# BU CS 332 – Theory of Computation

https://forms.gle/5oHvd11677Wi4W6c9

## Lecture 12:

- Church-Turing Thesis
- Decidable Languages

Reading:

Sipser Ch 3.3, 4.1

HW 6 is up, due Tuesday March 18?

Mark Bun

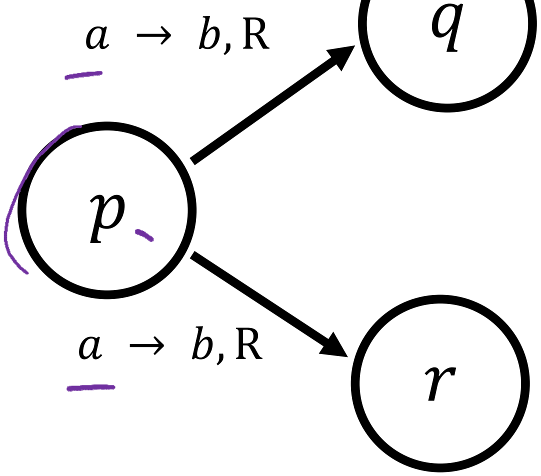
March 5, 2025

# Last Time: Nondeterministic TMs

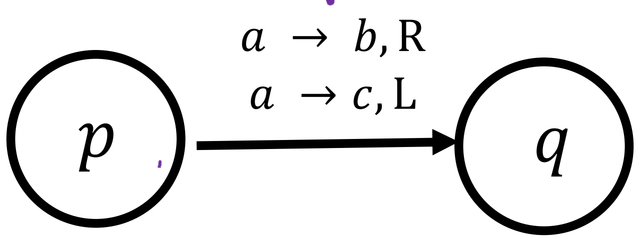
At any point in computation, may nondeterministically branch. Accepts iff there exists an accepting branch.

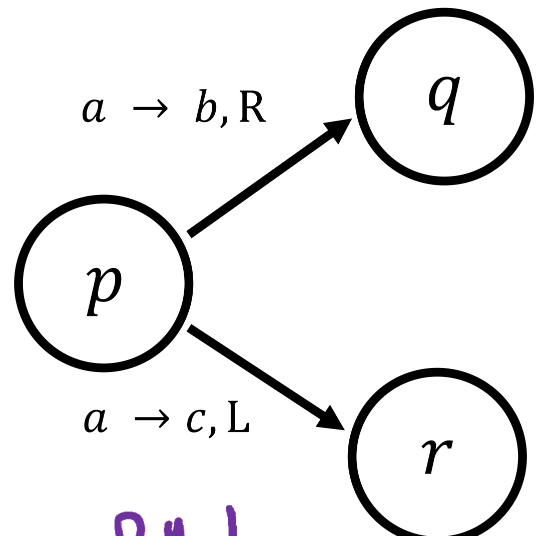
Transition function $\delta : \underset{\text{current state}}{Q} \times \underset{\text{curr. symbol}}{\Gamma} \rightarrow P(\underbrace{Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}, \mathrm{S}\}}_{\text{set of possible (next state, new symbol, movement) triples}})$

Multiple possible next states




Multiple possible write/move instructions




Both!

# Nondeterministic TMs

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

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

$w \in L(N) \Rightarrow$ there exists a branch of $N$'s computation leading it to accept input $w$

$w \notin L(N) \Rightarrow$ all branches of $N$'s computation lead it to reject, run forever, or fail to reach any state on input $w$

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

$w \in L(N) \Rightarrow$ there exists a branch of $N$'s computation leading it to accept input $w$

$w \notin L(N) \Rightarrow$ all branches of $N$'s computation lead it to reject input $w$

# Nondeterministic TMs

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

NTM N:

On input w:

1. Nondeterministically either:

   a) Run $M_1$ on input w, or

   b) Run $M_2$ on input w

2. If accepts, accept, if rejects, reject.

Analysis:

- If $w \in L(M_1) \cup L(M_2)$, either $M_1$ accepts w or $M_2$ accepts w.

  ⇒ Branch of computation in which correct machine was guessed leads to acceptance

- If $w \notin L(M_1) \cup L(M_2)$, neither branch of computation can lead to acceptance

# Nondeterministic TMs

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

High-Level Description:

NTM N:

On input $w$ (interpreted as a natural number):

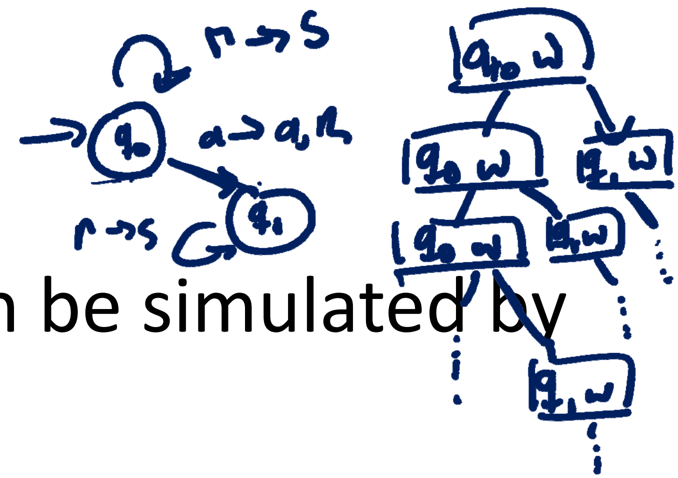
1) Nondeterministically guess factors $a, b \in \{2, 3, \ldots, \sqrt{w}\}$
2) If $a \times b = w$: accept

   Else, reject.

Analysis:

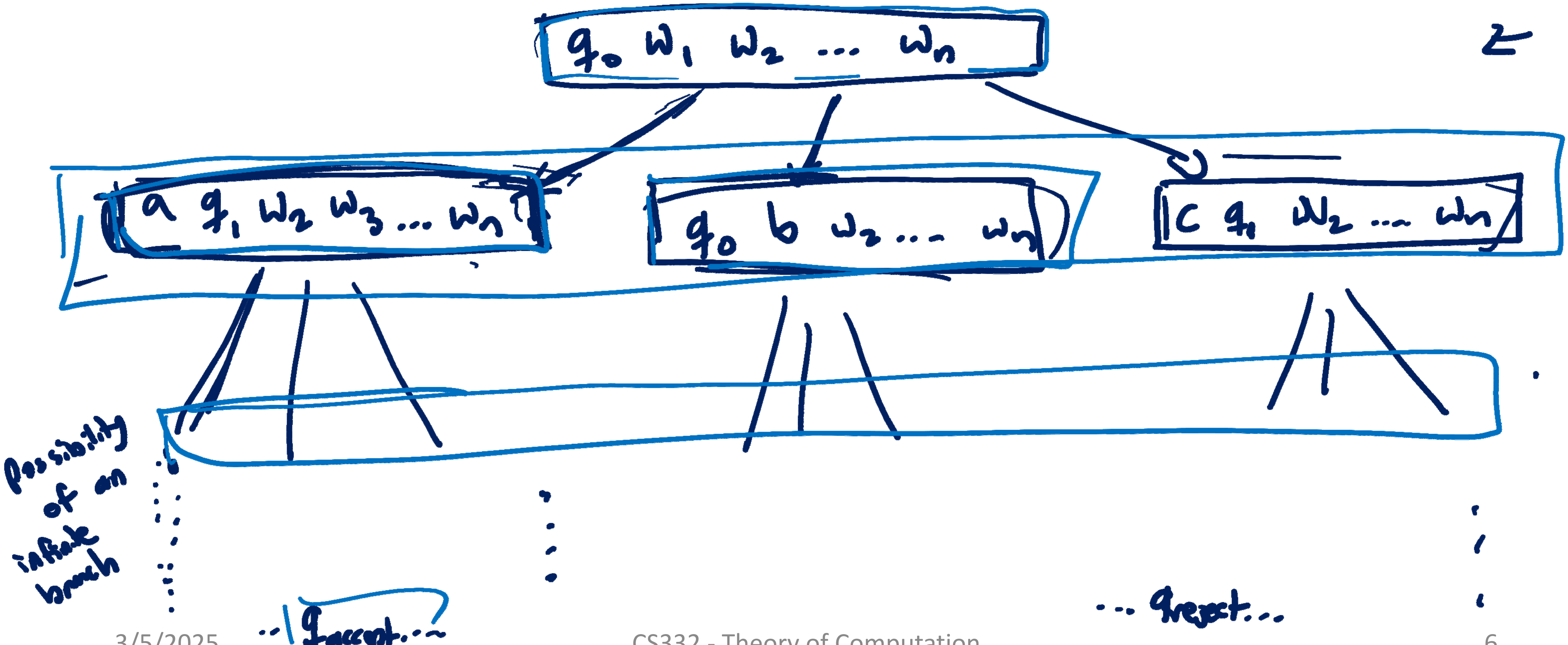
- If $w \in L$: $\exists\ a, b \in \{2, 3, \ldots, \sqrt{w}\}$ s.t. $a \times b = w$. $\Rightarrow$ Branch of computation in which these $a, b$ are guessed leads to acceptance
- If $w \notin L$: $\forall\ a, b$

# Nondeterministic TMs




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

**Proof idea:** Explore "tree of possible computations"
arising from running NTM N on input w.



# 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

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

c) Both algorithms will always work

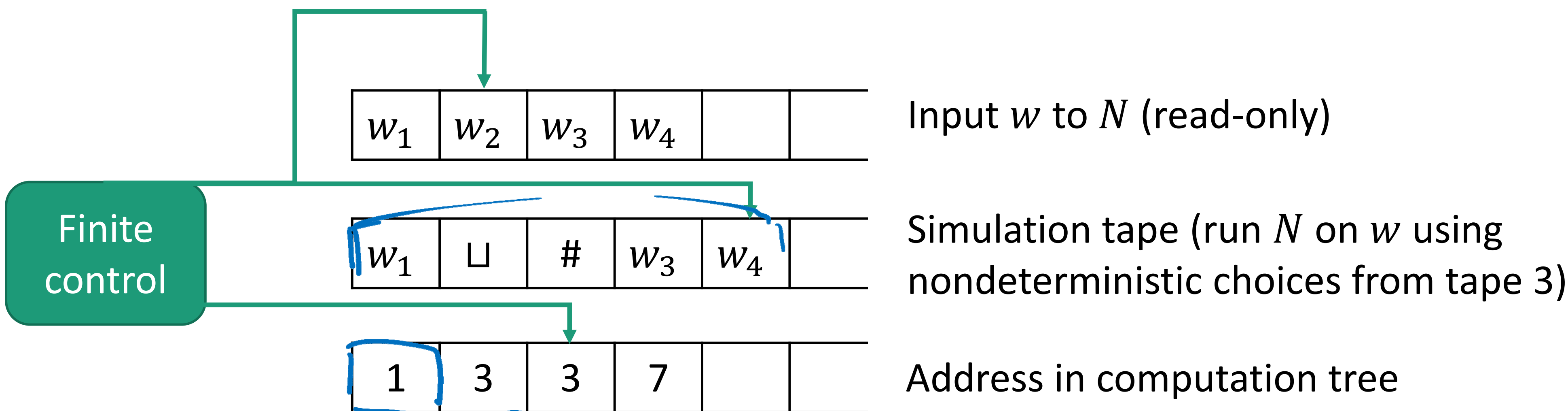
If original NTM is a decider, both DFS and BFS.

# 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)




made 3rd nondet. choice in step 2

made 1st nondet choice in step 1

# 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"

definitional, prescriptive, normative

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

descriptive, empirical, falsifiable

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

# Decidable Languages

# 1928 – The *Entscheidungsproblem*

*The "Decision Problem"*

mathematical statement

*Is there an algorithm which takes as input a formula (in first-order logic) and decides whether it's logically valid?*

- Can every true mathematical statement be proved automatically on a computer?
- Can mathematicians automate themselves out of a job?

# Questions about regular languages

- Given a DFA $D$ and a string $w$, does $D$ accept input $w$?
- Given a DFA $D$, does $D$ recognize the empty language?
- Given DFAs $D_1, D_2$, do they recognize the same language?

(Same questions apply to NFAs, regexes)

Goal: Formulate each of these questions as a language, and decide them using Turing machines

# Questions about regular languages

Design a TM which takes as input a DFA $D$ and a string $w$, and determines whether $D$ accepts $w$

## How should the input to this TM be represented?

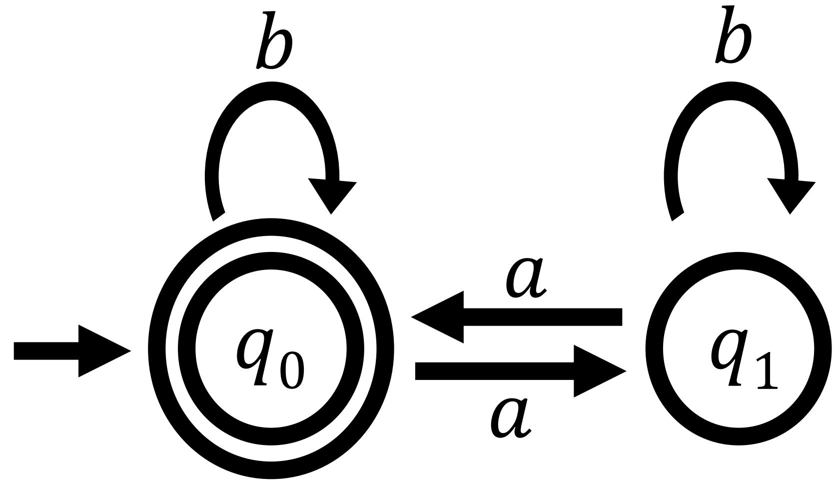
Let $D = (Q, \Sigma, \delta, q_0, F)$. List each component of the tuple separated by #

- Represent $Q$ by ,-separated binary strings
- Represent $\Sigma$ by ,-separated binary strings
- Represent $\delta : Q \times \Sigma \to Q$ by a ,-separated list of triples $(p, a, q)$, …

$\langle \text{blah} \rangle$ means blah.toString()

Denote the encoding of $D, w$ by $\langle D, w \rangle$

# Example




$D = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1\}$ $\quad$ $q_0$ rep'd by $\underline{0}$

$\quad$ $q_1$ rep'd by $\underline{1}$

$\Sigma = \{a, b\}$ $\quad$ $a$ rep'd by $\underline{0}$

$\quad$ $b$ rep'd by $1$

$\delta : Q \times \Sigma \to Q$ $\quad$ $\delta(q_0, a) = q_1$ rep'd by $(0, 0, 1)$

$\quad$ $\delta(q_1, b) = q_1$ rep'd by $(1, 1, 1)$

$$\langle D \rangle = \overbrace{0, 1}^{Q} \# \overbrace{0, 1}^{\Sigma} \# \overbrace{(0,0,1), (0,1,0), (1,0,0), (1,0,1)}^{\delta} \#$$

$$\underbrace{0}_{q_0} \# \underbrace{0}_{F}$$

# Representation independence

Computability (i.e., decidability and recognizability) is **not** affected by the precise choice of encoding

Suppose TM $M$ expects input encoded under $\langle \cdot \rangle$

Want to solve a problem under encoding $[\cdot]$

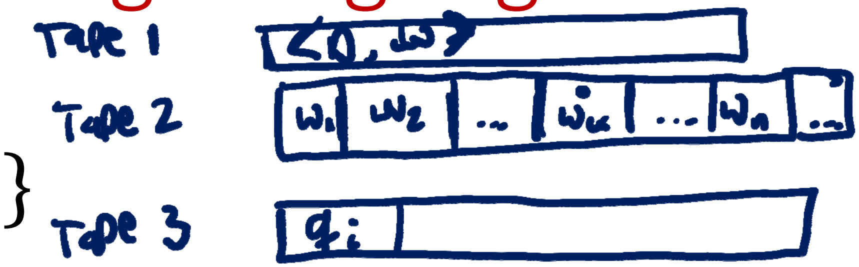
Why? A TM can always convert between different (reasonable) encodings

On input $[O]$:

1) Convert from $[O]$ to $\langle O \rangle$
2) Run $M$ on input $\langle O \rangle$. Accept if accepts, reject if rejects

From now on, we'll take $\langle \quad \rangle$ to mean "some reasonable encoding"

# A "universal" algorithm for recognizing regular languages




$A_{\text{DFA}} = \{\langle D, w\rangle \mid \text{DFA } D \text{ accepts } w\}$

**Theorem:** $A_{\text{DFA}}$ is decidable

Computational problem: Given DFA D, string w, does D accept when run on input w? - (i.e., is $w \in L(D)$?)

**Proof:** Define a (high-level) 3-tape TM $M$ on input $\langle D, w\rangle$:

1. Check if $\langle D, w\rangle$ is a valid encoding (reject if not)

   Should be done by every algorithm, but ok to omit.

2. Simulate $D$ on $w$, i.e.,
   - Tape 2: Maintain $w$ and head location of $D$
   - Tape 3: Maintain state of $D$, update according to $\delta$
3. Accept if $D$ ends in an accept state, reject otherwise

# Other decidable languages

$A_{\mathrm{DFA}} = \{\langle D, w\rangle \mid \text{DFA } D \text{ accepts } w\}$

$A_{\mathrm{NFA}} = \{\langle N, w\rangle \mid \text{NFA } N \text{ accepts } w\}$

$A_{\mathrm{REX}} = \{\langle R, w\rangle \mid \text{regular expression } R \text{ generates } w\}$

# NFA Acceptance

Want a TM that on input NFA N, string w, determines whether N accepts w.

Which of the following describes a **decider** for $A_{\text{NFA}} = \{\langle N, w\rangle \mid \text{NFA } N \text{ accepts } w\}$?

a) Using a deterministic TM, simulate $N$ on $w$, always making the first nondeterministic choice at each step. Accept if it accepts, and reject otherwise.

First choice might not be the right one

b) Using a deterministic TM, simulate all possible choices of $N$ on $w$ for 1 step of computation, 2 steps of computation, etc. Accept whenever some simulation accepts.


c) Use the subset construction to convert $N$ to an equivalent DFA $M$. Simulate $M$ on $w$, accept if it accepts, and reject otherwise.

# Regular Languages are Decidable

Theorem: Every regular language $L$ is decidable

Proof 1: If $L$ is regular, it is recognized by a DFA $D$. Convert this DFA to a TM $M$. Then $M$ decides $L$.

Proof 2: If $L$ is regular, it is recognized by a DFA $D$. The following TM $M_D$ decides $L$.

On input $w$:

$= \{\langle D, w\rangle \mid \text{DFA } D \text{ accepts } w\}$

1. Run the decider for $A_{\text{DFA}}$ on input $\langle D, w\rangle$
2. Accept if the decider accepts; reject otherwise

Correctness:
- If $w \in L$, $D$ accepts $w \Rightarrow \langle D, w\rangle \in A_{DFA} \Rightarrow$ TM accepts
- If $w \notin L$, $D$ rejects $w \Rightarrow \langle D, w\rangle \notin A_{DFA} \Rightarrow$ TM rejects