

# BU CS 332 – Theory of Computation

## Lecture 16:

- Examples of Reductions
- Test 2 Review

Reading:

Sipser Ch 5.1

Mark Bun

March 15, 2021

# Reductions

A **reduction** from problem  $A$  to problem  $B$  is an algorithm for problem  $A$  which uses an algorithm for problem  $B$  as a subroutine

If such a reduction exists, we say “ $A$  reduces to  $B$ ”

**Positive uses:** If  $A$  reduces to  $B$  and  $B$  is decidable, then  $A$  is also decidable

Ex.  $E_{\text{DFA}}$  is decidable  $\Rightarrow EQ_{\text{DFA}}$  is decidable

**Negative uses:** If  $A$  reduces to  $B$  and  $A$  is undecidable, then  $B$  is also undecidable

Ex.  $A_{\text{TM}}$  is undecidable  $\Rightarrow HALT_{\text{TM}}$  is decidable

# Equality Testing for TMs

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

**Theorem:**  $EQ_{\text{TM}}$  is undecidable

**Proof:** Suppose for contradiction that there exists a decider  $R$  for  $EQ_{\text{TM}}$ . We construct a decider for  $E_{\text{TM}}$  as follows:

On input  $\langle M \rangle$ :

1. Construct TMs  $N_1, N_2$  as follows:

$$N_1 =$$

$$N_2 =$$

2. Run  $R$  on input  $\langle N_1, N_2 \rangle$

3. If  $R$  accepts, **accept**. Otherwise, **reject**.

This is a reduction from  $E_{\text{TM}}$  to  $EQ_{\text{TM}}$



# Equality Testing for TMs

What do we want out of the machines  $N_1, N_2$ ?

- a)  $L(M) = \emptyset$  iff  $N_1 = N_2$       b)  $L(M) = \emptyset$  iff  $L(N_1) = L(N_2)$   
c)  $L(M) = \emptyset$  iff  $N_1 \neq N_2$       d)  $L(M) = \emptyset$  iff  $L(N_1) \neq L(N_2)$

On input  $\langle M \rangle$ :

1. Construct TMs  $N_1, N_2$  as follows:

$N_1 =$

$N_2 =$

2. Run  $R$  on input  $\langle N_1, N_2 \rangle$

3. If  $R$  accepts, **accept**. Otherwise, **reject**.

This is a reduction from  $E_{TM}$  to  $EQ_{TM}$

# Equality Testing for TMs

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

**Theorem:**  $EQ_{\text{TM}}$  is undecidable

**Proof:** Suppose for contradiction that there exists a decider  $R$  for  $EQ_{\text{TM}}$ . We construct a decider for  $A_{\text{TM}}$  as follows:

On input  $\langle M \rangle$ :

1. Construct TMs  $N_1, N_2$  as follows:

$$N_1 =$$

$$N_2 =$$

2. Run  $R$  on input  $\langle N_1, N_2 \rangle$

3. If  $R$  accepts, **accept**. Otherwise, **reject**.

This is a reduction from  $E_{\text{TM}}$  to  $EQ_{\text{TM}}$

# Regular language testing for TMs

$$REG_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$$

**Theorem:**  $REG_{TM}$  is undecidable

**Proof:** Suppose for contradiction that there exists a decider  $R$  for  $REG_{TM}$ . We construct a decider for  $A_{TM}$  as follows:

On input  $\langle M, w \rangle$ :

1. Construct a TM  $N$  as follows:
2. Run  $R$  on input  $\langle N \rangle$
3. If  $R$  accepts, **accept**. Otherwise, **reject**

This is a reduction from  $A_{TM}$  to  $REG_{TM}$

# Regular language testing for TMs

$$REG_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is regular}\}$$

**Theorem:**  $REG_{TM}$  is undecidable

**Proof:** Suppose for contradiction that there exists a decider  $R$  for  $REG_{TM}$ . We construct a decider for  $A_{TM}$  as follows:

On input  $\langle M, w \rangle$ :

1. Construct a TM  $N$  as follows:

$N =$  “On input  $x$ ,

1. If  $x \in \{0^n 1^n \mid n \geq 0\}$ , accept

2. Run TM  $M$  on input  $w$

3. If  $M$  accepts, accept. Otherwise, reject.”

2. Run  $R$  on input  $\langle N \rangle$

3. If  $R$  accepts, **accept**. Otherwise, **reject**

This is a reduction from  $A_{TM}$  to  $REG_{TM}$

# Test 2 Topics



# Turing Machines (3.1, 3.3)

- Know the three different “levels of abstraction” for defining Turing machines and how to convert between them: Formal/state diagram, implementation-level, and high-level
- Know the definition of a configuration of a TM and the formal definition of how a TM computes
- Know how to “program” Turing machines by giving state diagrams and implementation-level descriptions
- Understand the Church-Turing Thesis

## TM Variants (3.2)

- Understand the following TM variants: TM with stay-put, TM with two-way infinite tape, Multi-tape TMs, Nondeterministic TMs
- Know how to give a simulation argument (implementation-level and high-level description) to compare the power of TM variants
- Understand the specific simulation arguments we've seen: two-way infinite TM by basic TM, multi-tape TM by basic TM, nondeterministic TM by basic TM

# Decidability (4.1)

- Understand how to use a TM to simulate another machine (DFA, another TM)
- Know the specific decidable languages from language theory that we've discussed, and how to decide them:  $A_{DFA}$ ,  $E_{DFA}$ ,  $EQ_{DFA}$ , etc.
- Know how to use a reduction to one of these languages to show that a new language is decidable

# Undecidability (4.2)

- Know the definitions of countable and uncountable sets and how to prove countability and uncountability
- Understand how diagonalization is used to prove the existence of an explicit undecidable language  $UD$
- Know that a language is decidable iff it is recognizable and its complement is recognizable, and understand the proof

# Reducibility (5.1)

- Understand how to use a reduction (contradiction argument) to prove that a language is undecidable
- Know the reductions showing that  $HALT_{TM}$ ,  $E_{TM}$ ,  $REGULAR_{TM}$ ,  $EQ_{TM}$  are undecidable
- You are **not** responsible for understanding the computation history method.

# True or False

- It's all about the justification!
- The logic of the argument has to be clear
- Restating the question is not justification; we're looking for additional insight

True. If  $A$  is finite, it is regular, as shown in class. The regular languages are closed under intersection, so  $A \cap B$  is also regular.

# Simulation arguments, constructing deciders

Give a simulation argument, using an implementation-level description, to show that TMs with reset recognize the class of Turing-recognizable languages. *Hint:* You may want to simulate using a two-tape TM. (12 points)

We simulate a TM with reset using a two-tape TM as follows. The first tape of the new machine is read-only and used to store the input. We initialize the second tape by marking the left end of the tape with a special symbol \$, copying the input, and then marking the right end of the input with another special symbol #. (These special symbols are in place to allow us to know how much of the second tape is actually in use during simulation).

To simulate one ordinary step (i.e., read, write, and move) of the TM with reset, we simulate its action on the second tape of our new machine, treating the cell containing \$ as the left end of the tape and moving the # symbol to the right by one cell if we ever try to overwrite it.

To simulate a reset step, we scan the second tape of the new machine between the \$ symbol and the # to erase its contents and re-initialize the second tape by copying the input from the first tape, again demarcated by \$ and #.

- Full credit for a clear and correct description of the new machine
- Can still be a good idea to provide an explanation (partial credit, clarifying ambiguity)

# Countability proofs

A *DNA strand* is a finite string over the alphabet  $\{A, C, G, T\}$ . Show that the set of all DNA strands is countable. (8 points)

We may list the elements of this set in stages  $i = 0, 1, 2, \dots$  as follows. In stage 0, we list the empty string, the only string of length 0. In stage 1, we list all strings of length 1, etc. In general, in stage  $i$ , we list all  $4^i$  strings of length  $i$ . We obtain a correspondence  $f$  from the set of natural numbers into this set of strings by taking  $f(n)$  to be the  $n$ th string in this list.

- Describe how to list all the elements in your set, usually in a succession of finite “stages”
- Describe how this listing process gives you a bijection from the natural numbers



# Uncountability proofs

Let  $\mathcal{F} = \{f : \mathbb{Z} \rightarrow \mathbb{Z}\}$  be the set of all functions taking as input an integer and outputting an integer. Show that  $\mathcal{F}$  is uncountable. (10 points)

Suppose for the sake of contradiction that  $\mathcal{F}$  were countable, and let  $B : \mathbb{N} \rightarrow \mathcal{F}$  be a bijection. For each  $i \in \mathbb{N}$ , let  $f_i = B(i)$ . Define the function  $g \in \mathcal{F}$  as follows. For every  $i = 1, 2, \dots$  let  $g(i) = f_i(i) + 1$ . For every  $i = 0, -1, -2, \dots$ , let  $g(i) = 0$ . This definition of the function  $g$  ensures that  $g(i) \neq f_i(i)$  for every  $i \in \mathbb{N}$ . Hence,  $g \neq f_i = B(i)$  for any  $i$ , which contradicts the onto property of the map  $B$ .

- The 2-D table is useful for helping you think about diagonalization, but does not need to appear in the proof
- The essential part of the proof is the construction of the “inverted diagonal” element, and the proof that it works

# Undecidability proofs

Show that the language  $Y$  is undecidable. (10 points)

We show that  $Y$  is undecidable by giving a reduction from  $A_{\text{TM}}$ . Suppose for the sake of contradiction that we had a decider  $R$  for  $Y$ . We construct a decider for  $A_{\text{TM}}$  as follows:

“On input  $\langle M, w \rangle$ :

1. Use  $M$  and  $w$  to construct the following TM  $M'$ :  
 $M' =$  “On input  $x$ :
  1. If  $x$  has even length, *accept*
  2. Run  $M$  on  $w$
  3. If  $M$  accepts, *accept*. If  $M$  rejects, *reject*.”
2. Run  $R$  on input  $\langle M' \rangle$
3. If  $R$  accepts, *reject*. If  $R$  rejects, *accept*.”

If  $M$  accepts  $w$ , then the machine  $M'$  accepts all strings. On the other hand, if  $M$  does not accept  $w$ , then  $M'$  only accepts strings of even length.

Hence this machine decides  $A_{\text{TM}}$  which is a contradiction, since  $A_{\text{TM}}$  is undecidable. Hence  $Y$  must be undecidable as well.

# Practice Problems









# Decidability and Recognizability



Let  $A = \{\langle D \rangle \mid$   
 $D$  is a DFA that does not accept any string  
containing an odd number of 1's}  
Show that  $A$  is decidable

Prove that  $\overline{E_{TM}}$  is recognizable

Prove that if  $A$  and  $B$  are decidable, then so is  $A \setminus B$

# Countable and Uncountable Sets

Show that the set of all valid (i.e., compiling without errors) C++ programs is countable

A Celebrity Twitter Feed is an infinite sequence of ASCII strings, each with at most 140 characters. Show that the set of Celebrity Twitter Feeds is uncountable.

# Undecidability and Unrecognizability

Prove or disprove: If  $A$  and  $B$  are recognizable, then so is  $A \setminus B$



Prove that the language  $ALL_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \Sigma^*\}$  is undecidable