# BU CS 332 – Theory of Computation

https://forms.gle/1NhrfwEdVXhpinPV8

## Lecture 12:

- Church-Turing Thesis
- Decidable Languages
- Universal TM

Reading:

Sipser Ch 3.3, 4.1

HW 5 due 11:59 PM on Friday.

Mark Bun

March 6, 2024

# Last Time: 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

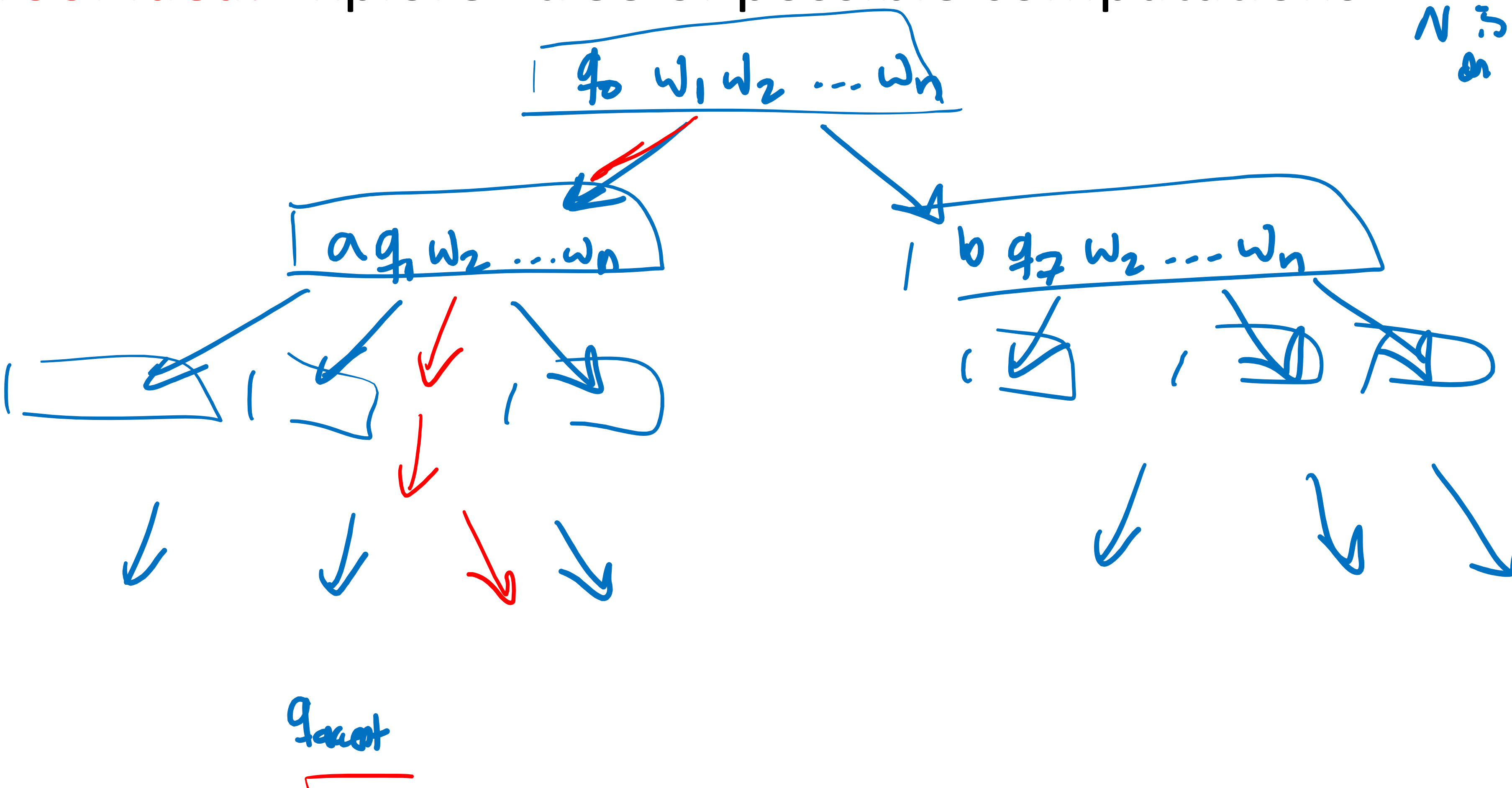
$w \in L(N) \Rightarrow$ there exists a branch of $N$'s computation leading it to accept input $w$

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

# Nondeterministic TMs

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

**Proof idea:** Explore “tree of possible computations” when NTM $N$ is run on input $w$

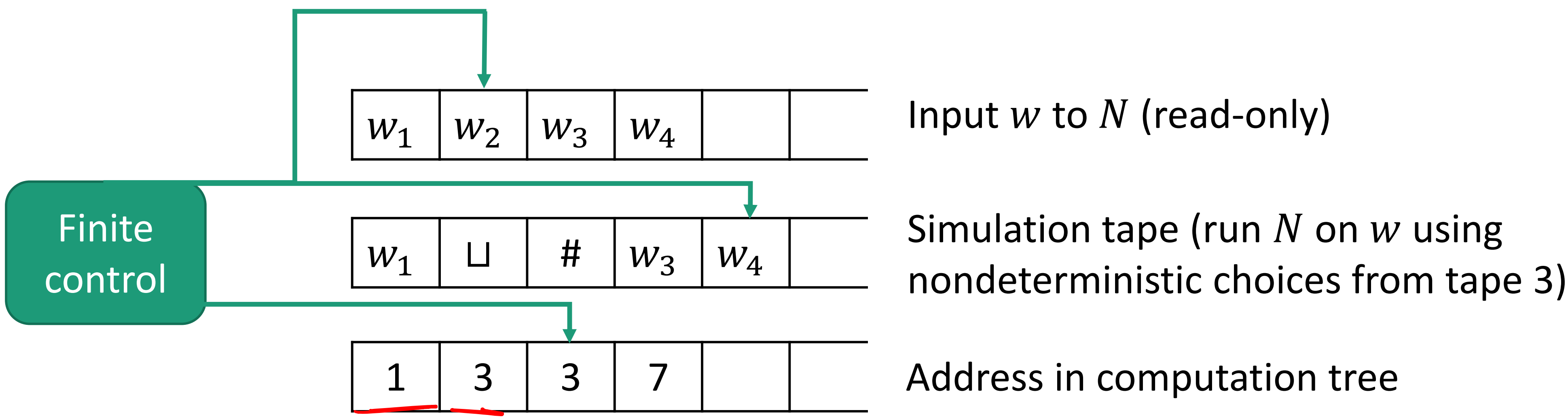


# Nondeterministic TMs

**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"

prescriptive, normative, definitional

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?*

Questions:

- Can we hope to automatically prove theorems?
- 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 $w$

Describe an encoding or "toString" method for DFAs

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 $(p, a, q)$, …

$\langle \cdot \rangle$ means $\cdot$.toString()

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

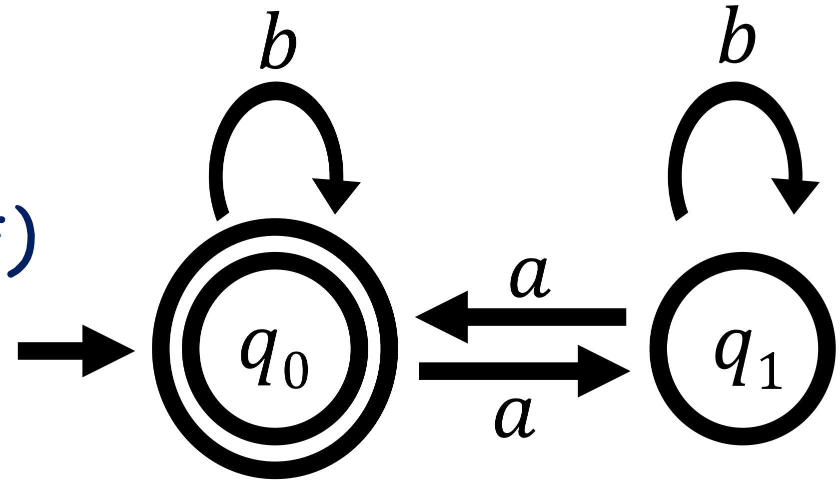
# Example

$\Sigma_1 = \{\underset{00}{a}, \underset{01}{b}, \underset{10}{c}\}$

$Q = \{\overset{0}{q_0}, \overset{1}{q_1}\}$

$\Sigma_1 = \{\overset{0}{a}, \overset{1}{b}\}$

$D = (Q, \Sigma_1, \delta, q_0, F)$




$\delta(q_0, a) = q_1, \ldots$

Start state $q_0$

Accept state $q_0$

$\langle D \rangle = \overbrace{0, 1}^{\text{state set}} \# \overbrace{0, 1}^{\text{alphabet}} \# \overbrace{(0, 0, 1), (0, 1, 0), (1, 0, 0), (1, 1, 1)}^{\text{transition function}}$

$\# \underbrace{0}_{\text{start}} \# \underbrace{0}_{\text{accept}}$

# Representation independence

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

If TM $M$ decides some language involving DFAs under encoding $\langle \cdot \rangle$

Have a different encoding $[\cdot]$

Why? A TM can always convert between different (reasonable) encodings The following TM $N$ decides that language under encoding $[\cdot]$:

On input $[D]$:

1) Convert $[D]$ to $\langle D \rangle$
2) Run $M$ on input $\langle D \rangle$. Accept if accepts.
   Reject if rejects.

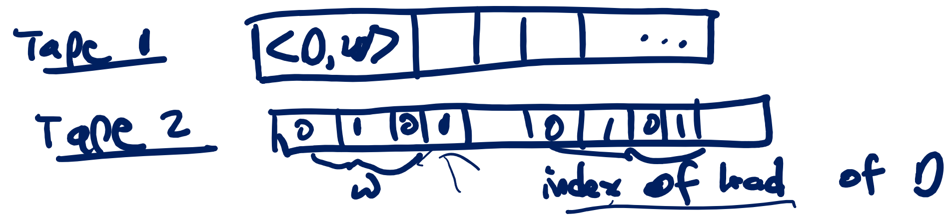
From now on, we'll take ⟨ ⟩ to mean "some reasonable encoding"

# A "universal" algorithm for recognizing regular languages

Problem: Given DFA $D$ and string $w$, does $D$ accept input $w$?

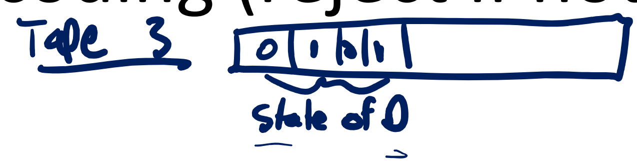
$$A_{\mathrm{DFA}} = \{\langle D, w\rangle \mid \text{DFA } D \text{ accepts } w\}$$

**Theorem:** $A_{\mathrm{DFA}}$ 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




A necessary part of every alg., but you can feel free to omit from your descriptions

# 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

Given an NFA $N$ and string $w$, does $N$ accept $w$?

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.

Might miss an accepting branch

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.

Might loop forever if NFA does not accept $w$.

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 $w$:

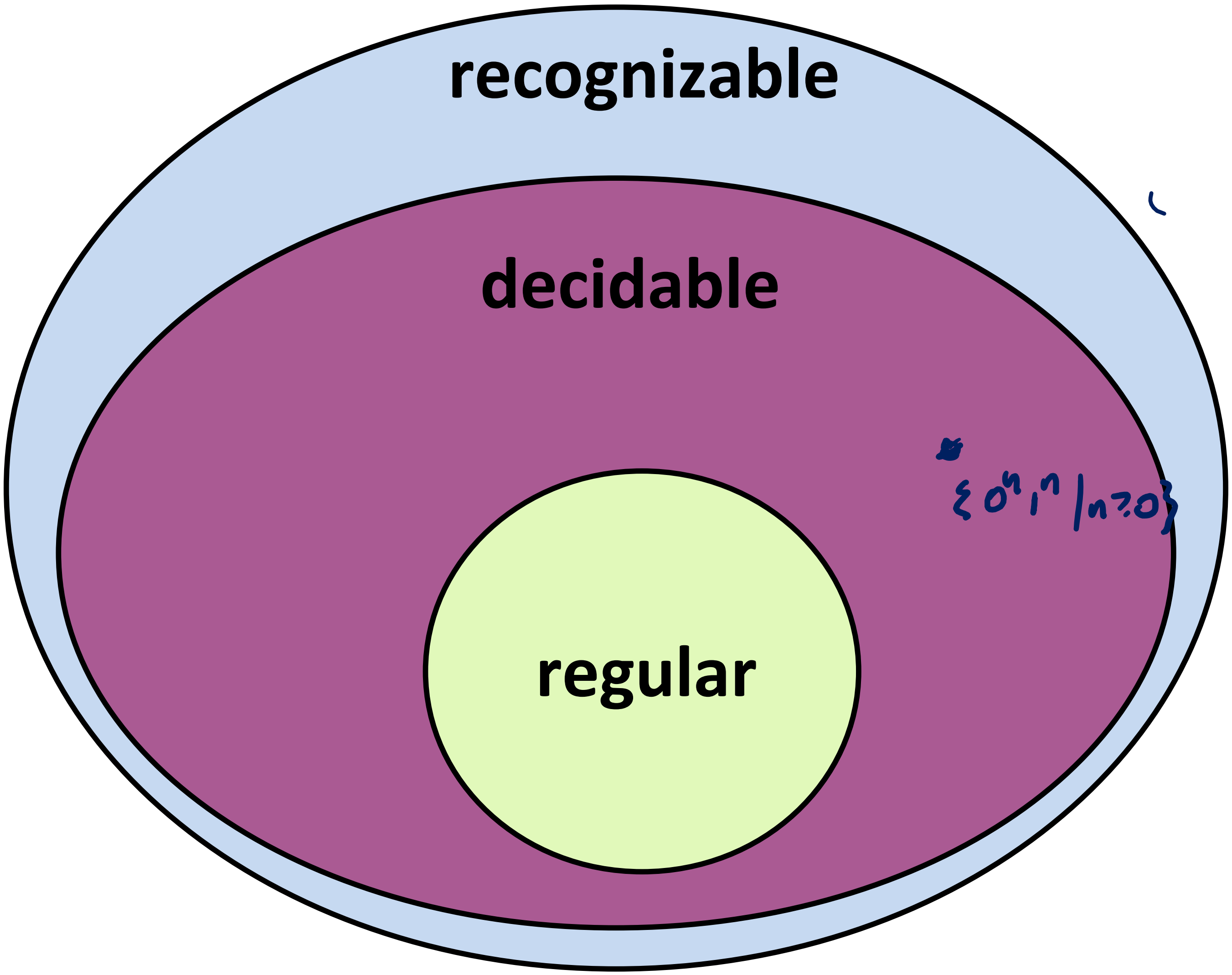
1. Run the decider for $A_{\text{DFA}}$ on input $\langle D, w\rangle$
2. Accept if the decider accepts; reject otherwise

Claim: $M_D$ decides $L$

Proof: $w \in L \Rightarrow D$ accepts $w \Rightarrow \langle D, w\rangle \in A_{DFA} \Rightarrow$ decider accepts

$w \notin L \Rightarrow D$ rejects $w \Rightarrow \langle D, w\rangle \notin A_{DFA} \Rightarrow$ decider rejects

# Classes of Languages



# More Decidable Languages: Emptiness Testing

Theorem: $E_{\text{DFA}} = \{\langle D \rangle \mid D \text{ is a DFA such that } L(D) = \emptyset\}$ is decidable

Computational problem: Given a DFA $D$, does $D$ recognize the empty language?

Proof: The following TM decides $E_{\text{DFA}}$

On input $\langle D \rangle$, where $D$ is a DFA with $k$ states:

1. Perform $k$ steps of breadth-first search on state diagram of $D$ to determine if an accept state is reachable from the start state

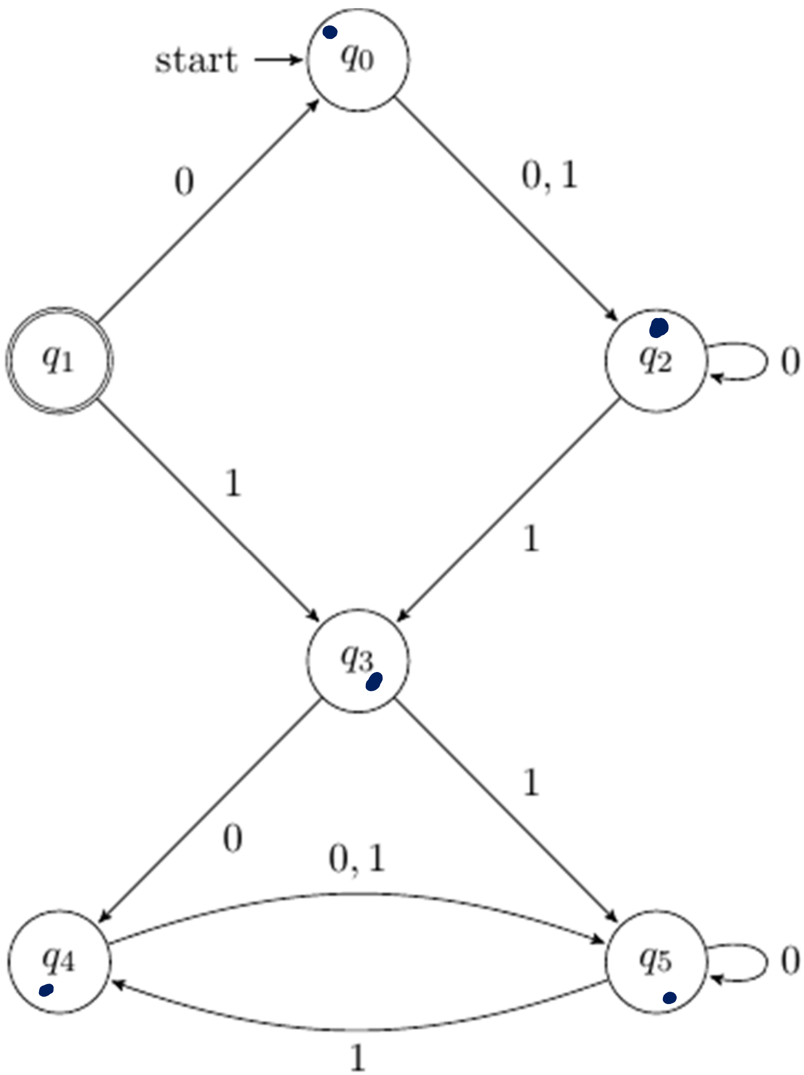
2. Reject if a DFA accept state is reachable; accept otherwise

$\langle D \rangle \in E_{DFA} \Rightarrow D$ rejects every string $w \Rightarrow$ impossible to reach an accept state from start state of $D$
$\Rightarrow$ BFS fails $\Rightarrow$ alg accepts

$\langle D \rangle \notin E_{DFA} \Rightarrow D$ accepts some string $w \Rightarrow$ is possible to reach an accept state
$\Rightarrow$ BFS succeeds $\Rightarrow$ alg rejects.

# $E_{DFA}$ Example

$D =$




BFS

1) $q_0$
2) $q_2$
3) $q_3$
4) $q_4$, $q_5$

Conclude accept state(s) not reachable

$\Rightarrow L(D) = \phi$

$\Rightarrow$ alg. accepts.

# New Deciders from Old: Equality Testing

$EQ_{\text{DFA}} = \{\langle D_1, D_2\rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2)\}$

Problem: Given DFAs $D_1, D_2$, do they recognize the same language?

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

**Proof:** The following TM decides $EQ_{\text{DFA}}$

On input $\langle D_1, D_2\rangle$, where $\langle D_1, D_2\rangle$ are DFAs:

1. Construct DFA $D$ recognizing the **symmetric difference** $L(D_1) \triangle L(D_2)$ $= \{w \in \Sigma^* \mid w \text{ is in exactly one of } D_1 \text{ or } D_2\}$
2. Run the decider for $E_{\text{DFA}}$ on $\langle D\rangle$ and return its output

Correctness:

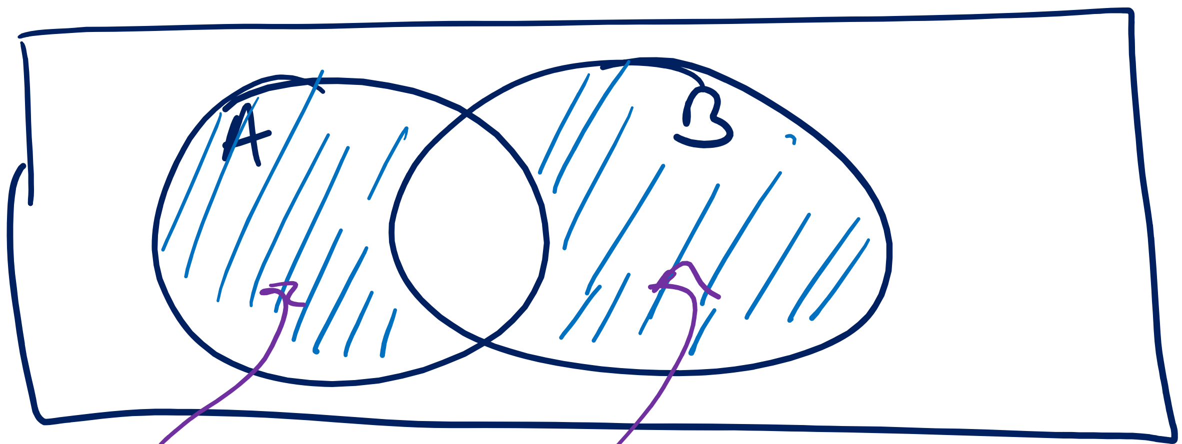
If $\langle D_1, D_2\rangle \in EQ_{DFA}$, then $\forall w$, $D_1$ accepts $w$ and $D_2$ accepts $w$
or $D_1$ rejects $w$ and $D_2$ rejects $w$

$\Rightarrow L(D_1) \triangle L(D_2) = \emptyset \Rightarrow \langle D\rangle \in E_{DFA} \Rightarrow$ accept.

If $\langle D_1, D_2\rangle \notin EQ_{DFA}$, $\Rightarrow L(D_1) \triangle L(D_2) \neq \emptyset \Rightarrow \langle D\rangle \notin E_{DFA} \Rightarrow$ reject.

# Symmetric Difference

$$A \triangle B = \{w \mid w \in A \text{ or } w \in B \text{ but not both}\}$$




$A \triangle B = (A \cap \overline{B}) \cup (B \cap \overline{A})$

Given DFAs $D_1$ and $D_2$, can use closure property constructions for intersection, complement, union to construct a DFA for $A \triangle B$.
(may go through NFAs + subset construction)

# Universal Turing Machine

# Meta-Computational Languages

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

$A_{\text{TM}} = \{\langle M, w\rangle \mid \text{TM } M \text{ accepts } w\}$

$E_{\text{DFA}} = \{\langle D\rangle \mid \text{DFA } D \text{ recognizes the empty language } \emptyset\}$

$E_{\text{TM}} = \{\langle M\rangle \mid \text{TM } M \text{ recognizes the empty language } \emptyset\}$

$EQ_{\text{DFA}} = \{\langle D_1, D_2\rangle \mid D_1 \text{ and } D_2 \text{ are DFAs}, L(D_1) = L(D_2)\}$

$EQ_{\text{TM}} = \{\langle M_1, M_2\rangle \mid M_1 \text{ and } M_2 \text{ are TMs}, L(M_1) = L(M_2)\}$

# The Universal Turing Machine

$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input } w\}$

**Theorem:** $A_{\text{TM}}$ is Turing-recognizable

The following "Universal TM" $U$ recognizes $A_{\text{TM}}$

On input $\langle M, w\rangle$:

1. Simulate running $M$ on input $w$
2. If $M$ accepts, accept. If $M$ rejects, reject.

# Universal TM and $A_{\mathrm{TM}}$

Why is the Universal TM not a decider for $A_{\mathrm{TM}}$?

$A_{TM} = \{\langle M, w\rangle \mid$ TM $M$ accepts input $w\}$

The following "Universal TM" $U$ recognizes $A_{\mathrm{TM}}$

On input $\langle M, w\rangle$:

1. Simulate running $M$ on input $w$
2. If $M$ accepts, accept. If $M$ rejects, reject.

a) It may reject inputs $\langle M, w\rangle$ where $M$ accepts $w$

b) It may accept inputs $\langle M, w\rangle$ where $M$ rejects $w$

c) It may loop on inputs $\langle M, w\rangle$ where $M$ loops on $w$

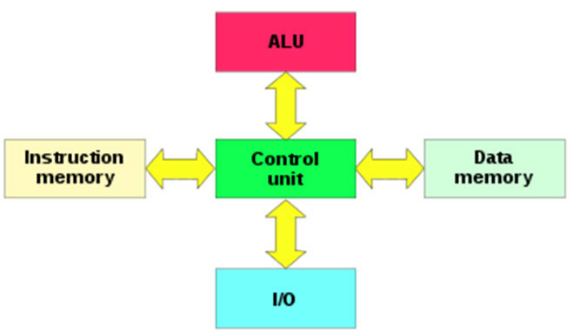
d) It may loop on inputs $\langle M, w\rangle$ where $M$ accepts $w$

# More on the Universal TM

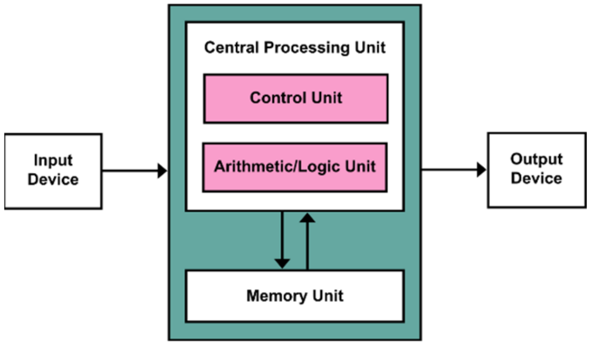
"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine **U** is supplied with a tape on the beginning of which is written the S.D ["standard description"] of some computing machine **M**, then **U** will compute the same sequence as **M**."

- Turing, "On Computable Numbers…" 1936

- Foreshadowed general-purpose programmable computers
- No need for specialized hardware: Virtual machines as software




Harvard architecture:
Separate instruction and data pathways




von Neumann architecture:
Programs can be treated as data