

BU CS 332 – Theory of Computation

<https://forms.gle/qdHnZnf77qMHFBrbA>



Lecture 16:

- Mapping Reduction Examples
- Asymptotic Notation

Reading:

Sipser Ch 5.3, 7.1

Mark Bun & Alexander Poremba

March 31, 2026

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?

Mapping Reductions

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 $f(w)$ ”)

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 $w \in A \iff f(w) \in B$

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

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 $\langle M_1, M_2 \rangle$

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

Recognizability and A_{TM}



Let L be a language. Which of the following is true?

- a) If $L \leq_m A_{TM}$, then L is recognizable
- b) If $A_{TM} \leq_m L$, then L is recognizable
- c) If L is recognizable, then $L \leq_m A_{TM}$
- d) If L is recognizable, then $A_{TM} \leq_m L$

Theorem: L is recognizable *if and only if* $L \leq_m A_{TM}$

Recognizability and A_{TM}

Theorem: L is recognizable *if and only if* $L \leq_m A_{\text{TM}}$

Proof:

Example: Another reduction to EQ_{TM}

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

Theorem: $A_{TM} \leq_m EQ_{TM}$

Proof: The following TM N computes the reduction f :



What should the inputs and outputs to f be?

- a) f should take as input a pair $\langle M_1, M_2 \rangle$ and output a pair $\langle M, w \rangle$
- b) f should take as input a pair $\langle M, w \rangle$ and output a pair $\langle M_1, M_2 \rangle$
- c) f should take as input a pair $\langle M_1, M_2 \rangle$ and either accept or reject
- d) f should take as input a pair $\langle M, w \rangle$ and either accept or reject

Example: Another reduction to EQ_{TM}

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

Theorem: $A_{TM} \leq_m EQ_{TM}$

Proof: The following TM computes the reduction f :

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

1. Construct TMs M_1, M_2 as follows:

$M_1 =$ “On input x ,

$M_2 =$ “On input x ,

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

Consequences of $A_{\text{TM}} \leq_m EQ_{\text{TM}}$

1. Since A_{TM} is undecidable, EQ_{TM} is also undecidable
2. $A_{\text{TM}} \leq_m EQ_{\text{TM}}$ implies $\overline{A_{\text{TM}}} \leq_m \overline{EQ_{\text{TM}}}$
Since $\overline{A_{\text{TM}}}$ is unrecognizable, $\overline{EQ_{\text{TM}}}$ is unrecognizable

EQ_{TM} itself is also unrecognizable

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

Theorem: $\overline{A_{TM}} \leq_m EQ_{TM}$ (Hence EQ_{TM} is unrecognizable)

Proof: The following TM computes the reduction:

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

1. Construct TMs M_1, M_2 as follows:

M_1 = “On input x ,

1. Ignore x
2. Run M on input w
3. If M accepts, **accept**.
Otherwise, **reject**.”

M_2 = “On input x ,

1. Ignore x and **reject**”

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

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

Where we are in CS 332

Automata	Computability	Complexity
----------	---------------	------------

Previous unit: **Computability theory**

What problems can / can't computers solve?

Final unit: **Complexity theory**

What problems can / can't computers solve under
constraints on their computational resources?

Time and space complexity

Today: Start answering the basic questions

1. How do we measure complexity? (as in CS 330)
2. Asymptotic notation (as in CS 330)
3. How robust is the TM model when we care about measuring complexity?
4. How do we mathematically capture our intuitive notion of “efficient algorithms”?

Time and space complexity

Time complexity of a TM = Running time of an algorithm
= Max number of steps as a function of input length n

Space complexity of a TM = Memory usage of algorithm
= Max number of tape cells as a function of input length n

Review of asymptotic notation

O -notation (upper bounds)

$f(n) = O(g(n))$ means:

There **exist** constants $c > 0$, $n_0 > 0$ such that
 $f(n) \leq cg(n)$ for every $n \geq n_0$

Example: $2n^2 + 12 = O(n^3)$ ($c = 3$, $n_0 = 4$)

Properties of asymptotic notation:

Transitive:

$f(n) = O(g(n))$ and $g(n) = O(h(n))$ means $f(n) = O(h(n))$

Not reflexive:

$f(n) = O(g(n))$ does **not** mean $g(n) = O(f(n))$



Example: $f(n) = 2n^2$, $g(n) = n^3$

Alternative (better) notation: $f(n) \in O(g(n))$

Examples

- $10^6 n^3 + 2n^2 - n + 10 =$

- $\sqrt{n} + \log n =$

- $n (\log n + \sqrt{n}) =$

Little-oh

If O -notation is like \leq , then o -notation is like $<$

$f(n) = o(g(n))$ means:

For **every** constant $c > 0$, there **exists** $n_0 > 0$ such that
 $f(n) \leq cg(n)$ for every $n \geq n_0$

Example: $2n^2 + 12 = o(n^3)$ ($n_0 = \max\{4/c, 3\}$)

True facts about asymptotic expressions

Which of the following statements is true about the function $f(n) = 2^n$?

a) $f(n) = O(3^n)$

b) $f(n) = o(3^n)$

c) $f(n) = O(n^2)$

d) $n^2 = O(f(n))$



Asymptotic notation within expressions

Asymptotic notation within an expression is shorthand for “there exists a function satisfying the statement”

Examples:

- $n^{O(1)}$
- $n^2 + O(n)$
- $(1 + o(1))n$

FAABs: Frequently asked asymptotic bounds

- **Polynomials.** $a_0 + a_1n + \dots + a_d n^d$ is $O(n^d)$ if $a_d > 0$
- **Logarithms.** $\log_a n = O(\log_b n)$ for all constants $a, b > 0$

For every $c > 0$, $\log n = o(n^c)$

- **Exponentials.** For all $b > 1$ and all $d > 0$, $n^d = o(b^n)$
- **Factorial.** $n! = n(n-1) \cdots 1$

By Stirling's formula,

$$n! = \left(\sqrt{2\pi n}\right) \left(\frac{n}{e}\right)^n (1 + o(1)) = 2^{O(n \log n)}$$