

BU CS 332 – Theory of Computation

<https://forms.gle/H3hgiGk4nj4KMHnV7>



Lecture 15:

- More Examples of Reductions
- Mapping Reductions

Reading:

Sipser Ch 5.1, 5.3

Alexander Poremba & Mark Bun

March 26, 2026

Last Time: 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_{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. UD is undecidable $\Rightarrow A_{\text{TM}}$ is undecidable

Halting Problem

$HALT_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM that halts on input } w\}$

Theorem: $HALT_{TM}$ is undecidable

Proof: Suppose for contradiction that there exists a decider H for $HALT_{TM}$. We construct a decider for V for A_{TM} as follows:

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

1. Run H on input $\langle M, w \rangle$
2. If H rejects, **reject**
3. If H accepts, run M on w
4. If M accepts, **accept**
Otherwise, **reject**.

If $\langle M, w \rangle \in A_{TM}$:

M accepts input w

$\Rightarrow H$ accepts input $\langle M, w \rangle$

\Rightarrow Proceed to line 3; V accepts in line 4

If $\langle M, w \rangle \notin A_{TM}$, then either:

a) M rejects on input w

$\Rightarrow H$ accepts input $\langle M, w \rangle$

\Rightarrow Proceed to line 3; V rejects in line 4

or b) M loops forever on input w

$\Rightarrow H$ rejects input $\langle M, w \rangle$

$\Rightarrow V$ rejects in line 2

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

Emptiness testing for TMs

(Failed) First Attempt

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

Theorem: E_{TM} is undecidable

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

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

1. Run R on input M
2. If R rejects, **accept**.
If R accepts, **reject**

If $\langle M, w \rangle \in A_{\text{TM}}$:

M accepts input w

$\Rightarrow w \in L(M)$, so $L(M) \neq \emptyset$

$\Rightarrow R$ accepts input $\langle M \rangle \Rightarrow V$ accepts

If $\langle M, w \rangle \notin A_{\text{TM}}$:

M does not accept input w

$\Rightarrow L(M)$ may or may not be empty

$\Rightarrow R$ may or may not accept

$\Rightarrow V$ may or may not accept

This is a reduction from A_{TM} to E_{TM}

Emptiness testing for TMs



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

Theorem: E_{TM} is undecidable

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

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

1. Construct a TM N such that:

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

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

What do we want out of machine N ?

- a) $L(N)$ is empty iff M accepts w
- b) $L(N)$ is non-empty iff M accepts w
- c) $L(M)$ is empty iff N accepts w
- d) $L(M)$ is non-empty iff N accepts w

This is a reduction from A_{TM} to E_{TM}

Emptiness testing for TMs

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

Theorem: E_{TM} is undecidable

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

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

1. Construct a TM N as follows:

“On input x :

Run M on w and output the result.”

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

3. If R rejects, **accept**. Otherwise, **reject**

This is a reduction from A_{TM} to E_{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_{TM} is undecidable

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

On input $\langle M \rangle$:

1. Construct TMs N_1, N_2 as follows:

$$N_1 = \quad \quad \quad N_2 =$$

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



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_{TM} is undecidable

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

On input $\langle M \rangle$:

1. Construct TMs N_1, N_2 as follows:

$$N_1 = \text{“On input } x: \quad N_2 = M \\ \text{reject”}$$

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}

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}

Mapping Reductions

Warning



What's wrong with the following “proof”?

Bogus “Theorem”: A_{TM} is not Turing-recognizable

Bogus “Proof”: Let R be an alleged recognizer for A_{TM} . We construct a recognizer S for unrecognizable language A_{TM} :

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

1. Run R on input $\langle M, w \rangle$
2. If R accepts, **reject**. If R rejects, **accept**.

This sure looks like a reduction from $\overline{A_{TM}}$ to A_{TM}

Mapping Reductions: Motivation

1. How do we formalize the notion of a reduction?
2. How do we use reductions to show that languages are unrecognizable?
3. How do we protect ourselves from accidentally “proving” bogus statements about recognizability?

Computable Functions

Definition:

A function $f: \Sigma^* \rightarrow \Sigma^*$ is **computable** if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape. (“Outputs $f(w)$ ”)

Computable Functions

Definition:

A function $f: \Sigma^* \rightarrow \Sigma^*$ is **computable** if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape. (“Outputs $f(w)$ ”)

Example 1: $f(w) = \text{sort}(w)$

Example 2: $f(\langle x, y \rangle) = x + y$

Computable Functions

Definition:

A function $f: \Sigma^* \rightarrow \Sigma^*$ is **computable** if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape. (“Outputs $f(w)$ ”)

Example 3: $f(\langle M, w \rangle) = \langle M' \rangle$ where M is a TM, w is a string, and M' is a TM that ignores its input and simulates running M on w

Mapping Reductions

Definition:

Let $A, B \subseteq \Sigma^*$ be languages. We say A is **mapping reducible** to B , written

$$A \leq_m B$$

if there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \iff f(w) \in B$

Mapping Reductions



Definition:

Language A is **mapping reducible** to language B , written

$$A \leq_m B$$

if there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that for all strings $w \in \Sigma^*$, we have $w \in A \iff f(w) \in B$

If $A \leq_m B$, which of the following is true?

- a) $\bar{A} \leq_m B$
- b) $A \leq_m \bar{B}$
- c) $\bar{A} \leq_m \bar{B}$
- d) $\bar{B} \leq_m \bar{A}$

Decidability

Theorem: If $A \leq_m B$ and B is decidable, then A is also decidable

Proof: Let M be a decider for B and let $f: \Sigma^* \rightarrow \Sigma^*$ be a mapping reduction from A to B . We can construct a decider N for A as follows:

On input w :

1. Compute $f(w)$
2. Run M on input $f(w)$
3. If M accepts, **accept**.
If it rejects, **reject**.

Undecidability

Theorem: If $A \leq_m B$ and B is decidable, then A is also decidable

Corollary: If $A \leq_m B$ and A is undecidable, then B is also undecidable

Old Proof: 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_{TM} is undecidable

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

On input $\langle M \rangle$:

1. Construct TMs M_1, M_2 as follows:

$$M_1 = M$$

$M_2 =$ “On input x ,
1. Ignore x and **reject**”

2. Run R on input $\langle M_1, M_2 \rangle$

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

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

New Proof: 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: $E_{\text{TM}} \leq_m EQ_{\text{TM}}$ (Hence EQ_{TM} is undecidable)

Proof: The following TM N computes the reduction f :

On input $\langle M \rangle$:

1. Construct TMs M_1, M_2 as follows:

$$M_1 = M$$

$M_2 =$ “On input x ,
1. Ignore x and **reject**”

2. Output $\langle M_1, M_2 \rangle$

Mapping Reductions: Recognizability

Theorem: If $A \leq_m B$ and B is recognizable, then A is also recognizable

Proof: Let M be a recognizer for B and let $f: \Sigma^* \rightarrow \Sigma^*$ be a mapping reduction from A to B . Construct a recognizer N for A as follows:

On input w :

1. Compute $f(w)$
2. Run M on input $f(w)$
3. If M accepts, **accept**.
If it rejects, **reject**.

Unrecognizability

Theorem: If $A \leq_m B$ and B is recognizable, then A is also recognizable

Corollary: If $A \leq_m B$ and A is **un**recognizable, then B is also **un**recognizable

Corollary: If $\overline{A_{TM}} \leq_m B$, then B is **un**recognizable

Other Undecidable Problems

Problems in Language Theory

Apparent dichotomy:

- TMs seem to be able to solve problems about the power of weaker computational models (e.g., DFAs)
- TMs can't solve problems about the power of TMs themselves

Question: Are there undecidable problems that do not involve TM descriptions?

A_{DFA} decidable	A_{TM} undecidable
E_{DFA} decidable	E_{TM} undecidable
EQ_{DFA} decidable	EQ_{TM} undecidable

Undecidability of mathematics [Sipser 6.2]

Peano arithmetic: Formalization of mathematical statements about the natural numbers, using $+$, \times , \leq

Ex: “There exist infinitely many primes”

Theorem [Church, Turing]:

$\text{TPA} = \{ \langle \varphi \rangle \mid \varphi \text{ is a true statement in PA} \}$ is undecidable

Corollary [Gödel’s First Incompleteness Theorem]:

There exists a true statement φ in Peano arithmetic that is not provable

A simple undecidable problem

Post Correspondence Problem (PCP) [Sipser 5.2]:

Domino: $\begin{bmatrix} a \\ ab \end{bmatrix}$. Top and bottom are strings.

Input: Collection of dominos.

$$\begin{bmatrix} aa \\ aba \end{bmatrix}, \begin{bmatrix} ab \\ aba \end{bmatrix}, \begin{bmatrix} ba \\ aa \end{bmatrix}, \begin{bmatrix} abab \\ b \end{bmatrix}$$

Match: List of some of the input dominos (repetitions allowed) where top = bottom

$$\begin{bmatrix} ab \\ aba \end{bmatrix}, \begin{bmatrix} aa \\ aba \end{bmatrix}, \begin{bmatrix} ba \\ aa \end{bmatrix}, \begin{bmatrix} aa \\ aba \end{bmatrix}, \begin{bmatrix} abab \\ b \end{bmatrix}$$

Problem: Does a match exist?

This is undecidable