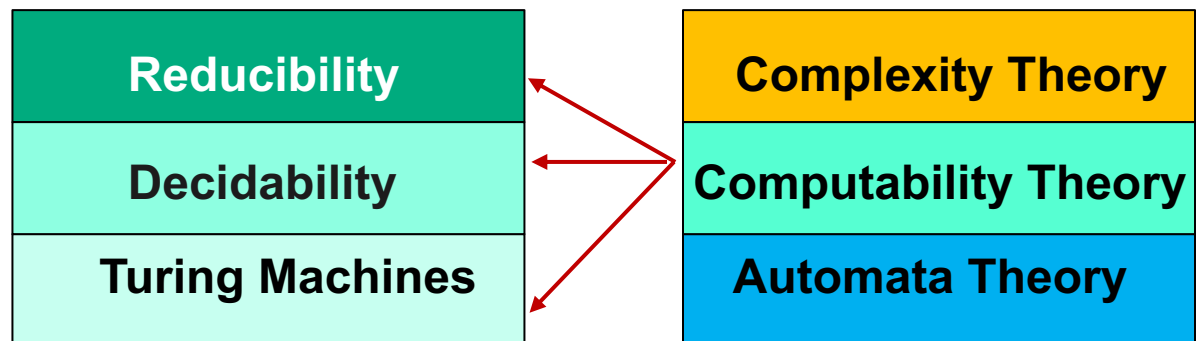




# Reducibility, part 2

---





# Agenda

---

- Proceed with discussion of reducibility
- Hints on HW 7 (last 10 min)

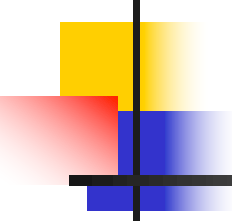


# Warm-up question (4/20/22)

---

What is the shorthand (or nickname) you use to refer to this course?

Send me your response by Canvas email.



In lectures from last week and before  
(chapter 4), we saw...

---

Decidable:

$A_{\text{DFA}}$  (acceptance)  
 $A_{\text{NFA}}$   
 $A_{\text{REG}}$   
 $E_{\text{DFA}}$  (emptiness)  
 $EQ_{\text{DFA}}$  (equivalence)  
 $A_{\text{CFG}}$   
 $E_{\text{CFG}}$   
Every CFG

Undecidable:

$A_{\text{TM}}$



## This week (chapter 5) we are...

---

- Examining several **additional** (besides  $A_{TM}$ ) **unsolvable problems**
- Learning about the primary method for proving that problems are computationally unsolvable – that method is called **reducibility**
- Examples we saw in last lecture (4/18)
  - Ex 1: Halting Problem
  - Ex 2:  $E_{TM}$
  - Ex 3:  $REGULAR_{TM}$



# Comment

---

- Using similar proof as in  $\text{REGULAR}_{\text{TM}}$ , the problems of testing whether the language of a TM is one of the following is undecidable:
  - Context-free language
  - Decidable language
  - Finite language
- **Rice's theorem** generalizes this
  - Determining *any property* of the languages recognized by Turing machines is undecidable



## Example 4: $EQ_{TM}$

---

**Let:**

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

**Theorem:**

$EQ_{TM}$  is undecidable



# Proof idea

---

- Show that if  $EQ_{TM}$  were decidable,  $E_{TM}$  also would be decidable by giving a reduction from  $E_{TM}$  to  $EQ_{TM}$
- The idea is simple.  $E_{TM}$  is the problem of determining whether the language of a TM is empty.  $EQ_{TM}$  is the problem of determining whether the languages of two TMs are the same.
- If one of these languages happens to be empty, we end up with the problem of determining whether the language of the other language is empty— that is, the  $E_{TM}$  problem.
- So in a sense, the  $E_{TM}$  problem is a special case of the  $EQ_{TM}$  problem wherein one of the machines is fixed to recognize the empty language.





# Proof

---

We let TM  $R$  decide  $EQ_{TM}$  and construct TM  $S$  to decide  $E_{TM}$  as follows

$S =$  “On input  $\langle M \rangle$ , where  $M$  is a TM:

1. Run  $R$  on input  $\langle M, M_1 \rangle$ , where  $M_1$  is a TM that rejects all inputs.
2. If  $R$  accepts, *accept*; if  $R$  rejects, *reject*.”

If  $R$  decides  $EQ_{TM}$ ,  $S$  decides  $E_{TM}$ . But  $E_{TM}$  is undecidable, so  $EQ_{TM}$  also must be undecidable



# Another look at this list (thanks to our growing body of knowledge)

---

## Decidable:

$A_{\text{DFA}}$  (acceptance)  
 $A_{\text{NFA}}$   
 $A_{\text{REG}}$   
 $E_{\text{DFA}}$  (emptiness)  
 $EQ_{\text{DFA}}$  (equivalence)  
 $A_{\text{CFG}}$   
 $E_{\text{CFG}}$   
 $EQ_{\text{CFG}}$

## Undecidable:

$A_{\text{TM}}$   
 $\text{HALT}_{\text{TM}}$   
 $E_{\text{TM}}$   
 $EQ_{\text{TM}}$   
 $\text{REGULAR}_{\text{TM}}$   
 $\text{CFL}_{\text{TM}}$   
 $\text{DECIDABLE}_{\text{TM}}$

} Rice's theorem



# Reductions via computation histories

---

- The computation history method is an important technique for proving that  $A_{TM}$  is reducible to certain languages
- Often useful when the problem to be shown undecidable involves testing for the existence of something
  - E.g. undecidability of Hilbert's tenth problem, testing for the existence of integral roots in a polynomial
- We will define computation history today and use it to show a decidable problem; in next lecture we will use it to show examples of undecidable problems



# Definition – computation history

---

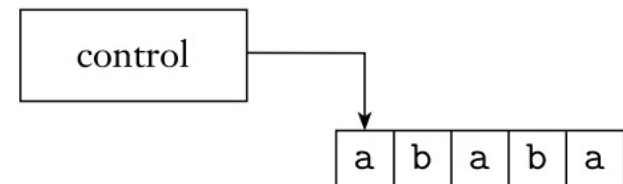
Let  $M$  be a Turing machine and  $w$  an input string. An ***accepting computation history*** for  $M$  on  $w$  is a sequence of configurations,  $C_1, C_2, \dots, C_l$ , where  $C_1$  is the start configuration of  $M$  on  $w$ ,  $C_l$  is an accepting configuration of  $M$ , and each  $C_i$  legally follows from  $C_{i-1}$  according to the rules of  $M$ . A ***rejecting computation history*** for  $M$  on  $w$  is defined similarly, except that  $C_l$  is a rejecting configuration.



## Definition – linear bounded automaton (LBA)

---

A *linear bounded automaton* is a restricted type of Turing machine wherein the tape head isn't permitted to move off the portion of the tape containing the input. If the machine tries to move its head off either end of the input, the head stays where it is—in the same way that the head will not move off the left-hand end of an ordinary Turing machine's tape.



An LBA is a TM with a limited amount of memory



# $A_{\text{LBA}}$

---

**Let:**

$$A_{\text{LBA}} = \{\langle M, w \rangle \mid M \text{ is an LBA that accepts string } w\}.$$

**Lemma:**

Let  $M$  be an LBA with  $q$  states and  $g$  symbols in the tape alphabet. There are exactly  $qng^n$  distinct configurations of  $M$  for a tape of length  $n$ .

**Theorem:**

$A_{\text{LBA}}$  is decidable.



# Proof – lemma on LBA

---

**PROOF** Recall that a configuration of  $M$  is like a snapshot in the middle of its computation. A configuration consists of the state of the control, position of the head, and contents of the tape. Here,  $M$  has  $q$  states. The length of its tape is  $n$ , so the head can be in one of  $n$  positions, and  $g^n$  possible strings of tape symbols appear on the tape. The product of these three quantities is the total number of different configurations of  $M$  with a tape of length  $n$ .



# Proof – theorem on LBA

**PROOF** The algorithm that decides  $A_{\text{LBA}}$  is as follows.

$L =$  “On input  $\langle M, w \rangle$ , where  $M$  is an LBA and  $w$  is a string:

1. Simulate  $M$  on  $w$  for  $qng^n$  steps or until it halts.
2. If  $M$  has halted, *accept* if it has accepted and *reject* if it has rejected. If it has not halted, *reject*.”

If  $M$  on  $w$  has not halted within  $qng^n$  steps, it must be repeating a configuration according to Lemma 5.8 and therefore looping. That is why our algorithm rejects in this instance.





# Hints on HW7

---



# Hints on HW 7

---

- **Problem 1:**

show that the collection of decidable languages is closed under: (a) complementation, (b) concatenation, and (c) star operation

- a) Given a decidable language  $L$ , let  $M$  be a TM that decides  $L$ . How can you construct a TM  $N$  that decides the complement of  $L$ ?
- b) Given any two decidable languages  $L_1$  and  $L_2$ , let  $M_1$  and  $M_2$  be the TMs that decide them. Show how you can construct a TM  $N$  that decides the concatenation of  $L_1$  and  $L_2$ .
  - hint: consider cutting an input string  $w$  into two parts
- c) Given a decidable language  $L$ , let  $M$  be a TM that decides  $L$ . Show how you can construct a TM  $N$  that decides  $L^*$ .
  - hint: consider cutting an input string  $w$  into  $n$  parts ( $n = |w|$ )



# Hints on HW 7

---

- **Problem 2:**

Let  $ALL_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^* \}$ .

Show that  $ALL_{DFA}$  is decidable.

- Hint:

- Construct a TM  $M$  that decides  $ALL_{DFA}$  on input  $\langle A \rangle$
- In the construction of  $M$ , consider how you make use of what you know about the complement of  $ALL_{DFA}$

- (side note: we will consider decidability of  $ALL_{CFG}$  in next lecture)



# Hints on HW 7

---

- **Problem 3:**

Let  $B$  be the set of all infinite sequences over  $\{0,1\}$ . Show that  $B$  is uncountable using a proof by diagonalization.

- **Hint:**

- Suppose a correspondence  $f: \mathbb{N} \rightarrow B$  exists only to arrive at a contradiction.
- Construct a sequence  $x$  in  $B$  that is not paired with anything in  $\mathbb{N}$ . (For inspiration on how to go about this, see what we did in the undecidability-part 1 lecture where we used the diagonalization method to show the set of real numbers is uncountable)



# Hints on HW 7

---

- **Problem 4:**

A *useless state* in a PDA is never entered on any input string. Consider the problem of determining whether a PDA has any useless state. Formulate this problem as a language and show that it is decidable.

- **Hint:**

- The language formulation is easy  
( $U = \{ \langle P \rangle \mid P \text{ is a PDA that has useless states} \}$ )
- Construct a TM  $M$  that decides  $U$   
(think what you would do to check *each* state of  $P$ )



# Hints on HW 7

---

- **Problem 5:**

If  $A \leq_M B$  (i.e.  $A$  is mapping reducible to  $B$ ) and  $B$  is a regular language, does that imply that  $A$  is a regular language? Why or why not?

- **Hint:**

- Language  $A$  is said to be mapping reducible to language  $B$  if there is a *computable* function  $f$  that maps every input of  $A$  to an element in  $B$
- It is just one way to formalize what we have been calling reduction so far in the past lectures

- (side note: we will discuss mapping reducibility as a topic in next lecture)



# Hints on HW 7

---

- **Problem 6:**

Use Rice's theorem to prove the undecidability of the following languages.

- a)  $\{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is an infinite language} \}$
- b)  $ALL_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^* \}$

- **Hint:**

- The statement of Rice's theorem is given in a footnote in the homework prompt. The theorem has **two conditions**.
- For each of the languages in a) and b), explain how the two conditions of Rice's theorems are satisfied.

- (side note: we encountered Rice's theorem in today's lecture)