

CSE 105

THEORY OF COMPUTATION

Spring 2018

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

Today's learning goals

Sipser Ch 5.1, 5.3

- 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.

Decidable	Undecidable but recognizable	Undecidable and unrecognizable
A_{DFA}	A_{TM}	A_{TM}^c
E_{DFA}		
EQ_{DFA}		

Give algorithm!

Diagonalization

Idea

Sipser pp. 215-216

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

Sipser p. 235

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 Σ^*

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

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 $\langle M \rangle$ (where M is a TM) maps to $\langle M' \rangle$ 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!

$\text{HALT}_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \}$

$A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

How is HALT_{TM} related to A_{TM} ?

- A. They're the same set.
- B. HALT_{TM} is a subset of A_{TM}
- C. A_{TM} is a subset of HALT_{TM}
- D. They have the same type of elements but no other relation.
- E. I don't know.

The halting problem!

$\text{HALT}_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \}$

$A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

But subset inclusion doesn't determine difficulty!

The halting problem!

$\text{HALT}_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \}$

$A_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } w \text{ is in } L(M) \}$

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

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

Reducing A_{TM} to $HALT_{TM}$ Sipser Example 5.24

Desired function by cases:

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

Reducing A_{TM} to $HALT_{TM}$ Sipser Example 5.24

Desired function by cases:

- If $x = \langle M, w \rangle$ and w is in $L(M)$: map to $\langle M', w' \rangle$ in $HALT_{TM}$
- If $x = \langle M, w \rangle$ and w is not in $L(M)$: map to $\langle M', w' \rangle$ not in $HALT_{TM}$
- If $x \neq \langle M, w \rangle$: map to some string not in $HALT_{TM}$
Pick some specific string constant not in $HALT_{TM}$

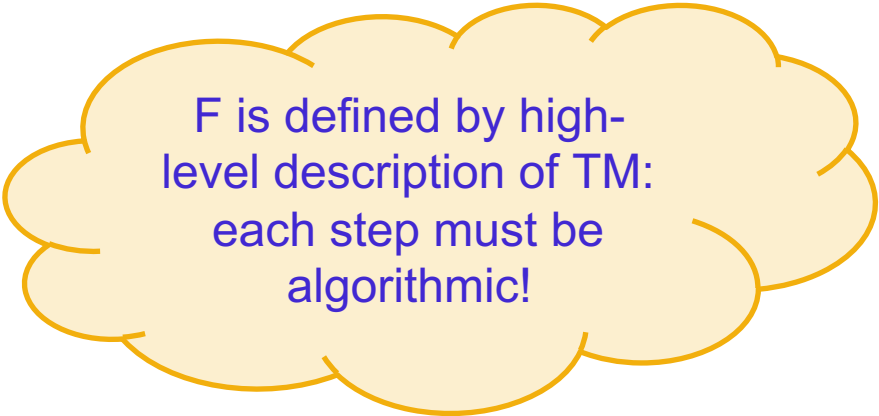
Reducing A_{TM} to $HALT_{TM}$

Sipser Example 5.24

Define **computable** function:

F = “On input x:

1. Type-check whether $x = \langle M, w \rangle$ 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!

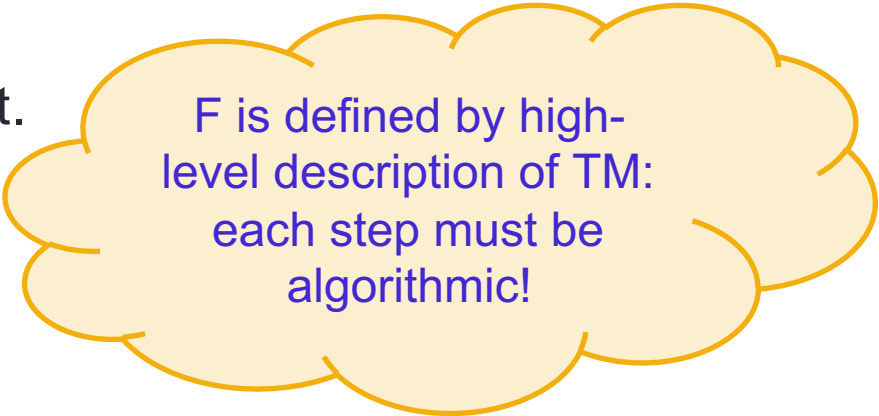
Reducing A_{TM} to $HALT_{TM}$

Sipser Example 5.24

Define **computable** function:

F = “On input x:

1. Type-check whether $x = \langle M, w \rangle$ for some TM M, and string w. If not, output $const_{out}$.
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!

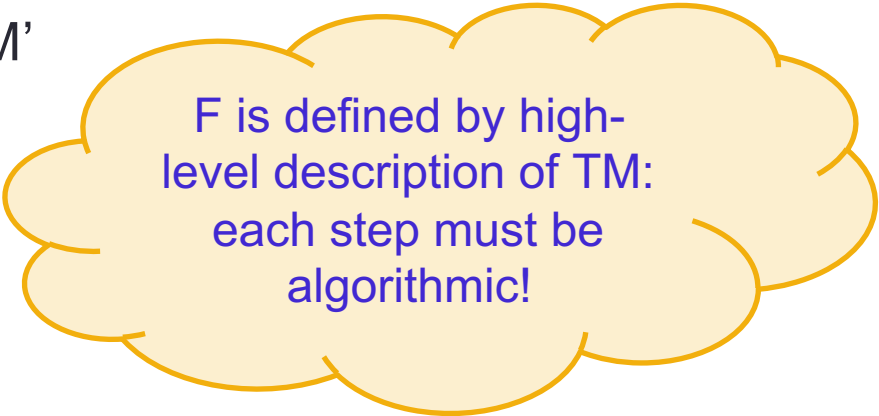
Reducing A_{TM} to $HALT_{TM}$

Sipser Example 5.24

Define **computable** function:

F = “On input x:

1. Type-check whether $x = \langle M, w \rangle$ for some TM M , and string w . If not, output $const_{out}$.
2. Construct the following machine M'
 M' = “On input x:
 1. Run M on x .
 2. If M accepts, accept.
 3. If M rejects, enter a loop.”
3. Output $\langle M', w \rangle$ ”



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

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

Other direction?

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

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

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!

Next time

Pre-class reading Example 5.24, Theorems 5.22