# CSE 105
## THEORY OF COMPUTATION

"Winter" 2018

http://cseweb.ucsd.edu/classes/wi18/cse105-ab/

# Today's learning goals

Sipser Section 4.1

- Explain what it means for a problem to be decidable.
- Justify the use of encoding.
- Give examples of decidable problems.

# High-level description = Algorithm

**Church-Turing thesis**

**Each algorithm can be implemented by some Turing machine.**

# Algorithms

So far: machines describing / recognizing sets.

What questions can we answer?

- Is a string a palindrome?
- Does a string have even length?

**Answering these questions is the same as describing the set of strings for which the answer is yes.**

# Computational problems

Sipser p. 194

A computational problem is **decidable** iff the language encoding the problem instances is decidable

- Does a specific DFA accept a given string? encoded by { representations of DFAs M and strings w such that w is in L(M) }
- Is the language generated by a specific CFG empty? encoded by { representations of CFGs G such that L(G) = Ø }
- Is a Turing machine a decider? encoded by
  { representations of Turing machines M such that M always halts }

# Representations for computational problems

*Sipser p. 185*

To decide these problems, we need to represent the objects of interest as **strings**

For inputs that aren't strings, we have to **encode the object** (represent it as a string) first

To define TM M:

"On input w …

1. ..
2. ..
3. …

**Notation:**

**<O>** is the **string** that represents (encodes) the object O

$\langle O_1, \ldots, O_n \rangle$ is the single string that represents the tuple of objects $O_1, \ldots, O_n$

# Representations for computational problems

*Sipser p. 185*

To decide these problems, we need to represent the objects of interest as **strings**

For inputs that aren't strings, we have to **encode the object** (represent it as a string) first

To define TM M:

"On input w …

1. ..
2. ..
3. …

**Assumption:**
**There are Turing machine subroutines that can decode the string representations of common objects so we can interact with them as intended**
e.g. from string representation of Turing machine, can decode $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$

# Representations for computational problems

Sipser p. 185

To decide these problems, we need to represent the objects of interest as **strings**

- $B = (\underbrace{\{q_0, q_1\}}_{Q}, \underbrace{\{0,1\}}_{\Sigma}, \underbrace{(\delta(q_0, 0) = q_1, \delta(q_0, 1) = q_1, \delta(q_1, 0) = q_0, \delta(q_1, 1) = q_0)}_{\delta}, q_0, \underbrace{\{q_1\}}_{F})$

$w = 010$

$q_0 \quad q_1$

# Encoding inputs

**Payoff**: problems we care about can be reframed as languages of strings

e.g. "Recognize whether a string is a palindrome."

$\{ w \mid w \text{ in } \{0,1\}^* \text{ and } w = w^R \}$

e.g. "Check whether a string is accepted by a DFA."

$\{ \langle B,w \rangle \mid B \text{ is a DFA over } \Sigma, w \text{ in } \Sigma^*, \text{ and } w \text{ is in } L(B) \}$ $= A_{DFA}$

e.g. "Check whether the language of a PDA is infinite."

$\{ \langle A \rangle \mid A \text{ is a PDA and } L(A) \text{ is infinite} \}$

# Computational problems

Does a specific DFA accept a given string? encoded by { representations of DFAs M and strings w such that w is in L(M) }

Define using high-level description a Turing machine M = "On input <B,w>, where B is a DFA and w is a string:

1. Type check encoding to check input is valid type.
2. Simulate B on input w (by keeping track of states in B, transition function of B, etc.)
3. If the simulations ends in an accept state of B, *accept*. If it ends in a non-accept state of B, *reject*. "

# Computational problems

- **Recall**: in high-level descriptions, can simulate (run) other Turing machines / algorithms as a subroutine of program being defined.

- **To prove decidable**: need to confirm that strings in the language are *accepted* **and** that strings not in the language are *rejected* (no looping allowed).

# Computational problems

Define using high-level description a Turing machine M = "On input <B,w>, where B is a DFA and w is a string:

1. Type check encoding to check input is valid type.
2. Simulate B on input w (by keeping track of states in B, transition function of B, etc.)
3. If the simulations ends in an accept state of B, *accept*. If it ends in a non-accept state of B, *reject*. "

***Why is M a decider?***

# Computational problems

*Vocabulary*

$A_{XX}$ "Is a given string accepted by a given machine of type XX?"

{ <B,w> | B is a XX over Σ, w in Σ*, and w is in L(B) } = $A_{xx}$

$E_{XX}$ "Is the language of a machine of type XX is empty?"

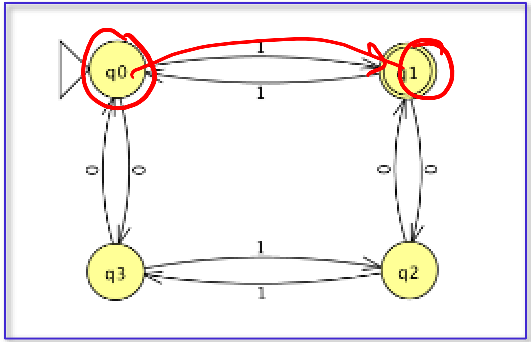
{ <A> | A is a XX over Σ, L(A) is empty }

$EQ_{XX}$ "Are the languages of two given machines of type XX equal?"

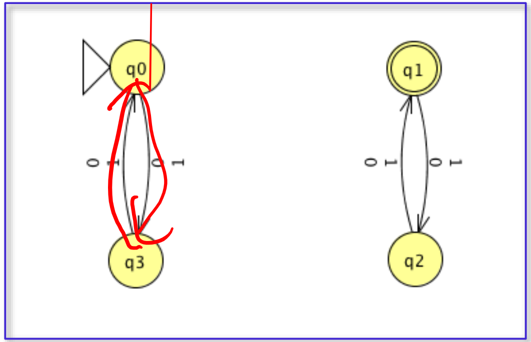
{ <A, B> | A and B are XX over Σ, L(A) = L(B)}

# For DFA

M1




M2




A. <M1, 1> is in $A_{DFA}$ ✓
~~B.~~ <M2, 01> is in $A_{DFA}$
~~C.~~ <M1> is in $E_{DFA}$
~~D.~~ <M1, M2> is in $EQ_{DFA}$
E. More than one of the above

$\langle M2 \rangle \in E_{DFA}$

# Computational problems

When the model in question is DFA: which of the following computational problems are **decidable**?

A. $A_{DFA}$
B. $E_{DFA}$
C. $EQ_{DFA}$
D. All of the above
E. None of the above

# Proving decidability

**Claim:** $E_{DFA}$ is decidable

**Proof:** WTS that { <A> | A is a DFA over Σ, L(A) is empty } is decidable.

*Informally: what do you look for in the state diagram of a DFA to determine if it accepts *at least one* string?*

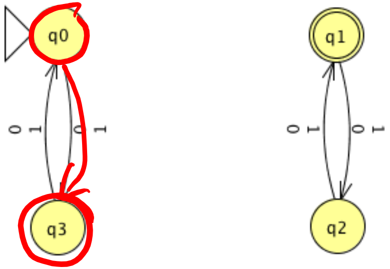
A path from $q_0$ to something in F

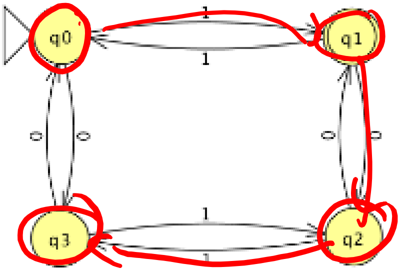
# Proving decidability

**Claim:** $E_{DFA}$ is decidable

**Proof:** WTS that { <A> | A is a DFA over Σ, L(A) is empty } is decidable.

*Test cases:*

e.g.< 

> is in $E_{DFA}$ ; < 

> is not in $E_{DFA}$

TM deciding $E_{DFA}$ should accept

and should reject

# Proving decidability

**Claim:** $E_{DFA}$ is decidable

**Proof:** WTS that { <A> | A is a DFA over Σ, L(A) is empty } is decidable.

## Step 1: construction

*Idea: breadth-first search in state diagram to look for paths to F*

$q_0 \quad q_1 \quad q_2 \quad q_3 \; \ldots$

$(q_0, 0) = q_2 \; , \; (q_0, 1) = q_3 \; \ldots$

# Proving decidability

**Claim:** $E_{DFA}$ is decidable
**Proof:** WTS that { <A> | A is a DFA over Σ, L(A) is empty } is decidable.

## Step 1: construction

*Idea: breadth-first search in state diagram to look for paths to F*

Define TM $M_2$ by: $M_2$ = "On input <A>:

1. Check whether input is a valid encoding of a DFA; if not, reject.
2. Mark the start state of A
3. Repeat until no new states get marked:
    i. Loop over states of A and mark any unmarked state that has an **incoming** edge from a marked state.
4. If no final state of A is marked, *accept*; otherwise, *reject.*"

# Proving decidability

## Step 2: correctness proof

WTS (1) $L(M_2) = E_{DFA}$ and (2) $M_2$ is a decider.

(1) $\Rightarrow$ $L(M_2) \subseteq E_{DFA}$: let $w \in L(M_2)$ $\Rightarrow$ $w$ is an encoding of a DFA s.t. there is no paths from start to accept.

$\Rightarrow L(w) = \{\} \Rightarrow w \in E_{DFA}$.

---

($\Leftarrow$) exercise $E_{DFA} \subseteq L(M_2)$

---

2 _ _ _ _ _

# Proving decidability


**Claim:** $EQ_{DFA}$ is decidable

**Proof:** WTS that { <A,B> | A and B are DFAs over Σ, L(A) = L(B)} is decidable.

$E_{DFA}$ is decidable

Can we use other theorem to show this?

use closure properties.

$$L(M) = \left(\overline{L(A)} \cap L(B)\right) \cup \left(L(A) \cap \overline{L(B)}\right)$$

# For next time

**GroupHW5 due** Saturday, February 24

For Monday, pre-class reading: Section 4.1, Theorem 4.5 (page 197) and Theorem 4.8 (page 199)