Today's learning goals

- Explain what it means for a problem to be decidable.
- Justify the use of encoding.
- Give examples of decidable problems.
- Prove that a computational problem about DFA, NFA, RegExp, etc. is decidable.

Group HW5 due Saturday
Practice Qs for exam 2 on website - Review Session Mon
Recommended additional problems on Piazza
Extra OH early next week
Computational problems over $\Sigma$

$A_{DFA}$ "Is a given string accepted by a given DFA?"
\{ $<B, w>$ | $B$ is a DFA, $w$ in $\Sigma^*$, and $w$ is in $L(B)$ \}

$E_{DFA}$ "Is the language of a DFA empty?"
\{ $<A>$ | $A$ is a DFA over $\Sigma$, $L(A)$ is empty \}

$EQ_{DFA}$ "Are the languages of two given DFAs equal?"
\{ $<A, B>$ | $A$ and $B$ are DFA over $\Sigma$, $L(A) = L(B)$ \}
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 $\text{TM } M = \text{"On input } <\ldots>,

1.

2. \ldots \text{"}

Show (1) $L(M) = \ldots$ and (2) $M$ is a decider.
Proving decidability

Claim: \( E_{DFA} \) is decidable

Proof: WTS that \( \{ <A> \mid A \text{ is a DFA over } \Sigma, L(A) \text{ is empty} \} \) is decidable.

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

\[ F = \emptyset \text{ is suff but not nec.} \]

\[ \rightarrow 0 \rightarrow 0 \rightarrow 0 \rightarrow \ldots \rightarrow 0 \]

no accept state is reachable from 0.
Proving decidability

Claim: $E_{\text{DFA}}$ is decidable

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

Test cases:

- e.g. $< \text{B} >$ is in $E_{\text{DFA}}$; $< \text{A} >$ is not in $E_{\text{DFA}}$

TM deciding $E_{\text{DFA}}$ should accept and should reject
Proving decidability

Claim: $E_{\text{DFA}}$ is decidable
Proof: WTS that \{ $\langle A \rangle$ | $A$ is a DFA over $\Sigma$, $L(A)$ is empty \} is decidable.

Step 1: construction

Idea: breadth-first search in state diagram to look for paths to $F$
Proving decidability

Claim: $E_{DFA}$ is decidable
Proof: WTS that $\{ <A> | A$ is a DFA over $\Sigma$, $L(A)$ 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: $M_2 = \langle$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.$\rangle$
Proving decidability

Step 2: correctness proof

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

First, let \( w \in \Sigma^* \) and suppose \( M_2 \) accepts \( w \).

Trace \( M_2 \) on \( w \). \( M_2 \) in steps 2-3 does a BFS on graph underlying state diagram of \( A \) and marks all states reachable from \( q_0 \) (guard to terminate b/c \( \# \) of states in \( A \) is finite). Since assume \( L(A) = \emptyset \), no states from
If any marked, so in step 4 $M_2$ accepts $w$.

Next, assume $w \notin E_{DFA}$.

WTS $w$ rejected by $M_2$.

\[ w \neq <A> \text{ for any } DFA_A \text{ or } w = <A> \text{ but } L(SA) \neq \emptyset. \]

Case 1: $M_2$ rejects immediately.

Case 2: $M_2$ rejects in step 4.
Non-emptiness?

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

M₁ is decider. \( L(M₁) = A_{DFA} = \{<A, w> | w \in \Sigma^*\} \)

Is this problem decidable?

A. Yes, using M₃ in the handout.
B. Yes, using M₄ in the handout. \( \text{NOT A DECIDER} \)
C. Yes, both M₃ and M₄ work.
D. Yes, but not using the machines in the handout.
E. No.

\( L(M₄) = E'_{DFA} \)
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 \( \{ <A, B> \mid A, B \text{ are DFA over } \Sigma, L(A) = L(B) \} \) is decidable. **Idea:** give high-level description

**Step 1:** construction

Will we be able to simulate?

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.

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

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 \( L(D) = \) symmetric difference of \( L(A) \) 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 = \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.

Step 2: correctness proof
WTS \((1)\) $L(M_5) = EQ_{DFA}$ and \((2)\) $M_5$ is a decider.
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. $A_{\text{REX}} = A_{\text{NFA}} = A_{\text{DFA}}$, $E_{\text{REX}} = E_{\text{NFA}} = E_{\text{DFA}}$, $EQ_{\text{REX}} = EQ_{\text{NFA}} = EQ_{\text{DFA}}$

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
  - $A_{\text{DFA}}$
  - $E_{\text{DFA}}$
  - $EQ_{\text{DFA}}$

- **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.