Today's learning goals

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

Solutions to IndiviHWS on Piazza
Practice Qs for Exam 2 on website
Recognition and enumeration

1. Assume $L$ is Turing-recognizable. WTS some enumerator enumerates it.

Let $M$ be a TM that recognizes $L$. We'll use $M$ in a subroutine for high-level description of enumerator $E$.

Idea: check each string in turn to see if it is in $L = L(M)$.

Standard string ordering: order strings first by length, then dictionary order. (p. 14)
1. Assume $L$ is Turing-recognizable. WTS some enumerator enumerates it.

Let $M$ be a TM that recognizes $L$. We'll use $M$ in a subroutine for high-level description of enumerator $E$.

Let $s_1, s_2, \ldots$ be a list of all strings in $\Sigma^*$ in standard string order

$E = "$\text{Ignore any input. Repeat the following for } i=1,2,3\ldots$

1. Run $M$ for (at most) $i$ steps on each input $s_1, \ldots, s_i$
2. If any of the $i$ computations of $M$ accepts, print out the accepted string."

Correctness? \( \times \)
Example: union

Claim: The class of recognizable languages over fixed alphabet $\Sigma$ is closed under union.

Proof idea: Let $E_1$ and $E_2$ be enumerators. Construct the enumerator $E = "<ignore input>:

1. Run $E_1$ and $E_2$ in parallel
   i.e. for $i=1, 2, 3\ldots$
   1a. Run $E_1$ for $i$ steps.
   1b. Run $E_2$ for $i$ steps.

Whenever any string is printed by either enumerator, print it."
Suppose M is TM that recognizes L. Suppose D is TM that decides L. Suppose E is enumerator that enumerates L.

<table>
<thead>
<tr>
<th>If string w is in L then …</th>
<th>M accepts w</th>
<th>D accepts w</th>
<th>E prints w (in finite time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>If string w is not in L then …</td>
<td>M rejects w or M loops on w</td>
<td>D rejects w</td>
<td>E never prints w.</td>
</tr>
</tbody>
</table>
High-level description $\Rightarrow$ Algorithm

"On input $w$:
1. If $|w|$ even then reject
2. Otherwise, if $w$ is the binary representation of an even number, report."

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.
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) = \emptyset \}

- Is a Turing machine a decider? encoded by
  \{ representations of Turing machines M such that M always halts \}
Representations for computational problems

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

To define TM M:

"On input w ...

1. ...
2. ...
3. ...

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

Notation:

- \(<O>\) is the string that represents (encodes) the object \(O\)
- \(<O_1, ..., O_n>\) is the single string that represents the tuple of objects \(O_1, ..., O_n\)
To decide these problems, we need to represent the objects of interest as **strings**

To define TM $M$: 

"On input $w$ ...

1. ...
2. ...
3. ...

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

**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 $(Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej})$. 

*Sipser p. 185*
Encoding inputs

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

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

\[
\{ \text{w} | \text{w in \{0,1\}^* and w = w^R} \}\]

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

\[
\{ <\text{B},\text{w}> | \text{B is a DFA over } \Sigma, \text{w in } \Sigma^*, \text{and w is in } L(\text{B}) \}\]

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

\[
\{ <\text{A}> | \text{A is a PDA and } L(\text{A}) \text{ is infinite} \}\]
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_1 = "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_1 = "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 $M_1$ a decider?
Computational problems

Vocabulary

$A_{XX}$ "Is a given string accepted by a given machine of type $XX$?"
\[ \{ <B,w> \mid B \text{ is a } XX \text{ over } \Sigma, w \in \Sigma^*, \text{ and } w \text{ is in } L(B) \} \]

$E_{XX}$ "Is the language of a machine of type $XX$ is empty?"
\[ \{ <A> \mid A \text{ is a } XX \text{ over } \Sigma, L(A) \text{ is empty} \} \]

$EQ_{XX}$ "Are the languages of two given machines of type $XX$ equal?"
\[ \{ <A, B> \mid A \text{ and } B \text{ are } XX \text{ over } \Sigma, L(A) = L(B) \} \]
For DFA

A. \(<A, 1>\) is in \(A_{DFA}\)
B. \(<A, 1>\) is in \(E_{DFA}\)
C. \(<A>\) is in \(E_{DFA}\)
D. \(<A, B>\) is in \(EQ_{DFA}\)
E. More than one of the above

\(B\) is a DFA, \(1 \in L(A)\)

\(E_{A}\) = \{ \(<M> | M \text{ is DFA}\} \) Type error

\(L(A) \neq \emptyset\), \(L(B) = \emptyset\) so \(L(A) \neq L(B)\)

\(<B> \in E_{DFA} \land <B, B> \in EQ_{DFA}\)
For DFAs: which of the following computational problems are \textit{decidable}?

A. $A_{\text{DFA}}$
B. $E_{\text{DFA}}$
C. $\text{EQ}_{\text{DFA}}$
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 $\Sigma, L(A)$ 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?
Proving decidability

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

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

Test cases:

- e.g. $<$ is in $E_{\text{DFA}}$; $<$
- $>$ 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 \{ <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
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: $M_2 =$ "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.
For next time

**GroupHW5 due** Saturday, May 19

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