CSE 105
THEORY OF COMPUTATION

"Winter" 2018

http://cseweb.ucsd.edu/classes/wi18/cse105-ab/

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)

Decidable	Undecidable
A_{DFA}	A_{TM}
E_{DFA}	$\overline{A_{TM}}^{C}$
EQ_{DFA}	$HALT_{TM}$
	E_{TM}

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

"Is the lang of some DFA empty?"

Computational Problem about TM

$\{ \langle M \rangle \mid$ M is Turing machine and - - - - - $\}$

Using reduction to prove undecidability

Claim: Problem X is undecidable

Proof strategy: Show that A_{TM} reduces to X.

Alternate Proof strategy: Show that $HALT_{TM}$ reduces to X.

Alternate Proof strategy: Show that E_{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.

$EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1, M_2 \text{ TMs}, L(M_1)=L(M_2)\}$

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 EQ_{TM}

A_{TM}, *E_{TM}, $HALT_{TM}$, $\overline{A_{TM}}$

Given: Turing machine M, string w, $\boxed{\text{magic genie for } EQ_{TM}}$

E_{TM}: Given Turing machine M

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

encode 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 $\langle M \rangle$

1. Build TM X "On input x. 1. reject"
2. Ask G_{EQ} if $L(M) = L(X)$ i.e. $\langle M, X \rangle \in EQ_{TM}$
3. Accept if <u>G_{EQ} says yes</u>
   Reject if <u>G_{EQ} says no</u> "

Pf of correctness

WTS ① $\langle M \rangle \in E_{TM} \Longrightarrow$ our alg accepts ② $\langle M \rangle \notin E_{TM} \Rightarrow$ our alg rejects

$HALT_{TM}$ reduces to EQ_{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.

$HALT_{TM}$ reduces to EQ_{TM}

"On input $\langle M, w \rangle$

1. Build X = "On input x 1. Run M on w
2. If M halts on w, accept"

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

3. Ask G_{EQ} if $L(X) = L(\tilde{X})$

4. If G_{EQ} says yes, accept
if G_{EQ} says no, reject"

Recap

Decidable	Undecidable
A_{DFA}	A_{TM} rec
E_{DFA}	$A_{TM}{}^{C}$ not rec
EQ_{DFA}	$HALT_{TM}$ rec
	E_{TM} not rec
	EQ_{TM}

(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