/ stay put can do, an ordinary TM can do



Extensions that do not increase the power of the TM model

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

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

TMs w/ stay put are equivalent in power: Show this. and show that ordinary TMs can be converted into TMs w/ stay put

Proof that TMs with "stay put" are no more powerful:

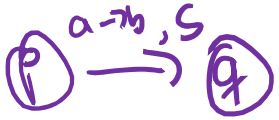
Simulation: Convert any TM M with "stay put" into an equivalent TM M' without

Implementation level

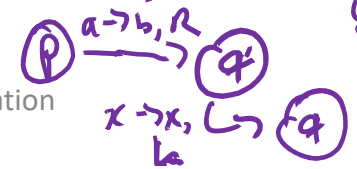
Replace every "stay put" instruction in M with a move right instruction, followed by a move left instruction in M'

If $\delta(p, a) = (q, b, S)$ in M

(a) level



← replace this w/ →

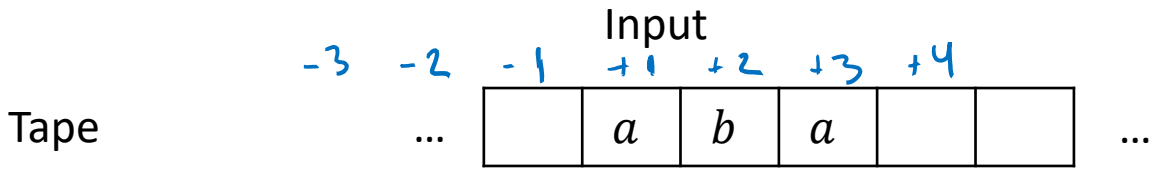


$\delta(p, a) = (q', b, R)$

$\delta(q', x) = (q, x, L)$ (for all x 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"

Formalizing the Simulation

$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

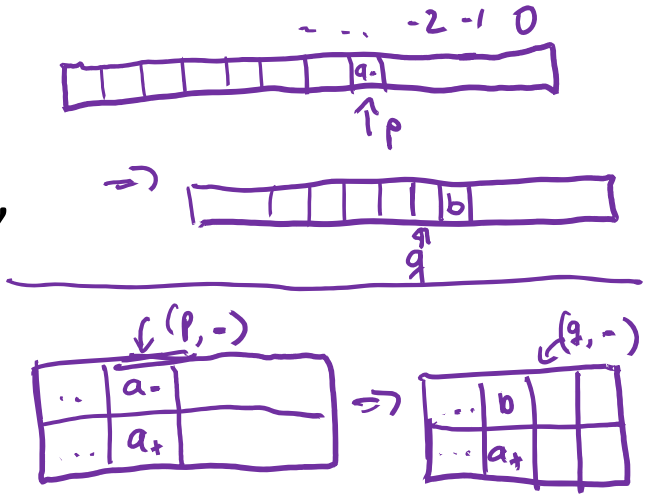
$M' = (Q', \Sigma, \Gamma', \delta', q_0', q_{accept}', q_{reject}')$

New tape alphabet: $\Gamma' = (\Gamma \times \Gamma) \cup \{\$\}$

New state set: $Q' = Q \times \{+, -\}$

$(q, -)$ means "q, working on upper track"

$(q, +)$ means "q, working on lower track"



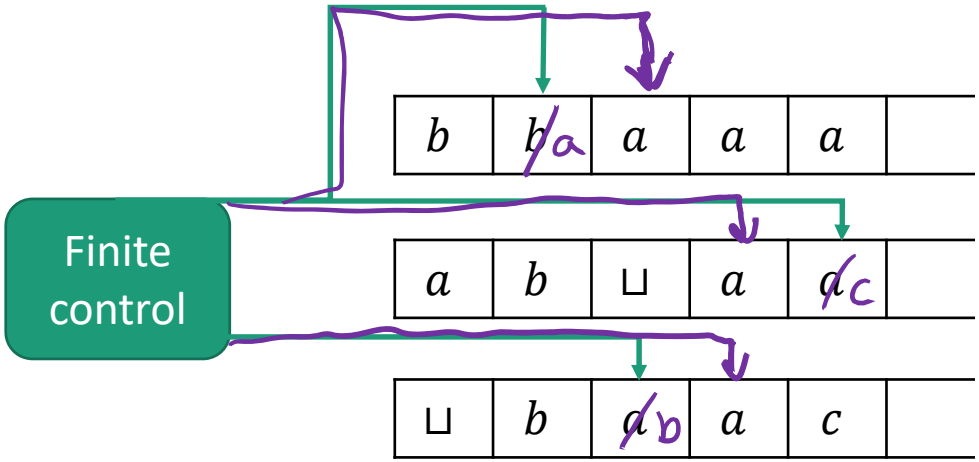
New transitions:

If $\delta(p, a_-) = (q, b, L)$, let $\delta'\big((p, -), (a_-, a_+)\big) = ((q, -), (b, a_+), R)$

Also need new transitions for moving right, lower track, hitting \$,

initializing input into 2-track format

Multi-Tape TMs



Convention

Input (read only)

work tapes

Fixed number of tapes k (can't change during computation)

Transition function $\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k$

$\delta(p, a_1, \ldots, a_k) = (q, b_1, \ldots, b_k, M_1, \ldots, M_k)$

read tape 1 … read tape k

write tape 1 … move tape 1 … move tape k

Multi-Tape TMs are Equivalent to Single-Tape TMs

Theorem: Every k-tape TM M with can be simulated by an equivalent single-tape TM M'