BU CS 332 – Theory of Computation

https://forms.gle/f5ApkNPUu5ucuDa49

Lecture 11:

- TM Variants
- Nondeterministic TMs
- Closure Properties

Reading:

Sipser Ch 3.2

Mark Bun

March 3, 2025

Last Time

Formal definition of a TM, configurations, how a TM computes

Recognizability vs. Decidability:

$A = L(M)$, meaning A is the language recognized by M

A is Turing-recognizable if there exists a TM M such that

- $w \in A \implies M$ 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 there exists a TM M such that

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

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 already a TM with stay put that never stays put)

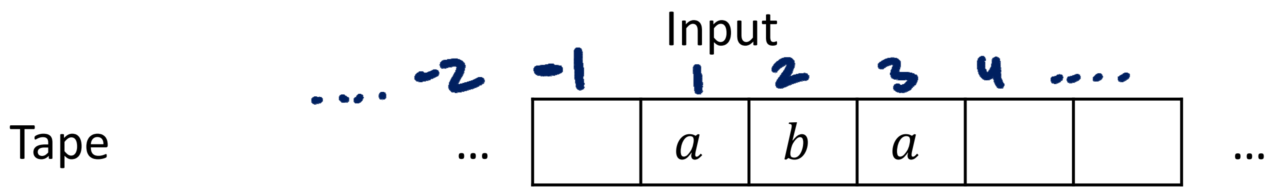
Proof that TMs with stay put are *no more* powerful:

Simulation: Our goal is to convert any TM M with stay put into an equivalent basic TM M'

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

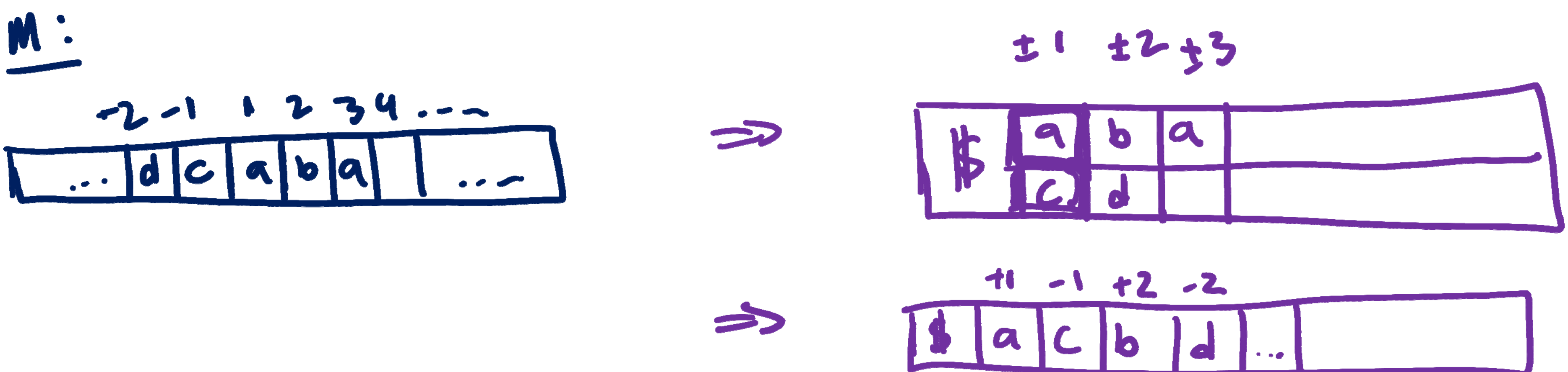
Equivalent TM models

- 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”



Implementation-Level Simulation

Given 2-way TM M construct a basic TM M' as follows.

TM M' = "On input $w = w_1 w_2 \ldots w_n$:

1. Format 2-track tape with contents

$$\$, (w_1, \sqcup), (w_2, \sqcup), \ldots, (w_n, \sqcup)$$

2. To simulate one move of M:

a) If working on upper track, read/write to the first position of cell under tape head, and move in the same direction as M

b) If working on lower track, read/write to second position of cell under tape head, and move in the opposite direction as M

c) If move results in hitting \$, switch to the other track. "

Formalizing the Simulation

Given 2-way TM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, construct $M' = (Q', \Sigma, \Gamma', \delta', q_0', q_{\text{accept}}', q_{\text{reject}}')$

upper symbol

lower symbol

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

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

$(q, +)$ means "in state q and working on upper track"

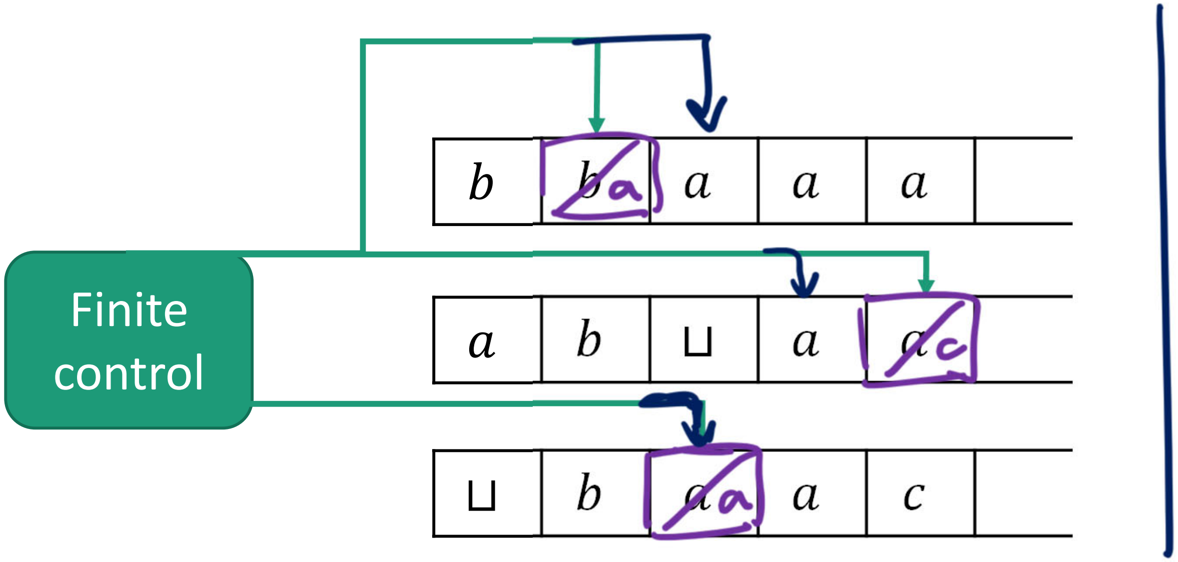
$(q, -)$ means "in state q and working on lower track"

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



Fixed number of tapes k

(<u>k can't depend on input or change during computation)</u>

new state

k new symbols

$$\text{Transition function } \delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \underbrace{\{L, R, S\}^k}_{k \text{ movement instructions}}$$

current state

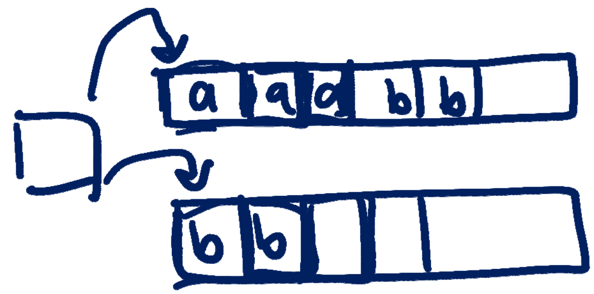
k alphabet symbols, one per tape

Why are Multi-Tape TMs Helpful?

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

$\Rightarrow$ To show a language is Turing-recognizable or decidable, it's enough to construct a multi-tape TM

Often easier to construct multi-tape TMs



Ex. Decider for $\{a^i b^j \mid i > j\}$

On input w:

1) Scan tape 1 left-to-right to check that $w \in L(a^* b^*)$
2) Scan tape 2 left-to-right to copy all b's to tape 2
3) Starting from left ends of tapes 1 and 2, scan both tapes to check that every b on tape 2 has an accompanying a on tape 1. If not, reject.
4) Check that the first blank on tape 2 has an accompanying a on tape 1. If so, accept; otherwise, reject.

Why are Multi-Tape TMs Helpful?

To show a language is Turing-recognizable or decidable, it's enough to construct a multi-tape TM

$\forall$ recognizable A, B
$A \cup B$ is also recognizable

Very helpful for proving closure properties

Ex. Closure of recognizable languages under union. Suppose M_1 is a single-tape TM recognizing L_1, M_2 is a single-tape TM recognizing L_2

Construct a multi-tape TM recognizing $L_1 \cup L_2$

N:

" On input $w = w_1 w_2 \ldots w_n$:

1. Copy w to tape 2
2. Run M_1 on tape 1. If M_1 accepts, accept.
3. Run M_2 on tape 2. If M_2 accepts, accept.
4. Else reject.

Attempted proof of correctness:

- If $w \in L_1 \cup L_2$, WTS that N accepts w

If $w \in L_1$, step 2 leads N to accept ✓

If $w \in L_2$ we would like N to accept in step 3, but might not happen if M_1 loops on w!

Why are Multi-Tape TMs Helpful?

To show a language is Turing-recognizable or decidable, it's enough to construct a multi-tape TM

Very helpful for proving closure properties

Ex. Closure of recognizable languages under union. Suppose M_1 is a single-tape TM recognizing L_1, M_2 is a single-tape TM recognizing L_2

On input w: TM N

1) Scan tapes 1, 2, and 3 left-to-right to copy w to tapes 2 and 3
2) Repeat forever:
   a) Run M_1 for one step on tape 2
   b) Run M_2 for one step on tape 3
   c) If either machine accepts, accept

3) If both rejected, reject.

- If $w \notin L_1 \cup L_2$, then neither M_1 nor M_2 will accept, so N does not accept

Proof.

- If $w \in L_1 \cup L_2$, then either $w \in L_1$ or $w \in L_2$.
  - If $w \in L_1$, then M_1 accepts w in some finite number k of steps
  $\Rightarrow$ N accepts by k'th time through loop
  - Similarly for $w \in L_2$

Closure Properties

The Turing-decidable languages are closed under:

- Union
- Concatenation
- Star
- Intersection
- Reverse
- Complement

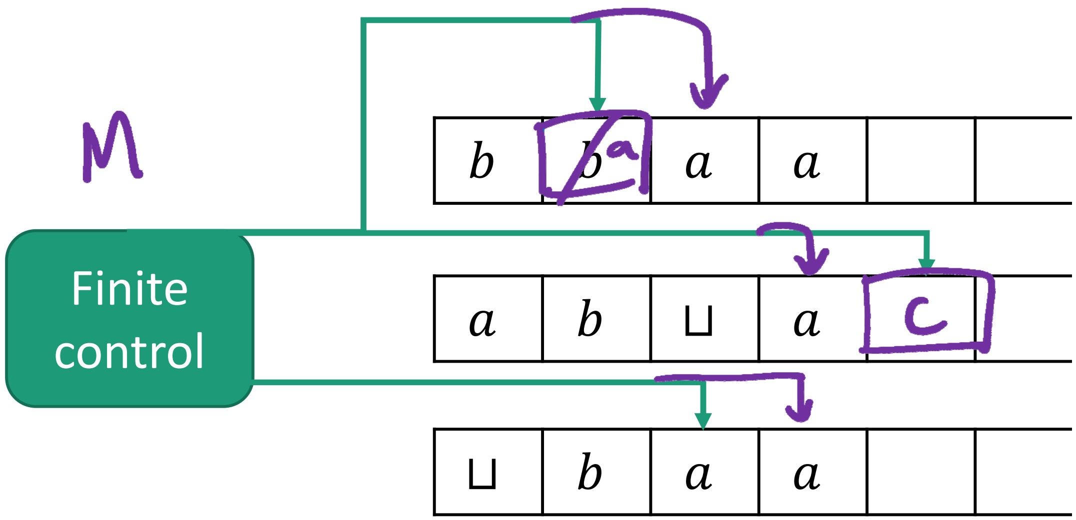
The Turing-recognizable languages are closed under:

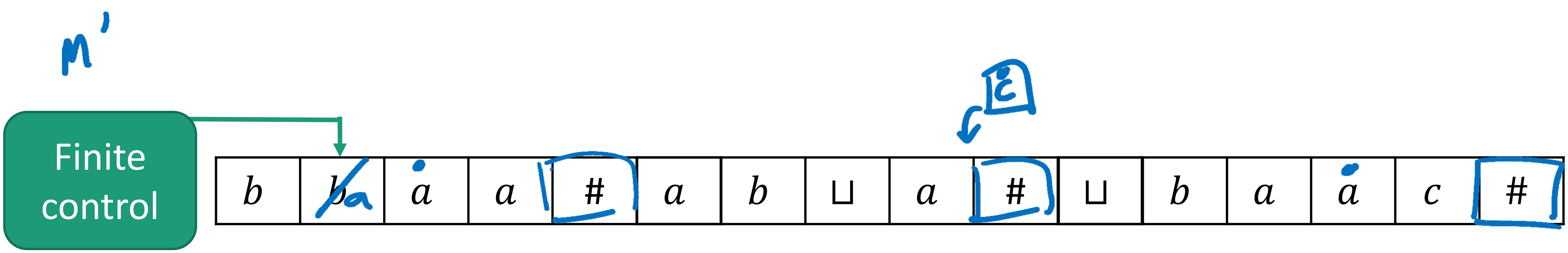
- Union
- Concatenation
- Star
- Intersection
- Reverse

Not closed under complement

Multi-Tape TMs are Equivalent to Single-Tape TMs

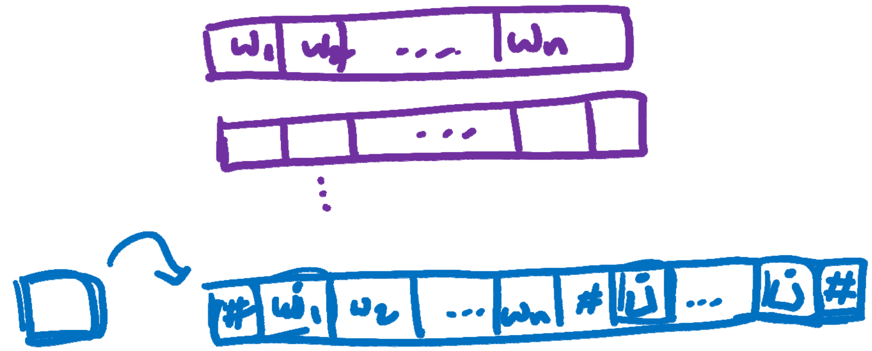
Theorem: Every k-tape TM M with can be simulated by an equivalent single-tape TM M'





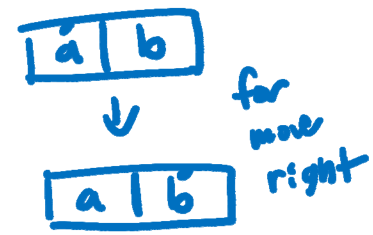
Simulating Multiple Tapes

Implementation-Level Description



On input $w = w_1 w_2 \ldots w_n$

1. Format tape into $\# \dot{w}_1 w_2 \ldots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \ldots \#$
2. For each move of M:
   - Scan left-to-right, finding current symbols
   - Scan left-to-right, writing new symbols,
   - Scan left-to-right, moving each tape head e.g.



If a tape head goes off the right end, insert blank using left-to-right pass

If a tape head goes off left end, move back right

Why are Multi-Tape TMs Helpful?

To show a language is Turing-recognizable or decidable, it's enough to construct a multi-tape TM

Very helpful for proving closure properties

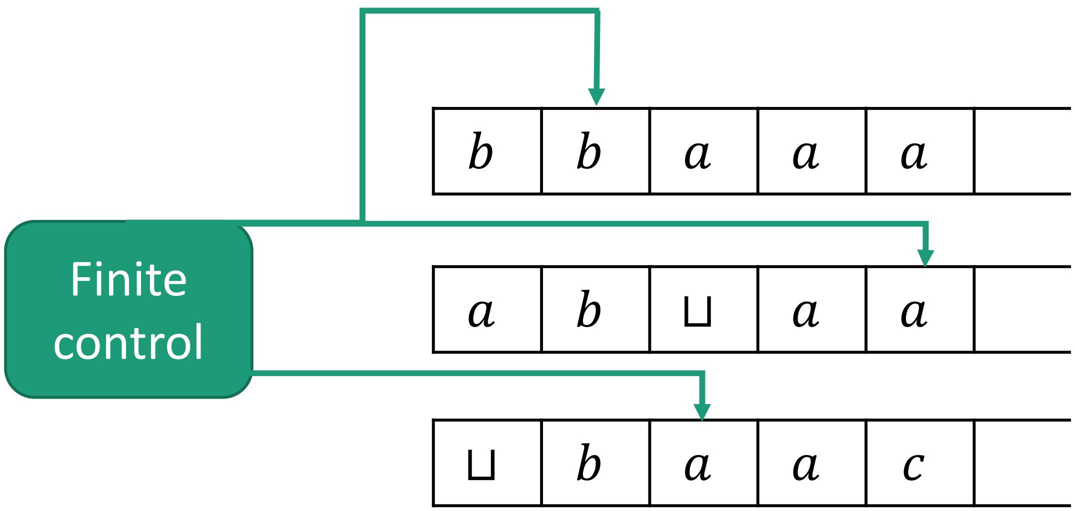
Ex. Closure of recognizable languages under union. Suppose M_1 is a single-tape TM recognizing L_1, M_2 is a single-tape TM recognizing L_2

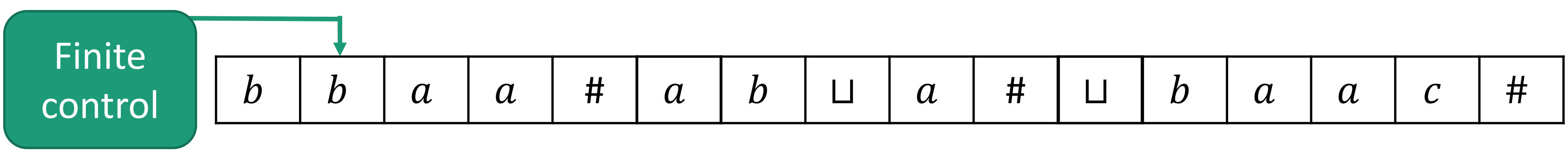
On input w:

1) Scan tapes 1, 2, and 3 left-to-right to copy w to tapes 2 and 3
2) Repeat forever:
    a) Run M_1 for one step on tape 2
    b) Run M_2 for one step on tape 3
    c) If either machine accepts, accept

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'





How to Simulate It

e.g. multi-tape TM

To show that a TM variant is no more powerful than the basic, single-tape TM:

Show that if M is any variant machine, there exists a basic, single-tape TM M' that can simulate M

(Usual) parts of the simulation:

- Describe how to initialize the tapes of M' based on the input to M
- Describe how to simulate one step of M's computation using (possibly many steps of) M'

Simulating Multiple Tapes

Implementation-Level Description of M'

On input $w = w_1 w_2 \dots w_n$

1. Format tape into $\# \dot{w}_1 w_2 \dots w_n \# \dot{\sqcup} \# \dot{\sqcup} \# \dots \#$
2. For each move of M:

    Scan left-to-right, finding current symbols

    Scan left-to-right, writing new symbols,

    Scan left-to-right, moving each tape head

    If a tape head goes off the right end, insert blank

    If a tape head goes off left end, move back right

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

…

Nondeterministic TMs

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting branch.

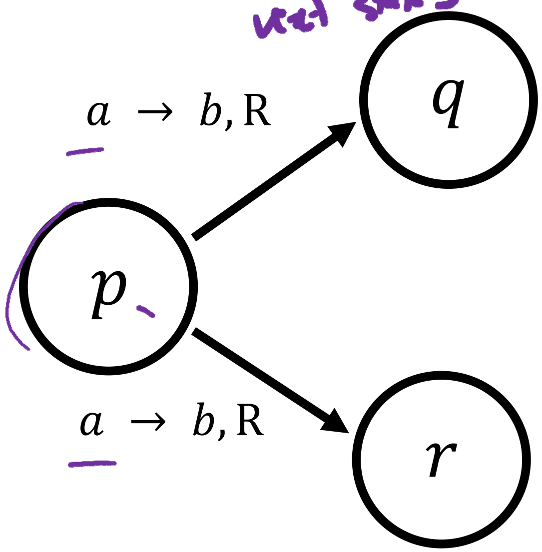
Transition function $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\})$

current state

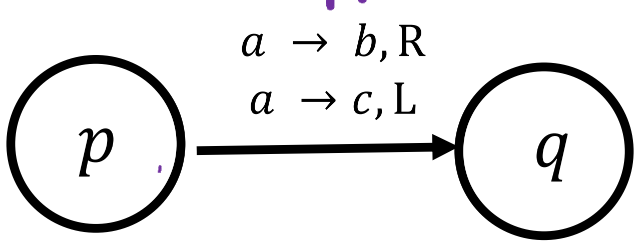
cur. symbol

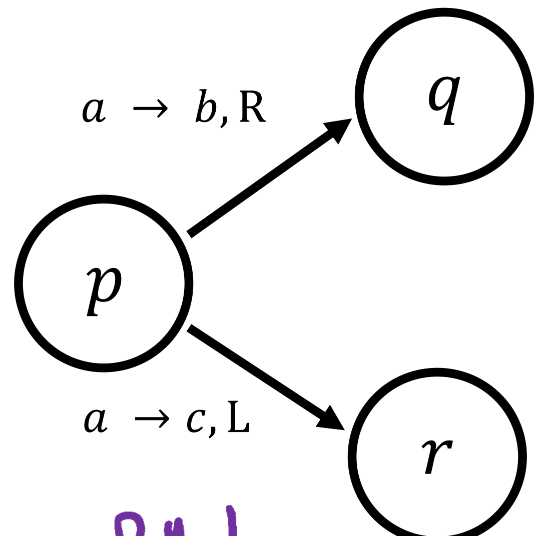
set of possible (next state, new symbol, movement) triples

Multiple possible next states



Multiple possible write/move instructions

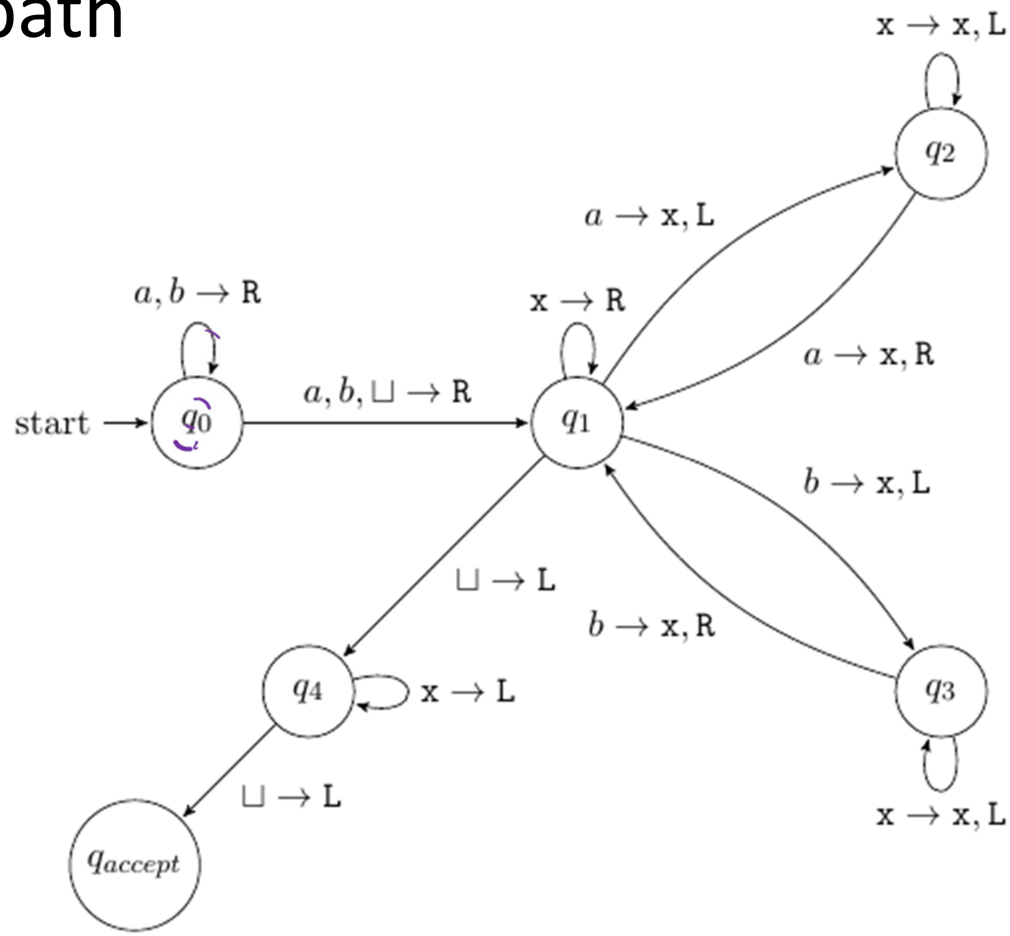




Both!

Nondeterministic TMs

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting computation path

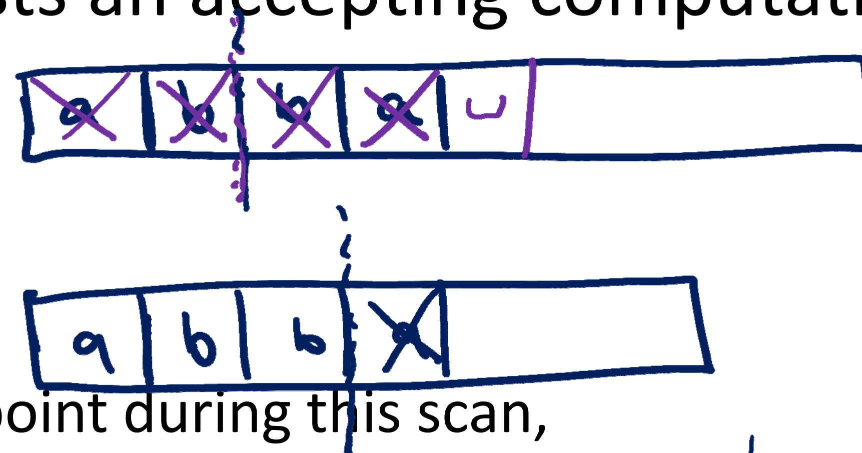


Nondeterministic TMs

abba accepted overall b/c there exists an accepting branch

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting computation path

Branch of computation leading to accept



Implementation-Level Description

Different branch: reject

On input string w:

1) Scan tape left-to-right. At some point during this scan, nondeterministically go to step 2

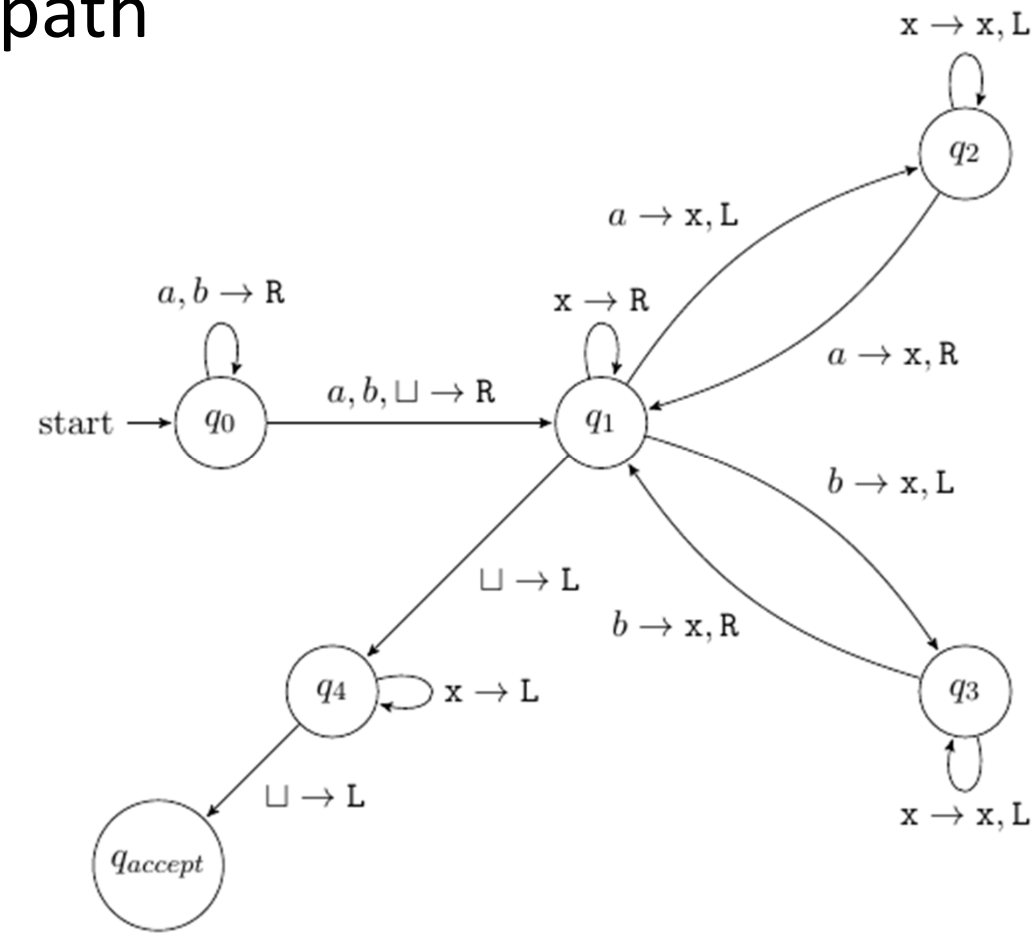
a a a b b x a b b b a

2) a) Read the next symbol s and cross it off
   b) Move the head left repeatedly until a non-x symbol is found. If it matches s, cross it off. Else, reject.
   c) Move the head right until a non-x symbol is found. If blank is hit, go to step 3.
   d) Go back to 2a)

3) Check that the entire tape consists of x's. If so, accept. Else, reject.

Nondeterministic TMs

At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting computation path



What is the language recognized by this NTM?

a) $\{ ww \mid w \in \{a, b\}^* \}$
b) $\{ ww^R \mid w \in \{a, b\}^* \}$
c) $\{ ww \mid w \in \{a, b, \mathrm{x}\}^* \}$
d) $\{ w\mathrm{x}^n w^R \mid w \in \{a, b\}^*, n \geq 0 \}$