BU CS 332 – Theory of Computation

https://forms.gle/h1xzMYnBnYKm3yMJ8

Lecture 11:

- More TM Variants and Closure Properties
- Church-Turing Thesis

Reading:

Sipser Ch 3.2

Mark Bun

October 14, 2021

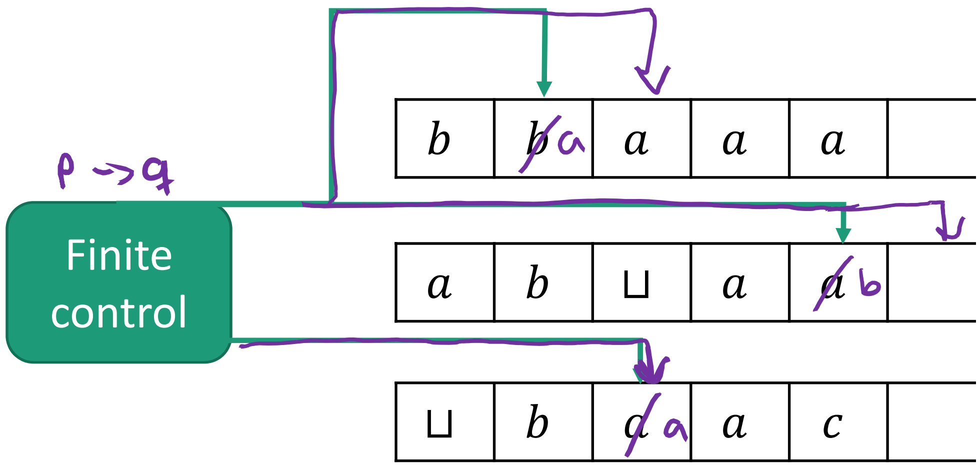
TM Variants

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

…

Multi-Tape TMs



Fixed number of tapes k

(k can't depend on input or change during computation)

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

current state | read k symbols | new state | write k symbols | k movement instructions

p | (b, a, a) | q | (a, b, a) | (R, R, S

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

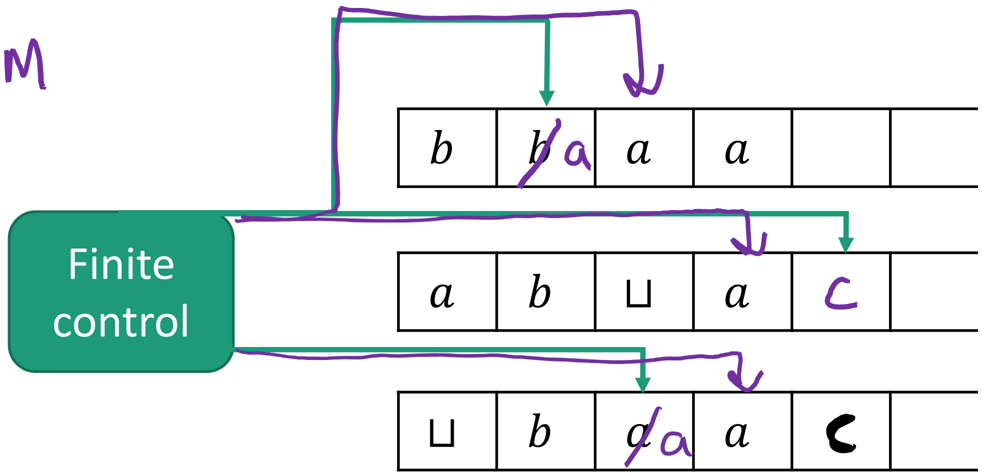
(Usual) parts of the simulation:

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

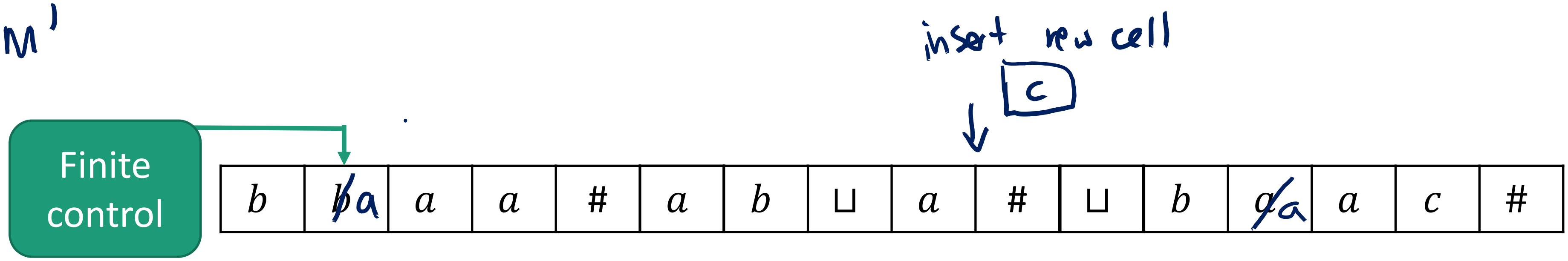
Multi-Tape TMs are Equivalent to Single-Tape TMs

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

M



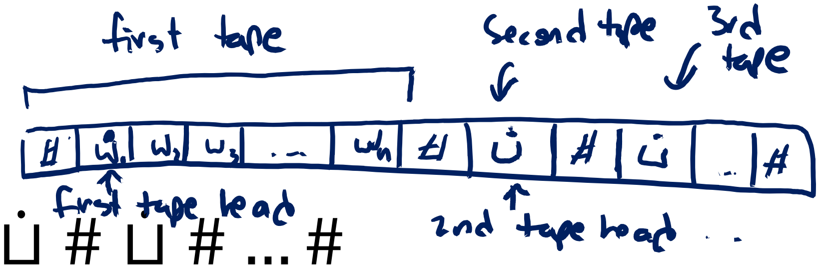
M'



Simulating Multiple Tapes

$\dot{a}$ means tape head of M is over a

Implementation-Level Description of M'



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 — find all symbols w/ dots

    Scan left-to-right, writing new symbols, — update symbols under dots

    Scan left-to-right, moving each tape head — move dots

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

    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

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?

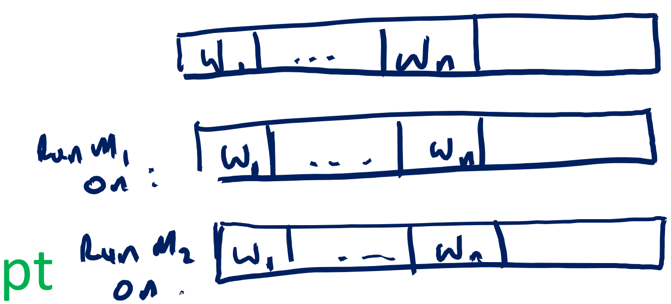
To show a language is Turing-recognizable or decidable, it's enough to construct a multi-tape TM

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: New TM recognizing $L_1 \cup L_2$

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



Interlaced computation deals w/ possibility that M_1 loops on w, while M_2 accepts w.

Closure Properties

L decidable, $\exists$ TM M s.t. $\forall w$:
$w \in L \Rightarrow M$ accepts w
$w \notin L \Rightarrow M$ rejects w

The Turing-decidable languages are closed under:

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

M' interchanges accept/reject states of M

The Turing-recognizable languages are closed under:

L recognizable $\Rightarrow$ $\exists$ TM M s.t. $\forall w$:
$w \in L \Rightarrow M$ accepts w
$w \notin L \Rightarrow M$ either rejects w or loops forever

- Union
- Concatenation
- Star
- Intersection
- Reverse

Not Complement:
Above construction fails because it may loop on strings it should accept

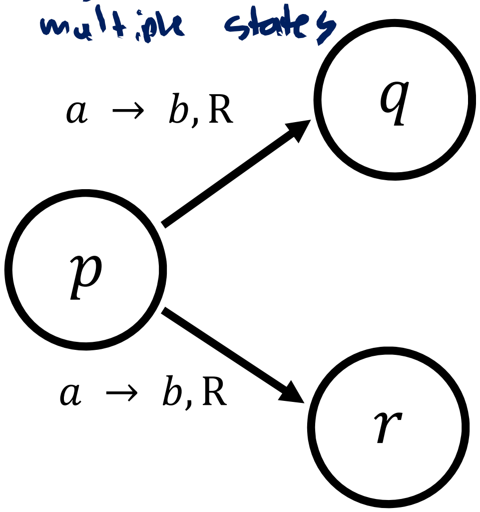
Later: we'll see $\exists$ recognizable L s.t. $\bar{L}$ is not recognizable

Nondeterministic TMs

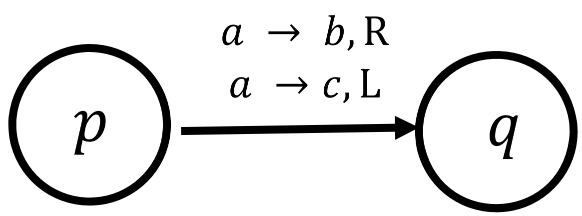
At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting branch.

Transition function $\delta : \underbrace{Q}_{\text{Curr state}} \times \underbrace{\Gamma}_{\text{Curr symbol}} \rightarrow \underbrace{P(Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\})}_{\text{Set of possible next moves}}$

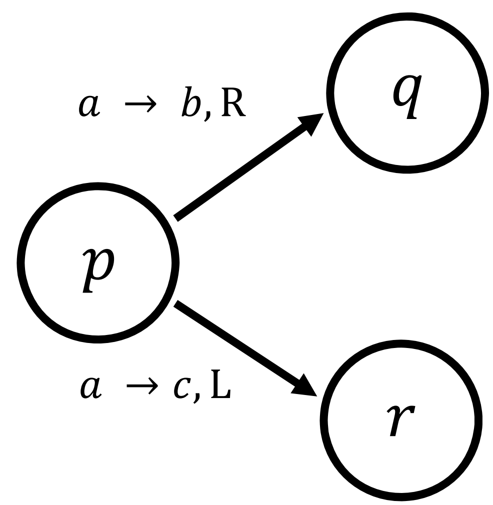
Same read/write move may take you to multiple states



Multiple ways to update tape



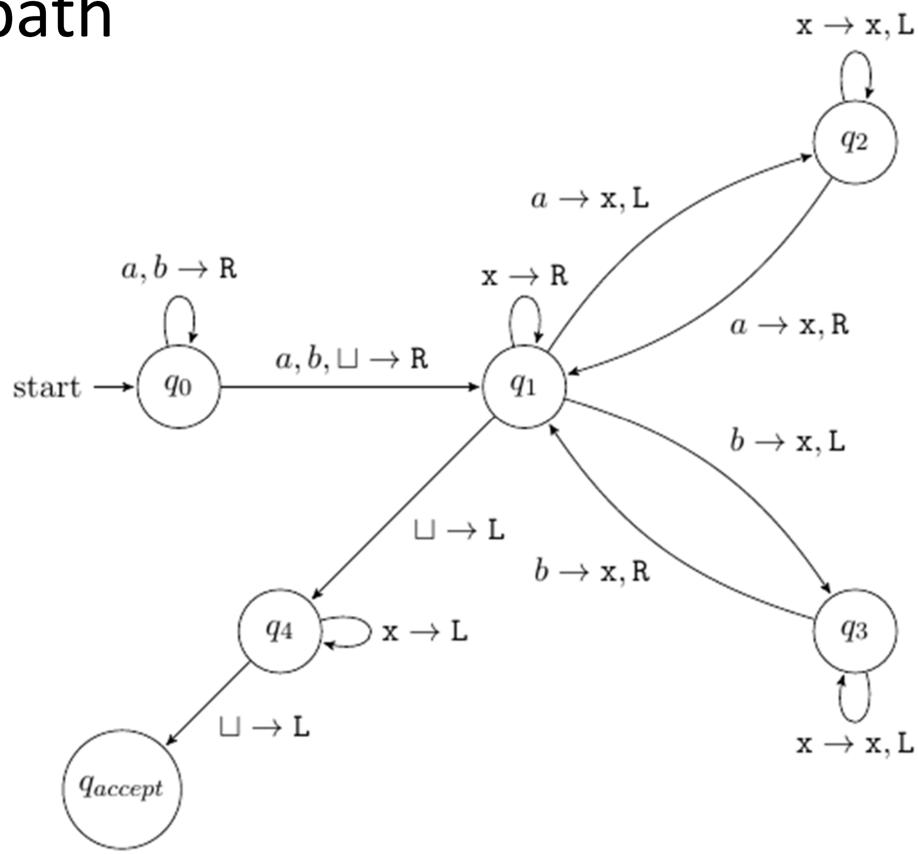
Or both!



Nondeterministic TMs

Computation path 2 means NTM accepts abba

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



Input abba

Computation path 1:

$q_0\, a\, b\, b\, a$

$a\, q_0\, b\, b\, a$

$a\, b\, q_0\, b\, a$

$a\, b\, b\, q_0\, a$

$a\, b\, b\, a\, q_0\, \sqcup$

reject

Computation path 2:

$q_0\, a\, b\, b\, a$

$a\, q_0\, b\, b\, a$

$a\, b\, q_1\, b\, a$

$a\, q_3\, b\, x\, a$

$a\, x\, q_1\, x\, a$

$a\, x\, x\, q_1\, a$

$a\, x\, q_2\, x\, x$

$a\, q_2\, x\, x\, x$

$q_2\, a\, x\, x\, x$

$x\, q_1\, x\, x\, x$

$\vdots$

$x\, x\, x\, x\, q_1\, \sqcup$

$x\, x\, x\, q_4\, x$

$\vdots$

$q_4\, \sqcup\, x\, x\, x\, x$

accept

Nondeterministic TMs

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

~~abba~~ a~~b~~xxx~~b~~

Implementation-Level Description:

On input string w: (over alphabet {a, b})

1) Scan tape left to right, at some point (nondeterministically) goto step 2

2) a) Read next symbol s, cross it off

b) Move head left repeatedly, looking for s. If found, cross it off. (otherwise reject)

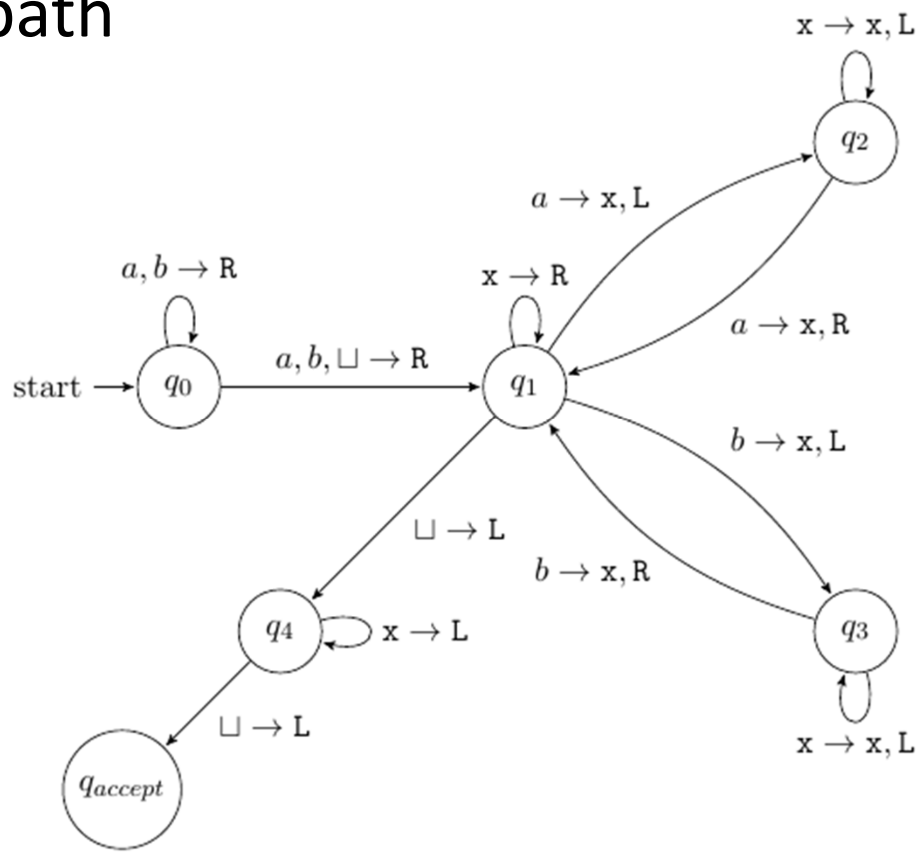
c) Move head right until reaches non-X symbol. If ⊔ hit; go to step 3

d) Repeat (go back to 2 a)

3) Check that entire tape is X's. If so accept.

Nondeterministic TMs

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



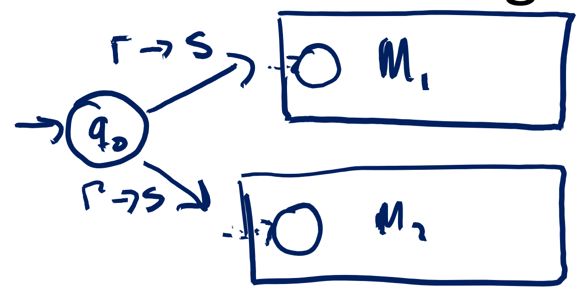
What is the language recognized by this NTM?

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

Nondeterministic TMs

Ex. Given TMs M_1 and M_2, construct an NTM recognizing $L(M_1) \cup L(M_2)$

NTM recognizing $L(M_1) \cup L(M_2)$:



low-level description

Implementation-level:

On input w:

1) Nondeterministically, either:

   a) Run M_1 on tape, accept if accepts, or

   b) Run M_2 on tape, accept if accepts.

Nondeterministic TMs

Ex. NTM for $L = \{w \mid w$ is a binary number representing the product of two integers $a, b \geq 2\}$

High-Level Description:

On input w:

1) Nondeterministically "guess" $a \in \{2, \ldots, w\}$ and $b \in \{2, \ldots, w\}$
2) Multiply $a \cdot b$, check equal to w. Accept if so, reject o.w.

Analysis:

- If $w \in L$, $\exists\ \hat{a}, \hat{b} \in \{2, \ldots, w\}$ s.t. $\hat{a} \cdot \hat{b} = w$. So path of computation that guessed these factors leads to accept.
- If $w \notin L$, every a, b guessed will lead to reject

Nondeterministic TMs

An NTM N accepts input w if when run on w it accepts on at least one computational branch

$$L(N) = \{w \mid N \text{ accepts input } w\}$$

w ∉ L ⟹ every computation path of N on w leads to reject, looping or failure to reach any state leading to accept

NTM N recognizes language L if:

w ∈ L ⟹ ∃ computation path of N on w leading to accept

An NTM N is a decider if on **every** input, it halts on **every** computational branch

N decides L if:

w ∈ L ⟹ ∃ computation path leading to accept

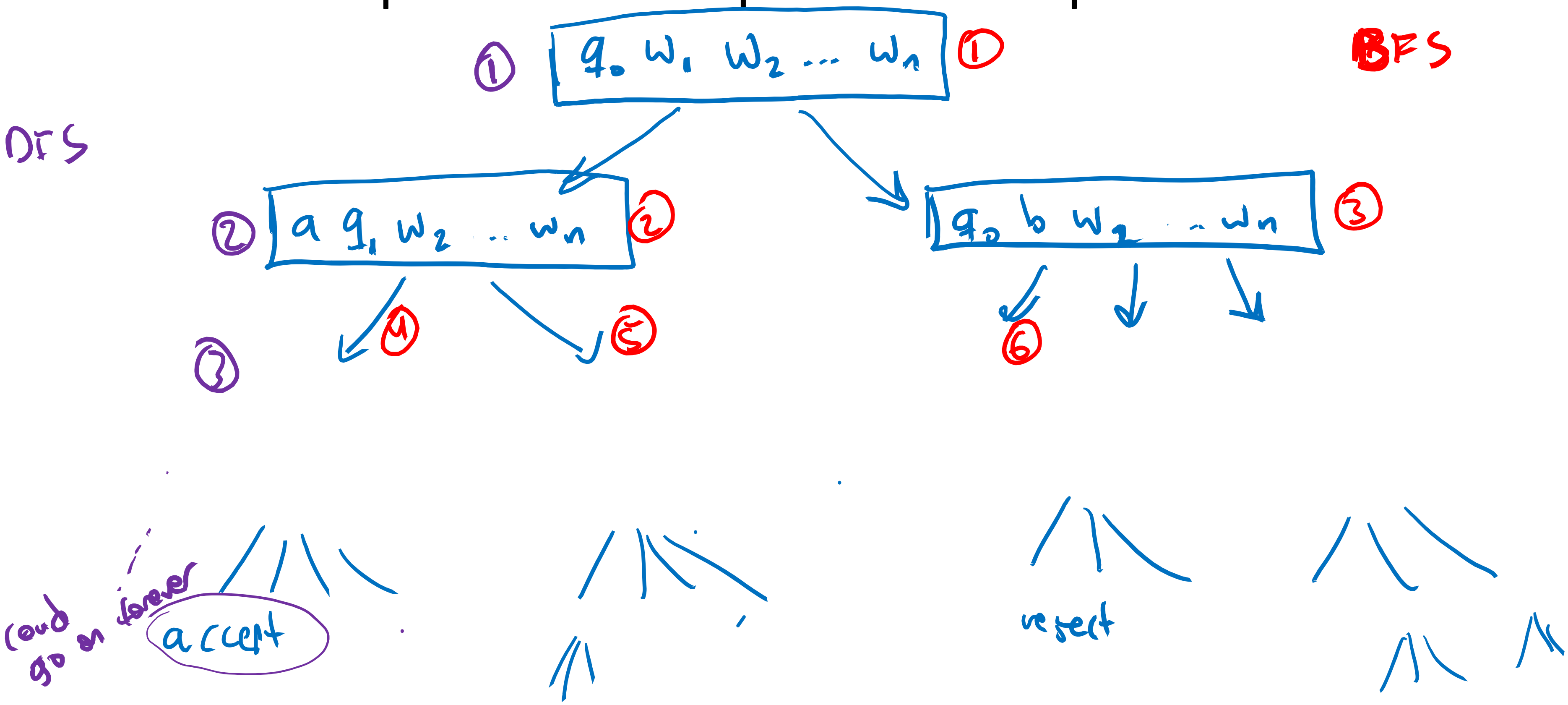
w ∉ L ⟹ every computation path leads to reject (must halt)

Nondeterministic TMs

Theorem: Every nondeterministic TM can be simulated by an equivalent deterministic TM

of N on input w

Proof idea: Explore “tree of possible computations”



Simulating NTMs

Which of the following algorithms is always appropriate for searching the tree of possible computations for an accepting configuration?

a) Depth-first search: Explore as far as possible down each branch before backtracking

Works if NTM N was a decider

b) Breadth-first search: Explore all configurations at depth 1, then all configurations at depth 2, etc.

Always works $\therefore$ Accepts if $w \in L$

Rejects or loops if $w \notin L$

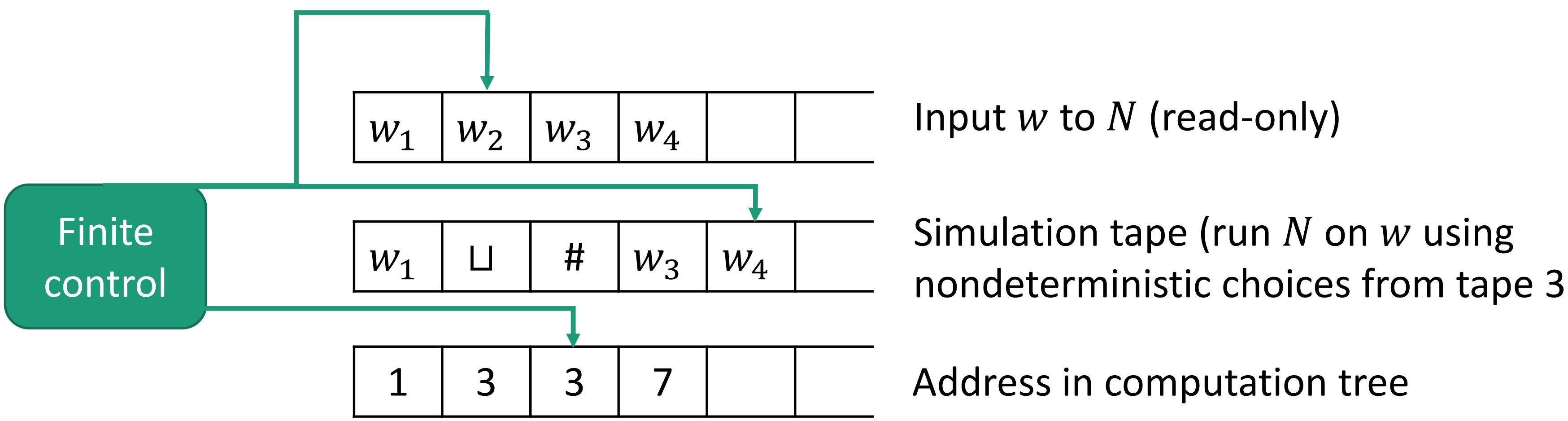
c) Both algorithms will always work

Nondeterministic TMs

Theorem: Every nondeterministic TM has an equivalent deterministic TM

Proof idea: Simulate an NTM N using a 3-tape TM

(See Sipser for full description)



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

…

Church-Turing Thesis

The equivalence of these models is a mathematical theorem (you can prove that each can simulate another)

Church-Turing Thesis v1: The basic TM (hence all of these models) captures our intuitive notion of algorithms

Normative, prescriptive

Church-Turing Thesis v2: Any physically realizable model of computation can be simulated by the basic TM

Empirical

The Church-Turing Thesis is not a mathematical statement! Can't be mathematically proved

"meta-mathematical"