Lecture 13

Last time:
• Turing Machine Variants
• Church-Turing Thesis
• Universal Turing Machine
• Decidable languages

Today
• Decidable languages
• Designing deciders

Sofya Raskhodnikova
Recall

- We can encode DFAs, NFAs, regular expressions, PDAs, CFGs, etc into strings of 0s and 1s.
- We defined the following languages:

  \[ A_{\text{DFA}} = \{ \langle D, w \rangle \mid D \text{ is a DFA that accepts string } w \} \]

  \[ A_{\text{NFA}} = \{ \langle N, w \rangle \mid N \text{ is an NFA that accepts string } w \} \]

  \[ A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \]
Recall: Examples of decidable languages

\[ A_{\text{DFA}} = \{ \langle D, w \rangle \mid D \text{ is a DFA that accepts string } w \} \]

\[ A_{\text{NFA}} = \{ \langle N, w \rangle \mid N \text{ is an NFA that accepts string } w \} \]

\[ A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \} \]
Decidable languages: more examples

\[
\begin{align*}
\mathbf{E}_{\text{DFA}} &= \{ \langle D \rangle \mid D \text{ is a DFA that recognizes the empty language} \} \\
\mathbf{EQ}_{\text{DFA}} &= \{ \langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2) \} \\
\mathbf{E}_{\text{CFG}} &= \{ \langle G \rangle \mid G \text{ is a CFG that generates the empty language} \}
\end{align*}
\]
Theorem. $E_{DFA}$ is decidable.

$E_{DFA} = \{ \langle D \rangle | D$ is a DFA that recognizes $\emptyset. \}$

Proof: The following TM $M$ decides $E_{DFA}$.  
$M = \text{``On input } \langle D \rangle, \text{ where } D \text{ is a DFA:} \$

1. Use BFS to determine if an accepting state of $D$ is reachable from from its start state.
2. Accept if not. O.w. reject.”
**Theorem.** $\text{EQ}_{\text{DFA}}$ is decidable.

$$\text{EQ}_{\text{DFA}} = \{ \langle D_1, D_2 \rangle | \ D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2) \}$$

**Proof:** The following TM $M$ decides $\text{EQ}_{\text{DFA}}$.

$M = \text{``On input } \langle D_1, D_2 \rangle, \text{ where } D_1, D_2 \text{ are DFAs:}$$

1. Construct a DFA $D$ that recognizes the set difference of $L(D_1)$ and $L(D_2)$.
2. Run the decider for $E_{\text{DFA}}$ on $<D>$.
3. If it accepts, accept. O.w. reject."
Theorem. \( E_{\text{CFG}} \) is decidable.

\[
E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG that generates no strings} \}
\]

Proof: The following TM \( M \) decides \( E_{\text{CFG}} \).

\[ M = \"\text{On input } \langle G \rangle, \text{ where } G \text{ is a CFG:} \]

1. Mark all terminals in \( G \).
2. Repeat until no new variable is marked:
3. Mark any variable \( A \) where \( G \) has a rule \( A \rightarrow \cdots \) and each variable/terminal on the RHS is already marked.
4. Accept if the start variable is unmarked. O.w. reject."
Prove that the following language is decidable:

\[ R_{DFA} = \{ \langle D, w \rangle \mid D \text{ is a DFA that rejects string } w \} \]

Formulate the following problem as a language and prove that it is decidable:

Given a PDA and a string, determine if the PDA accepts the string.

\[ A_{PDA} = \{ \langle P, w \rangle \mid P \text{ is a PDA that accepts string } w \} \]

Can a TM just simulate P on w, accept if it accepts and reject o.w.?
Exercise

A decider for $A_{PDA}$ can, on input $<P, w>$

A. simulate $P$ on $w$, accept if it accepts and reject o.w.
B. convert $P$ to an equivalent CFG $G$ and then run a decider for $A_{CFG}$, accept if it accepts and reject o.w.
C. convert $P$ to an equivalent CFG $G$ and then run a decider for $A_{CFG}$, accept if it rejects and reject o.w.
D. None of the above.
E. More than one choice above works.
Examples of decidable languages so far

\[ \text{A}_{\text{DFA}} = \{ \langle D, w \rangle \mid D \text{ is a DFA that accepts string } w \} \]
\[ \text{E}_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA and } L(D) = \emptyset \} \]
\[ \text{EQ}_{\text{DFA}} = \{ \langle D_1, D_2 \rangle \mid D_1, D_2 \text{ DFAs and } L(D_1) = L(D_2) \} \]

\text{A}_{\text{DFA}}, \text{E}_{\text{DFA}}, \text{EQ}_{\text{DFA}}, \text{A}_{\text{CFG}}, \text{E}_{\text{CFG}} \text{ are decidable.}
Classes of languages

recognizable

decidable

CFL

regular
Theorem. Every CFL is decidable.

Proof: Let $G$ be a CFG for $L$. Design a TM $M_G$ that decides $L$. 
- Is it a good idea to convert $G$ to an equivalent PDA $P$ and have $M_G$ simulate $P$?
Exercise

G is a CFG for L. Design a TM $M_G$ that decides L.

Is it a good idea to convert G to an equivalent PDA P and have $M_G$ simulate P?

A. Yes. Why not?
B. No, we can’t always convert G to an equivalent PDA.
C. No, P might loop on some inputs.
D. No, because we don’t have any input to run P on.
E. None of the above.
G is a CFG for L. Design a TM $M_G$ that decides L. A decider for which language is useful as a subroutine?

A. for $A_{DFA}$
B. for $E_{DFA}$
C. for $EQ_{DFA}$
D. for $A_{CFG}$
E. for $E_{CFG}$
Theorem. Every CFL is decidable.

Proof: Let G be a CFG for L. Design a TM $M_G$ that decides L.

- Is it a good idea to convert G to an equivalent PDA P and have $M_G$ simulate P?

$M =$ ``On input $w$:"

1. Run the decider for $A_{CFG}$ on input $<G,w>$.
2. Accept if it accepts. O.w. reject."
Classes of languages

recognizable

decidable

CFL

regular
Theorem. \( \text{INFINITE}_{\text{DFA}} \) is decidable.

\( \text{INFINITE}_{\text{DFA}} = \{ \langle D \rangle | \text{D is a DFA and } L(D) \text{ is infinite}\} \)

Idea: Let \( n \) be the number of states in \( D \). 
L(D) is infinite iff \( D \) accepts a string of length \( \geq n \).

Proof: The following TM \( M \) decides \( \text{INFINITE}_{\text{DFA}} \).

\( M = \text{``On input } \langle D \rangle, \text{ where } D \text{ is a DFA:} \)

1. Let \( n \) be the number of states in \( D \).
2. Let \( C \) be a DFA for \( \{ w \mid |w| \geq n \} \).
3. Build a DFA \( B \) for \( L(C) \cap L(D) \).
4. Run a decider for \( E_{\text{DFA}} \) on \( \langle B \rangle \).
5. Accept if it rejects. O.w. reject.”
Theorem. $\text{PAL}_{\text{DFA}}$ is decidable.

- Formulate the following problem as a language and prove that it is decidable:
  
  Given a DFA, determine if it accepts some palindrome.
## Problems in language theory

<table>
<thead>
<tr>
<th>$A_{\text{DFA}}$</th>
<th>$A_{\text{CFG}}$</th>
<th>$A_{\text{TM}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>decidable</td>
<td>decidable</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$E_{\text{DFA}}$</th>
<th>$E_{\text{CFG}}$</th>
<th>$E_{\text{TM}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>decidable</td>
<td>decidable</td>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$EQ_{\text{DFA}}$</th>
<th>$EQ_{\text{CFG}}$</th>
<th>$EQ_{\text{TM}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>decidable</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>