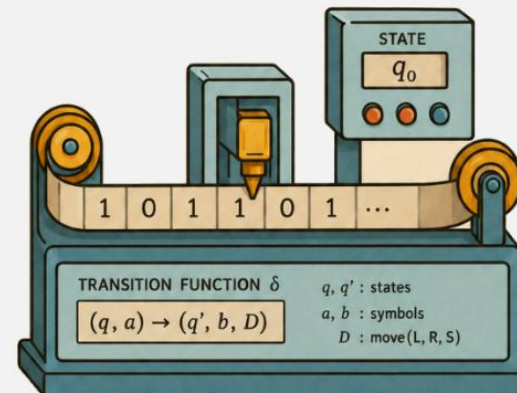


# BU CS 332 – Theory of Computation

---

## Lecture 11:

- Closure Properties
- Nondeterminism
- Church-Turing Thesis
- Decidable Languages



Alexander Poremba  
October 15, 2026

---



# Why Are Multi-Tape TMs Useful?

## Closure Properties

# Closure Properties

---

**Fact:** The Turing-recognizable languages are closed under the union operation:

If  $L_1$  and  $L_2$  are both Turing-recognizable, then  $L_1 \cup L_2$  is also Turing-recognizable.

# Closure Properties

---

**Fact:** The Turing-recognizable languages are closed under the union operation:

If  $L_1$  and  $L_2$  are both Turing-recognizable, then  $L_1 \cup L_2$  is also Turing-recognizable.

On input  $w$ :

1. Scan tapes 1,2,3 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 machines reject, **reject**.

# 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

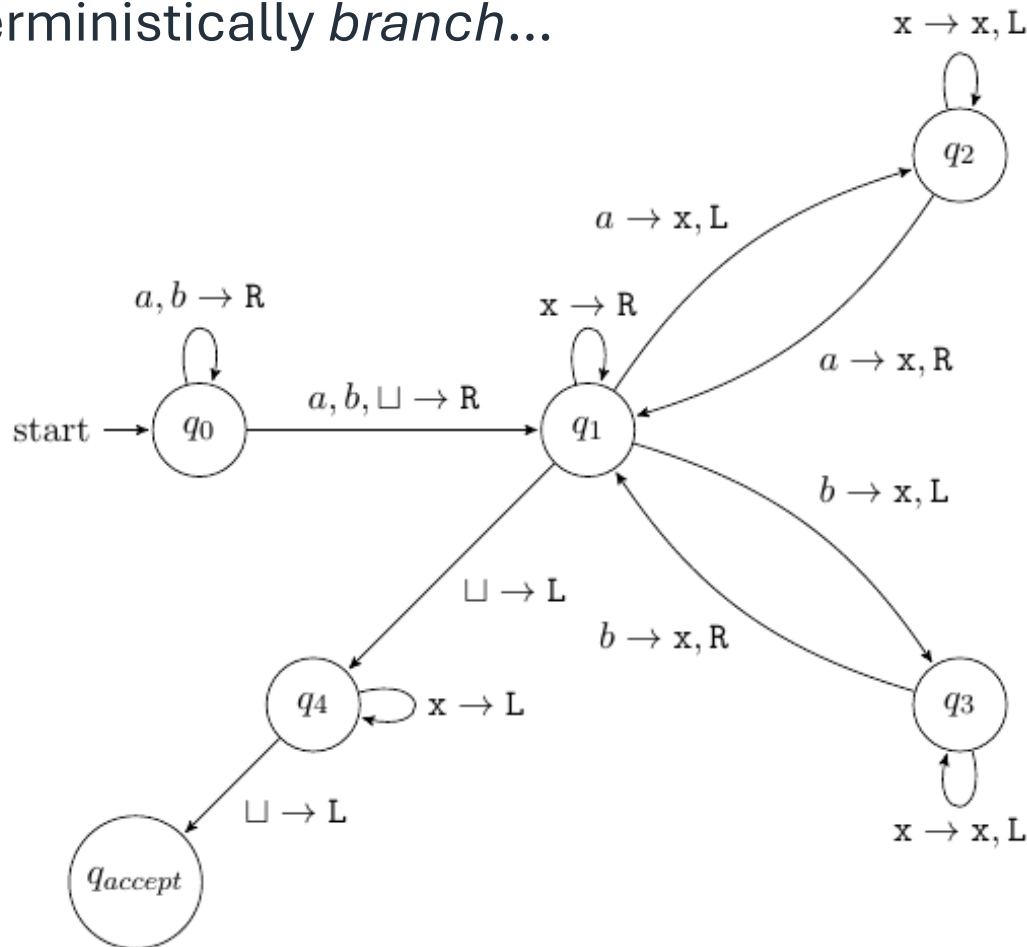
# TMs are equivalent to...

---

- 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
- ...

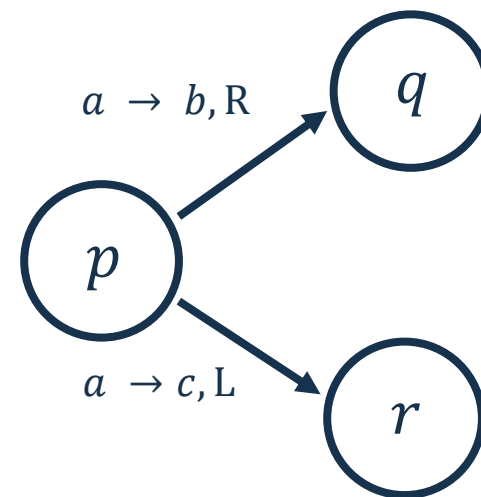
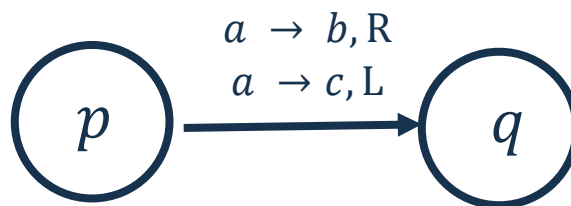
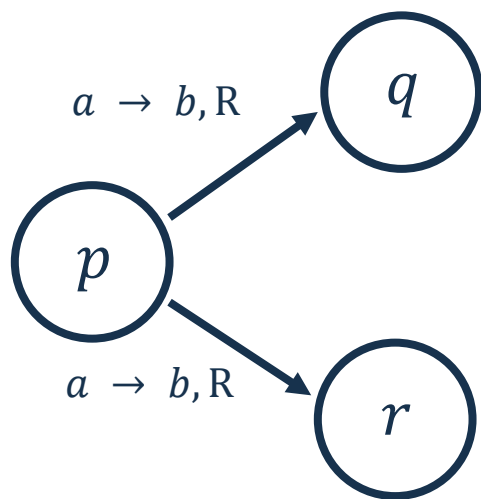
# Nondeterministic TMs

At any point in the computation, the machine may nondeterministically *branch*...



# Nondeterministic TMs

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



# Nondeterministic TMs

---

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

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

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

# Nondeterministic TMs

---

At any point in computation, it may **nondeterministically** branch; accepts iff there is an accepting computation path

**Implementation-Level Description:** On input string  $w$ :

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

2a) Read the next symbol  $s$  and cross it off.

2b) Move the head left repeatedly until a non- $x$  symbol is found. If it matches  $s$ , cross it off. Else, **reject**.

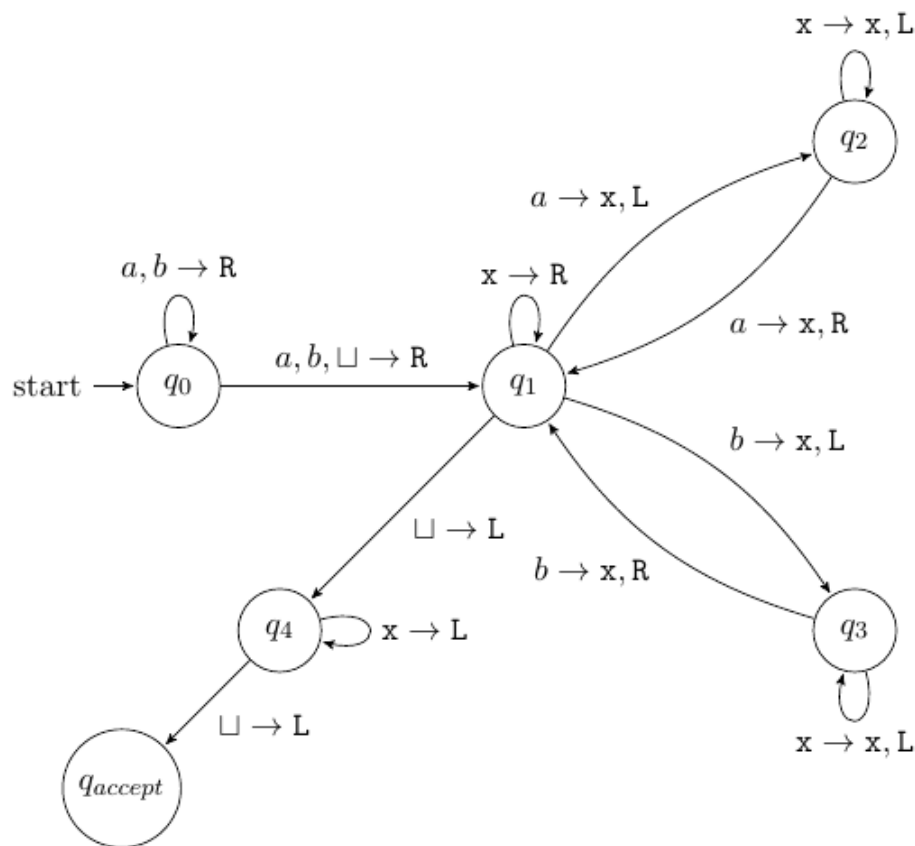
2c) Move the head right until a non- $x$  symbol is found. If blank is hit, go to step 3.

2d) Go back to 2a).

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

## Question: Nondeterministic TMs

At any point in computation, it may **nondeterministically** branch; accepts iff there is 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)  $\{ ww \mid w \in \{a, b, x\}^* \}$
- d)  $\{ wx^n w^R \mid w \in \{a, b\}^*, n \geq 0 \}$

# Nondeterministic TMs

---

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

**Proof idea:** Explore “tree of possible computations”

## Question: Simulating NTMs

---

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

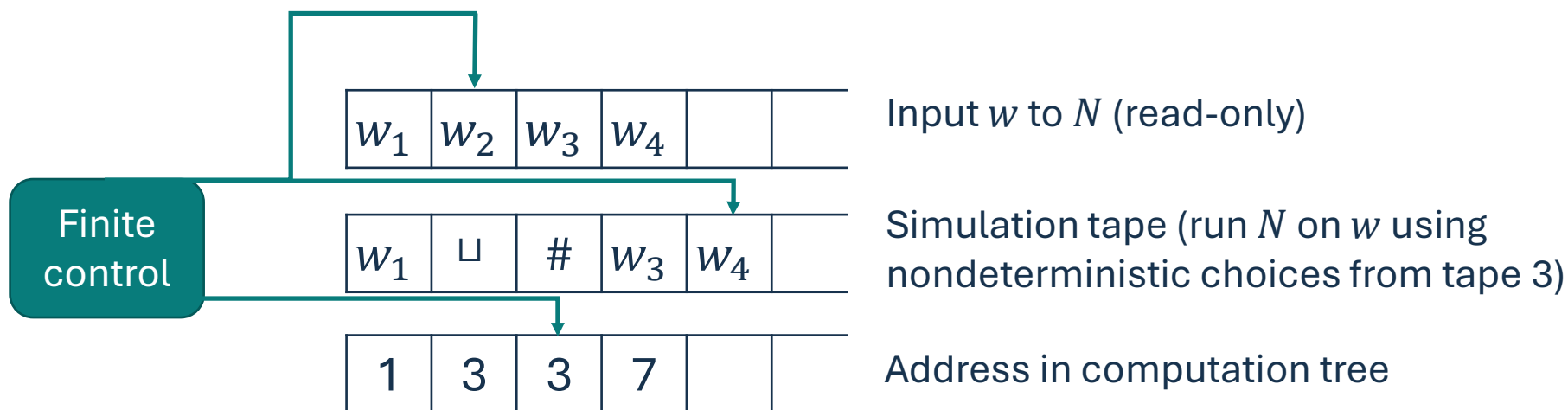
- a) **Depth-first search:** Explore as far as possible down each branch before backtracking
- b) **Breadth-first search:** Explore all configurations at depth 1, then all configurations at depth 2, etc.
- c) **Both** algorithms will always work



# 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

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

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



# Decidable Languages

# 1928 – The *Entscheidungsproblem*

---

## *The “Decision Problem”*

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



# 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?

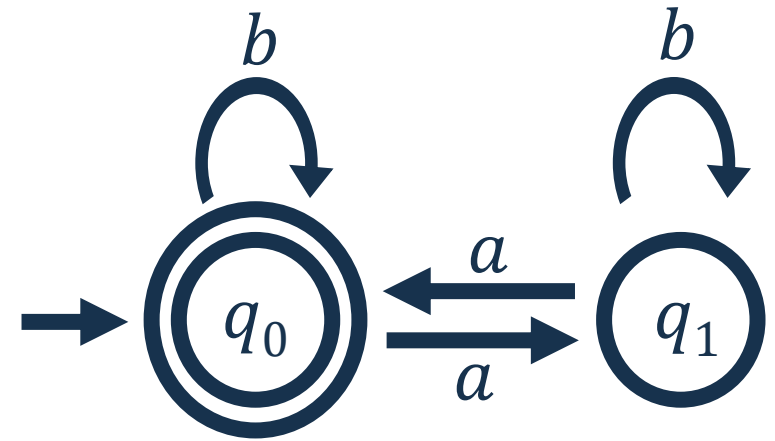
Let  $D = (Q, \Sigma, \delta, q_0, F)$ . List each component of the tuple separated by #

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

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

# Example

---



# Representation independence

---

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

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

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

**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)
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

---

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

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

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

## Question: NFA Acceptance

---

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.
- 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  $w$ :

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

# Classes of Languages

---

