CSE 105
THEORY OF COMPUTATION

"Winter" 2018

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

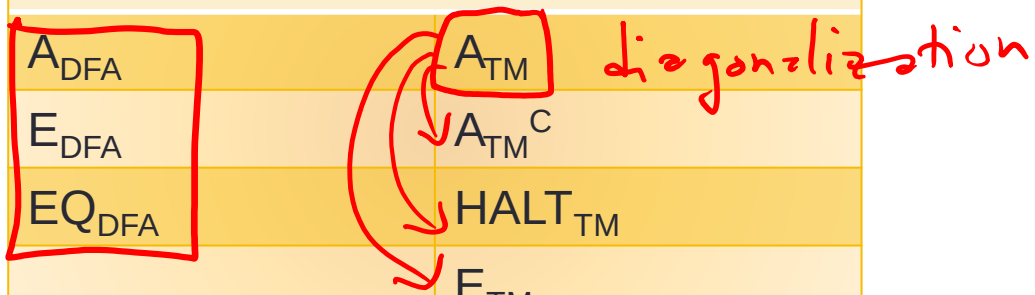
Today's learning goals

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)

- Complexity.

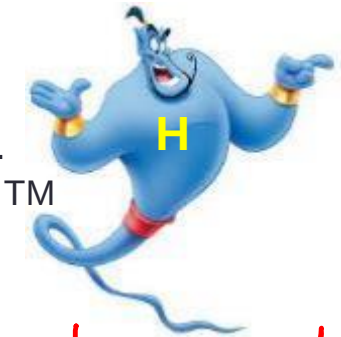
Decidable	Undecidable
A_{DFA}	A_{TM}
E_{DFA}	A_{TM}^{C}
EQ_{DFA}	$HALT_{TM}$
	E_{TM}



Claim: E_{TM} is no harder than $HALT_{TM}$

In other words: we could use $HALT_{TM}$ to solve E_{TM}

Given: Turing machine M, ~~string w~~, magic genie for $HALT_{TM}$



does 2 machine halt on > given input? Yes or No

"On input <M>

1. Build a new Turing machine X:
   X = "On input x, _____"
2. Ask Genie about X, ⟨ε⟩
3. If Genie says no, then _____; if Genie says yes, then _____.

accept

reject.

X does not halt ⇒ L(M) = ∅

X halts ⇒ L(M) ≠ ∅

Goal: Accept if L(M) is empty; Reject if L(M) is nonempty?

X:
On input x
for i = 1, 2, 3, ...:
Run M on s_i;
if M accepts then accept
if M rejects then continue.

Not good enough

X:
on input x
for i = 1 2 3 . . .
run M on s_1 i times
run M on s_2 i times
run M on s_i i times

→ if M ever accepts then halt and accept.

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}

$\langle M, M \rangle$

{ Turing machine that accepts on all inputs , Turing Machine that rejects on all inputs

Claim: ?? is no harder than EQ_{TM}

Given: Turing machine M, string w, magic genie for EQ_{TM}

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

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}

X:
on input x
reject.

$L(X) = \emptyset$

we have access to a genie that can answer the question.

does $L(M_1) = L(M_2)$ for two turing machines M_1, M_2. Yes or No

Goal: Given a machine $\langle M \rangle$.
accept if $L(M) = \emptyset$
reject if $L(M) \neq \emptyset$

Ask Genie about $\langle X, M \rangle$. if genie says yes then accept($L(M) = \emptyset$)
says no then reject($L(M) \neq \emptyset$)

$HALT_{TM}$ 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.

X_1: on input x

Run M on w.

if M accepts w then accept.

if M rejects w then accept.

---

if M halts on w then $L(X_1) = \Sigma^*$

if M does not halt on w then $L(X_1) = \emptyset$

X_2: on input x $\quad L(X_2) = \Sigma^*$

accept.

Ask Genie is $L(X_1) \simeq L(X_2)$?

Yes: M halts on w so accept

No: M does not halt on w so reject.

Recap

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

recognizable. (by construction)

No recognizable. (by construction)

is not recognizable

P is decidable $\Leftrightarrow$ P and P^C are recognizable.

Which are recognizable?

E_{TM} is un-recognizable

E_{TM} is undecidable

$\overline{E_{TM}}$ is recognizable.

$E_{TM} \leq \{\langle M \rangle \mid L(M) \neq \emptyset\}$

X: on input $\langle M \rangle$

for i = 1, 2, 3, ...

Run M on s_1 i transitions

Run M on s_2 i transitions

⋮

Run M on s_i i transitions

if M ever accepts then X accepts.

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?

Complexity theory

Chapter 7

In the "real world", computers (and Turing machines) don't have infinite tape, and we can't afford to wait unboundedly long for an answer.

"Decidable" isn't good enough – we want "Efficiently decidable"

Not just decidable …

- For a given **algorithm** working on a given **input**, how long do we need to wait for an answer? How does the running time depend on the input in the worst-case? average-case? Expect to have to spend more time on larger inputs.

Measuring time

- For a given **algorithm** working on a given **input**, how long do we need to wait for an answer? Count steps! How does the running time depend on the input in the worst-case? average-case? Big-O

Can we detect problems that are **efficiently solvable**?

Time complexity

For M a deterministic decider,

its **running time** or **time complexity** is the function

f: N → R^+ given by

f(n) = maximum number of steps M takes before halting,
over all inputs of length n.

worst-case analysis

Plus, instead of calculating precisely, **estimate** f(n) by using big-O notation.

Time complexity classes

TIME(t(n)) = { L | L is decidable by a TM running in O(t(n)) }

- **Exponential** $EXPTIME = \bigcup_k TIME(2^{n^k})$

Brute-force search

- **Polynomial** $P = \bigcup_k TIME(n^k)$

Invariant under many models of TMs

- **Logarithmic**

May not need to read all of input

$$P = \bigcup_k TIME(n^k)$$

Why is it okay to group all polynomial running times?

- Contains all the "feasibly solvable" problems.
- Invariant for all the "usual" deterministic TM models
  - multitape machines (Theorem 7.8)
  - multi-write

Working with P

- Problems encoded by languages of strings
  - Need to make sure coding/decoding of objects can be done in polynomial time.
- Algorithms can be described in high-level or implementation level

CAUTION: not allowed to guess / make non-deterministic moves.

Next time

Pre-class reading skim Chapter 7