CSE 105
THEORY OF COMPUTATION

Spring 2018

Discussion today: Ch 4 + 5
Group HW 6 due Saturday
Review Quiz due Sunday
* Group HW 7 due Tuesday*
Optional HW 8 - Discussion wk 10

http://cseweb.ucsd.edu/classes/sp18/cse105-ab/

Review session Wednesday evening.
Today's learning goals

- 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
- Define computable functions, and use them to give mapping reductions between computational problems.
<table>
<thead>
<tr>
<th>Decidable</th>
<th>Undecidable but recognizable</th>
<th>Undecidable and unrecognizable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_{DFA}$</td>
<td>$A_{TM}$</td>
<td>$A_{TM}^C$</td>
</tr>
<tr>
<td>$E_{DFA}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$EQ_{DFA}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Give algorithm!

Diagonalization
Idea

If problem X is no harder than problem Y
  ...and if Y is **decidable**
  ...then X must also be **decidable**

If problem X is no harder than problem Y
  ...and if X is **undecidable**
  ...then Y must also be **undecidable**

“Problem X is no harder than problem Y” means
“Can convert questions about membership in X to questions about membership in Y”
Problem A is mapping reducible to problem B means there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $x$ in $\Sigma^*$

$$x \text{ is in } A \iff f(x) \text{ is in } B$$

**Computable function?**

A function $f: \Sigma^* \rightarrow \Sigma^*$ is computable iff there is some Turing machine such that, for each $x$, on input $x$ halts with exactly $f(x)$ followed by all blanks on the tape
Computable functions (aka maps)

Which of the following functions are computable?

A. The string x maps to the string xx.
B. The string <M> (where M is a TM) maps to <M'> where M' is the Turing machine that acts like M does, except that if M tries to reject, M' goes into a loop; strings that are not the codes of TMs map to ε.
C. The string x maps to y, where x is the binary representation of the number n and y is the binary representation of the number 2^n
D. All of the above.
E. None of the above.
The halting problem!

\[ \text{HALT}_{TM} = \{ <M,w> | M \text{ is a TM and } M \text{ halts on input } w \} \]

\[ A_{TM} = \{ <M,w> | M \text{ is a TM and } w \text{ is in } L(M) \} \]

How is \( \text{HALT}_{TM} \) related to \( A_{TM} \)?

A. They're the same set.

B. \( \text{HALT}_{TM} \) is a subset of \( A_{TM} \)

C. \( A_{TM} \) is a subset of \( \text{HALT}_{TM} \)

D. They have the same type of elements but no other relation.

E. I don't know.
The halting problem!

\[ \text{HALT}_{TM} = \{ <M,w> \mid M \text{ is a TM and } M \text{ halts on input } w \} \]

\[ A_{TM} = \{ <M,w> \mid M \text{ is a TM and } w \text{ is in } L(M) \} \]

But subset inclusion doesn't determine difficulty!

What about mapping reduction?
The halting problem!

$\text{HALT}_{\text{TM}} = \{ <M,w> | M \text{ is a TM and } M \text{ halts on input } w \}$

$A_{\text{TM}} = \{ <M,w> | M \text{ is a TM and } w \text{ is in } L(M) \}$

Goal: build function $f: \Sigma^* \rightarrow \Sigma^*$ such that for every string $x$, $x$ is in $A_{\text{TM}}$ iff $f(x)$ is in $\text{HALT}_{\text{TM}}$

i.e. $A_{\text{TM}}$ reduces to $\text{HALT}_{\text{TM}}$
Reducing $A_{TM}$ to $HALT_{TM}$  

Desired function by cases:

- If $x = <M, w>$ and $w$ is in $L(M)$: map to $<M', w'>$ in $HALT_{TM}$
- If $x = <M, w>$ and $w$ is not in $L(M)$: map to $<M', w'>$ not in $HALT_{TM}$
- If $x \neq <M, w>$: map to some string not in $HALT_{TM}$
Reducing $A_{TM}$ to $HALT_{TM}$  

Sipser Example 5.24

Desired function by cases:

- If $x = <M,w>$ and $w$ is in $L(M)$: map to $<M’, w’>$ in $HALT_{TM}$
- If $x = <M,w>$ and $w$ is not in $L(M)$: map to $<M’, w’>$ not in $HALT_{TM}$
- If $x \neq <M,w>$: map to some string not in $HALT_{TM}$

Pick some specific string constant not in $HALT_{TM}$
Reducing $A_{TM}$ to $HALT_{TM}$  

Sipser Example 5.24

Define *computable* function:

$F = \text{"On input } x:"
1. Type-check whether $x = <M,w>$ for some TM $M$, and string $w$. If not, output const out.
2. ...
3. ...
4. ....  

$F$ is defined by high-level description of TM: each step must be algorithmic!
Reducing $A_{TM}$ to $HALT_{TM}$  

Sipser Example 5.24

Define **computable** function:

$F =$ “On input $x$:
1. Type-check whether $x = <M,w>$ for some TM $M$, and string $w$. If not, output constant.
2. Simulate $M$ on $w$.
3. If accepts, accept. If rejects, reject.
4. .... ”

F is defined by high-level description of TM: each step must be algorithmic!
Reducing $A_{TM}$ to $HALT_{TM}$

Define **computable** function:

$F = \text{"On input x:}\
\begin{enumerate}
\item Type-check whether $x = <M, w>$ for some TM $M$, and string $w$. If not, output const\text{\textsubscript{out}}.
\item Construct the following machine $M'$\text{\textsuperscript{2}}
$M' = \text{"On input x:}\
\begin{enumerate}
\item Run $M$ on $x$.
\item If $M$ accepts, accept.
\item If $M$ rejects, enter a loop.
\end{enumerate}
\item Output $<M', w>$.\text{\textsuperscript{4}}
\end{enumerate}$

$F$ is defined by high-level description of TM: each step must be algorithmic!

$L(M) = L(M')$ but $M$ may accept/reject/loop on input, $M'$ may only accept/loop on input.

\textsuperscript{2} $M'$ is similar to $M$ but different.

\textsuperscript{4} Output $<M', w>$ is distinct from the definition of $F$.
Reducing $A_{TM}$ to $HALT_{TM}$  

Sipser Example 5.24

Check how function behaves by cases:

- If $x = <M,w>$ and $w$ is in $L(M)$: map to $<M', w'>$ in $HALT_{TM}$?
  - so $M'$ simulates $M$ on $w$ and since $M$ accepts $w$, $M'$ will too

- If $x = <M,w>$ and $w$ is not in $L(M)$: map to $<M', w'>$ not in $HALT_{TM}$?
  - so $M'$ simulates $M$ on $w$, loops, rejects or $M'$ loops on $w$
  - so $F(x) = <M',w> \not\in HALT_{TM}$

- If $x \neq <M,w>$: map to some string not in $HALT_{TM}$?

i.e. $X \in A_{TM}$ if $F(x) \not\in HALT_{TM}$,
Other direction?

Goal: build function that $f: \Sigma^* \to \Sigma^*$ such that for every string $x$,

$$x \text{ is in } \text{HALT}_\text{TM} \iff f(x) \text{ is in } A_{\text{TM}}$$

What function should be used for $f(x)$ in the reduction?

A. Use the function $F$ from previous reduction
B. Use the inverse of the function $F$ from previous reduction
C. Use a different computable function
D. Impossible to find a computable function that works!
$\text{HALT}_T^\text{m}$ mapping reduces to $\text{ATM}$?

Need $f : \Sigma^* \rightarrow \Sigma^*$ s.t.

$$x \in \text{HALT}_T^\text{m} \iff f(x) \in \text{ATM}$$

if computable

Given $ct \notin \text{ATM}$.

$f = \text{``On input } \Sigma^*, \text{ do the following:}$$

1. Typecheck to see if $\sigma = \langle M, w \rangle$ is a string of $\Sigma^*$.
2. If not, output $ct$. Otherwise, $x = \langle M, w \rangle$.

2. Build $M' = \text{``On input } y$$
1. Run $M$ on $y$
2. If $M$ accepts, accept.$$

Output $\langle M', w \rangle$.
Next time

Pre-class reading Example 5.24, Theorems 5.22