CSE 105
THEORY OF COMPUTATION

Spring 2018

Discussion today: Ch 4 + 5
Group HW6 due Saturday
Review Quiz due Sunday
* Group HW7 due Tuesday
Optional HW8 - 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
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”
Mapping reduction

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 \( \epsilon \).
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!

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

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

How is \text{HALT}_{TM} related to \text{A}_{TM}?

A. They're the same set.
B. \text{HALT}_{TM} is a subset of \text{A}_{TM}
C. \text{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!
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 \( f: \Sigma^* \to \Sigma^* \) such that for every string \( x \),
\[ x \text{ is in } A_{\text{TM}} \text{ iff } f(x) \text{ is in } \text{HALT}_{\text{TM}} \]

i.e. \( A_{\text{TM}} \) reduces to \( \text{HALT}_{\text{TM}} \)
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 $HALT_{TM}$
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 = “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:

$$F: \Sigma^* \rightarrow \{\#, \square\}$$

$F = \text{“On input } x:\text{ “}$

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 $HALT_{TM}$  

Sipser Example 5.24

Define **computable** function:

$F = \text{"On input } x:\text{ do }$ 

1. Type-check whether $x = <M, w>$ for some TM $M$, and string $w$. If not, output $\text{const}_{\text{out}}$. 
2. Construct the following machine $M'$ 

   $M' = \text{"On input } x:\text{ do }$ 
   
   1. Run $M$ on $x$. 
   2. If $M$ accepts, accept. 
   3. If $M$ rejects, enter a loop. 
3. Output $<M', w>$ 

$\langle M' \rangle = \langle \langle M \rangle \rangle$  

i.e. potentially $x \in A_{TM}$ 

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

Example 5.24

Check how function behaves by cases:

- If $x = \langle M, w \rangle$ and $w$ is in $L(M)$: map to $\langle M', w' \rangle$ in $HALT_{TM}$?
  - $M$ accepts $w$.
  - $M'$ loops on $w$ for an infinite number of steps.
  - So, $\langle M', w' \rangle$ is not in $HALT_{TM}$.

- If $x = \langle M, w \rangle$ and $w$ is not in $L(M)$: map to $\langle M', w' \rangle$ not in $HALT_{TM}$?
  - $M$ rejects $w$, by construction $M'$ loops on $w$.
  - $\langle M', w' \rangle$ is not in $HALT_{TM}$.

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

i.e. $x \in A_{TM}$ if $f(x) \in HALT_{TM}$.
Other direction?

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

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

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!
Goal: Computable \( F_2 : \Sigma^* \rightarrow \Sigma^* \)

\( X \in \text{HALT}_T \) if \( F_2(x) \in \text{ATM} \)

\[
\begin{cases}
X = <M, w> \\
\text{where } M \text{ halts on } w
\end{cases}
\begin{cases}
X = <M, w> \text{ where } M \text{ loops on } w
\end{cases}
\]  

output \( c_t \), \((c_t \notin \text{ATM})\)

\( F_2 = "\text{On input } x, \)"

1. Typecheck; if \( x \neq <M, w> \), output \( c_t \).
2. 6w: \( x = <M, w> \)
3. Build \( M' = "\text{On input } y, \)"
4. Output \( <M', w> "\)"
Next time

Pre-class reading Example 5.24, Theorems 5.22