Today's learning goals

Sipser Ch 5.1

• Define and explain core examples of computational problems, include $A^{**}$, $E^{**}$, $EQ^{**}$, $HALT_{TM}$ (for ** either DFA or TM)
• Explain what it means for one problem to reduce to another
• Use reductions to prove undecidability (or decidability)
<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$\overline{A}_{TM}$</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>$HALT_{TM}$</td>
</tr>
<tr>
<td>$E_{TM}$</td>
<td></td>
</tr>
</tbody>
</table>

"Does this TM halt on input string that is given?"

"Is the lang of some DFA empty?"

Computational Problem about TM

$\{ <M> \mid M \text{ is Turing machine} \}$
Using reduction to prove undecidability

Claim: Problem X is undecidable

Proof strategy: Show that $A_{\text{TM}}$ reduces to X.
Alternate Proof strategy: Show that $\text{HALT}_{\text{TM}}$ reduces to X.
Alternate Proof strategy: Show that $E_{\text{TM}}$ reduces to X.

In each of these, have access to Genie which can answer questions about X.
Reduction does not mean reduction

Caution: Section 5.2, 5.3 won't be covered in CSE 105.

Mapping reducibility from Section 5.3 is *different* from the reductions we see in Section 5.1.

The results from 5.3 do not necessarily carry over to the reductions from 5.1.
\( EQ_{TM} = \{ <M_1, M_2> \mid M_1, M_2 \text{ TMs, } L(M_1) = L(M_2) \} \)

A. Decidable
B. Undecidable
C. No way to tell

Give an example of a string in \( EQ_{TM} \), and a string not in \( EQ_{TM} \).
Claim: ?? is no harder than \( \text{EQ}_{\text{TM}} \)

Given: Turing machine \( M \), string \( w \), magic genie for \( \text{EQ}_{\text{TM}} \)

\( \text{ATM}, \ \text{ETM}, \ \text{HALT}_{\text{TM}}, \ \text{Aim} \)

Goal: ??
Building machines

In reduction proofs, we often need to build two different machines:

1. machine to decide problem
2. auxiliary machine to ask Genie about encoded information
$E_{TM}$ reduces to $EQ_{TM}$

For machine that decides $E_{TM}$, what is input?

A. M
B. w
C. $<M,w>$
D. $<M>$
E. None of the above.
$E_{TM}$ reduces to $EQ_{TM}$

"On input $<M>$

1. Build $TM$ $X$ "On input $x$: 1. reject"

2. Ask $G_{EQ}$ if $L(M) = L(X)$

   i.e. $<M, X> \in EQ_{TM}$

3. Accept if $G_{EQ}$ says yes
   Reject if $G_{EQ}$ says no

Proof of correctness

WTS 1. $<M > \notin E_{TM} \Rightarrow$ our alg accepts
   2. $<M > \in E_{TM} \Rightarrow$ our alg rejects
HALT\textsubscript{TM} reduces to EQ\textsubscript{TM}

- Input: \(<M,w>\)

- Goal: Accept if \(M\) halts on input \(w\), Reject if \(M\) loops on input \(w\)

Auxiliary machine goal: build \(X\) based on \(M,w\) such that \(L(X) = \Sigma^*\) if \(M\) halts on \(w\), and \(L(X) \neq \Sigma^*\) if \(M\) loops on \(w\).
HALT \text{ reduces to } EQ \rightarrow

"On input \langle M, w \rangle"

1. Build $X = "\text{On input } x. 1. \text{ Run } M \text{ on } w \text{. } x "$

2. Build $\tilde{X} = "\text{On input } x. 1. \text{ accept } x"$

3. Ask $\text{Geq}$ if $L(X) = L(\tilde{X})$

4. If $\text{Geq}$ says yes, accept
   if $\text{Geq}$ says no, reject"
## Recap

### Decidable vs Undecidable

<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td>$A_{TM}^c$</td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td>$HALT_{TM}$</td>
</tr>
<tr>
<td>$E_{TM}$</td>
<td>$not\ rec$</td>
</tr>
<tr>
<td>$EQ_{TM}$</td>
<td>$not\ rec$</td>
</tr>
</tbody>
</table>

(Practice $E_{TM}$ reduces to $HALT_{TM}$)

Which are recognizable?
Why care about Genies?

Reductions are central in
- (un)computability theory
- complexity theory
- cryptography

Central idea: how do we convert information about one problem to information about another?
Next time

Pre-class reading skim Chapter 7