Today's learning goals

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

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

*We won't specify the encoding.*

To prove decidable, define TM $M = "On input <…>, 1. 2. … "$

Show (1) $L(M) = …$ and (2) $M$ is a decider.
Computational problems over $\Sigma$

$A_{DFA}$ "Is a given string accepted by a given DFA?"
\[
\{ \langle B, w \rangle \mid B \text{ is a DFA, } w \text{ in } \Sigma^*, \text{ and } w \text{ is in } L(B) \} \]

$E_{DFA}$ "Is the language of a DFA empty?"
\[
\{ \langle A \rangle \mid A \text{ is a DFA over } \Sigma, L(A) \text{ is empty} \} \]

$EQ_{DFA}$ "Are the languages of two given DFAs equal?"
\[
\{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFA over } \Sigma, L(A) = L(B) \} \]
From last class

M₁ = "On input <B,w>, where B is a DFA and w is a string:
1. Simulate B on input w (by keeping track of states in B, transition function of B, etc.)
2. If the simulations ends in an accept state of B, accept. If it ends in a non-accept state of B, reject."

M₂ = "On input <A>, where A is a DFA:
1. Mark the start state of A.
2. 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.
3. If no final state of A is marked, accept; otherwise, reject."
Non-emptiness?

$E'_{DFA}$ "Is the language of a DFA non-empty?"

Is this problem decidable?

A. Yes, using $M_3$ in the handout.
B. Yes, using $M_4$ in the handout.
C. Yes, both $M_3$ and $M_4$ work.
D. Yes, but not using the machines in the handout.
E. No.
Proving decidability

Claim: \( \text{EQ}_{\text{DFA}} \) is decidable

Proof: WTS that \( \{ <A, B> \mid A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} \) is decidable. Idea: give high-level description

Step 1: construction

Will we be able to simulate A and B?
What does set equality mean?
Can we use our previous work?
Proving decidability

**Claim:** $\text{EQ}_{\text{DFA}}$ is decidable

**Proof:** WTS that \{ \langle A, B \rangle \mid A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} is decidable. **Idea:** give high-level description

**Step 1: construction**

Will we be able to simulate $A$ and $B$?

What does set equality mean?

Can we use our previous work?

\[ X = Y \iff (X \cap Y^c) \cup (Y \cap X^c) = \emptyset \]
Proving decidability

Claim: $\text{EQ}_{\text{DFA}}$ is decidable

Proof: WTS that $\{ <A, B> | A, B \text{ are DFA over } \Sigma, L(A) = L(B) \}$ is decidable. Idea: give high-level description

Step 1: construction

Very high-level:

Build new DFA recognizing symmetric difference of $L(A)$, $L(B)$. Check if this set is empty.

$$X = Y \text{ iff } ( (X \cap Y^c) \cup (Y \cap X^c) ) = \emptyset$$
Claim: $\text{EQ}_{\text{DFA}}$ is decidable

Proof: WTS that $\{ <A, B> | A, B \text{ are DFA over } \Sigma, L(A) = L(B) \}$ is decidable. Idea: give high-level description

Step 1: construction

Define TM $M_5$ by: $M_5 = "\text{On input } <A,B> \text{ where } A,B \text{ DFAs:}"

1. Construct a new DFA, $D$, from $A,B$ using algorithms for complementing, taking unions of regular languages such that $L(D) = \text{symmetric difference of } L(A) \text{ and } L(B)$.  
2. Run machine $M_2$ on $<D>$.  
3. If it accepts, accept; if it rejects, reject."
Proving decidability

Step 1: construction
Define TM $M_5$ by: $M_5 = "On\ input\ <A,B>\ where\ A,B\ DFAs$

1. Construct a new DFA, $D$, from $A,B$ using algorithms for complementing, taking unions of regular languages such that $L(D) = \text{symmetric difference of } L(A) \text{ and } L(B)$.
2. Run machine $M_2$ on $<D>$.
3. If it accepts, accept; if it rejects, reject."

Step 2: correctness proof
WTS (1) $L(M_5) = EQ_{DFA}$ and (2) $M_5$ is a decider.
Computational problems

Which of the following computational problems are decidable?

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

Compare:

A. $A_{\text{rex}} = A_{\text{nfa}} = A_{\text{dfa}}$, $E_{\text{rex}} = E_{\text{nfa}} = E_{\text{dfa}}$, $\text{eq}_{\text{rex}} = \text{eq}_{\text{nfa}} = \text{eq}_{\text{dfa}}$

B. They're all decidable, some are equal and some not.

C. They're of different types so all are different.

D. None of the above
Techniques

- **Subroutines**: can use decision procedures of decidable problems as subroutines in other algorithms
  - $A_{DFA}$
  - $E_{DFA}$
  - $EQ_{DFA}$

- **Constructions**: can use algorithms for constructions as subroutines in other algorithms
  - Converting DFA to DFA recognizing complement (or Kleene star).
  - Converting two DFA/NFA to one recognizing union (or intersection, concatenation).
  - Converting NFA to equivalent DFA.
  - Converting regular expression to equivalent NFA.
  - Converting DFA to equivalent regular expression.
Next time

- Are all computational problems decidable?

For Monday, pre-class reading: Section 4.3, page 207-209.