BU CS 332 – Theory of Computation

https://forms.gle/ziHz9k95spAvq9NK6

Lecture 17:

- Mapping Reduction Examples
- Asymptotic Notation

Reading:

Sipser Ch 5.3, 7.1

Mark Bun

March 31, 2025

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:

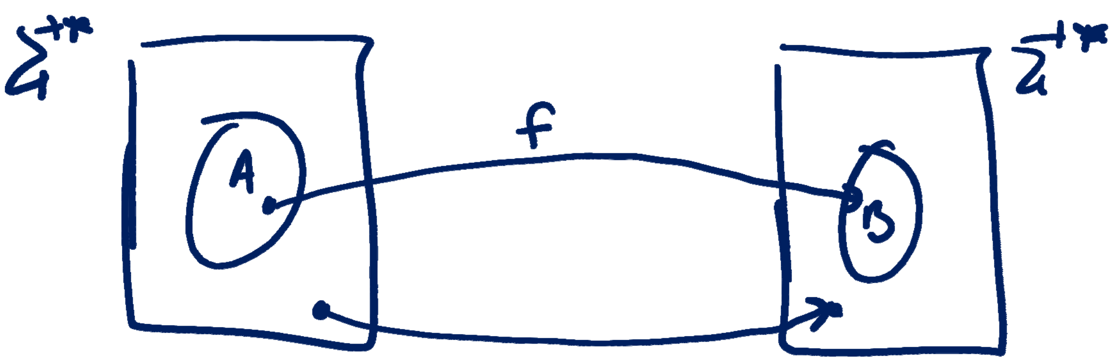
A function $f: \Sigma^* \to \Sigma^*$ is computable if there is a TM M which, given as input any $w \in \Sigma^*$, halts with only $f(w)$ on its tape. ("Outputs $f(w)$")

Definition:

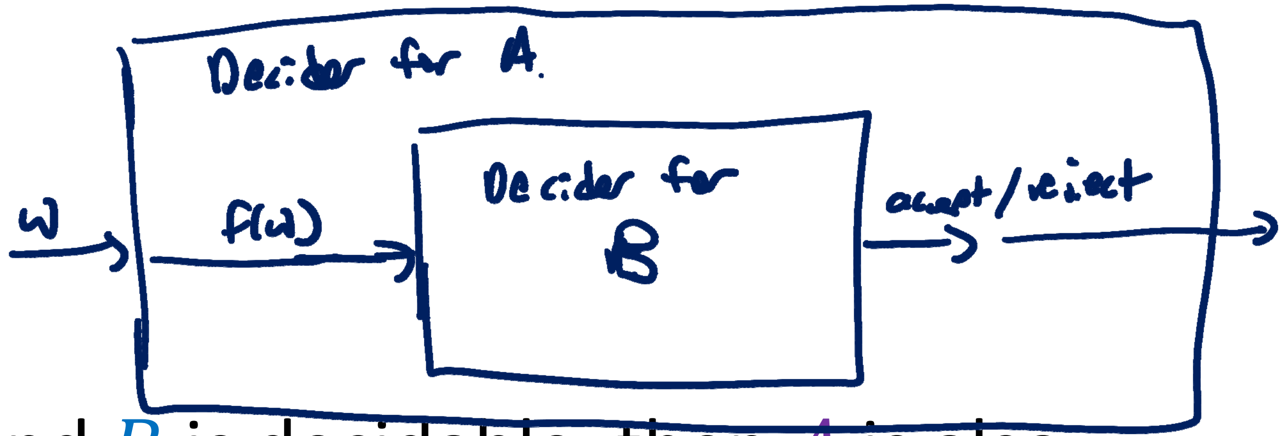
Let $A, B \subseteq \Sigma^*$ be languages. We say A is mapping reducible to B, written

$$A \leq_{\mathrm{m}} B$$

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



Undecidability



Theorem: If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is also decidable

Corollary: If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is also undecidable

New Proof: Equality Testing for TMs

undecidable

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

Theorem: $\boxed{E_{\text{TM}}} \leq_{\text{m}} EQ_{\text{TM}}$ (Hence EQ_{TM} is undecidable)

Proof: The following TM N computes the reduction f:

$E_{TM} = \{\langle M\rangle \mid \text{TM } M \text{ recognizes } \emptyset\}$

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 $\langle M_1, M_2\rangle$

Proof of correctness for reduction f:

- If $\langle M\rangle \in E_{TM}$
  $L(M_1) = L(M) = \emptyset$
  $L(M_2) = \emptyset \Rightarrow \langle M_1, M_2\rangle \in EQ_{TM}$
- If $\langle M\rangle \notin E_{TM}$
  $L(M_1) = L(M) \neq \emptyset$
  $L(M_2) = \emptyset \Rightarrow \langle M_1, M_2\rangle \notin EQ_{TM}$

Unrecognizability

Theorem: If $A \leq_{\mathrm{m}} B$ and B is recognizable, then A is also recognizable

Corollary: If $A \leq_{\mathrm{m}} B$ and A is unrecognizable, then B is also unrecognizable

Corollary: If $\overline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} B$, then B is unrecognizable

$\overline{A_{TM}} = \{\langle M, w\rangle \mid$ TM M does not accept $w\}$

Corollary: If $A_{TM} \leq_m \overline{B}$ then B is unrecognizable

Recognizability and A_{TM}

Know: A_{TM} is recognizable

Thm: $A \le_m B$ and B is recognizable $\Rightarrow$ A recognizable

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

a) If $L \le_{\mathrm{m}} A_{\mathrm{TM}}$, then L is recognizable
b) If $A_{\mathrm{TM}} \le_{\mathrm{m}} L$, then L is recognizable
c) If L is recognizable, then $L \le_{\mathrm{m}} A_{\mathrm{TM}}$
d) If L is recognizable, then $A_{\mathrm{TM}} \le_{\mathrm{m}} L$

Theorem: L is recognizable *if and only if* $L \le_{\mathrm{m}} A_{\mathrm{TM}}$

Recognizability and A_{TM}

Theorem: L is recognizable *if and only if* $L \leq_{\mathrm{m}} A_{\mathrm{TM}}$

Proof: $\Leftarrow$ If $L \leq_m A_{TM}$, then combine
1) A_{TM} is recognizable
2) $A \leq_m B$, B recognizable $\Rightarrow A$ recognizable

$\Rightarrow$ Suppose L recognizable. We construct a mapping reduction from L to A_{TM} as follows:

Let M be a TM that recognizes L

On input w: // w is an instance of problem L

Output $\langle M, w \rangle$

Proof of correctness:

- $w \in L \Rightarrow M$ accepts w
$\Rightarrow \langle M, w \rangle \in A_{TM}$
$\Rightarrow f(w) \in A_{TM}$

- $w \notin L \Rightarrow M$ does not accept w
$\Rightarrow \langle M, w \rangle \notin A_{TM}$
$\Rightarrow f(w) \notin A_{TM}$

A_{TM} is "complete" for the class of Turing-recognizable languages under mapping reductions

Example: Another reduction to EQ_{TM}

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

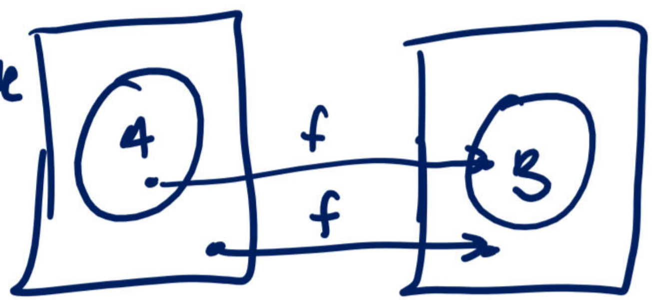
Theorem: $A_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$ $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid$ TM M accepts $w\}$

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 $\boxed{\langle M, w\rangle}$ and output a pair $\boxed{\langle M_1, M_2\rangle}$
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

$A \leq_m B$ means $\exists$ computable f s.t.



Example: Another reduction to EQ_{TM}

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

Theorem: $A_{\mathrm{TM}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$

Proof: The following TM computes the reduction f:

Want: M accepts $w \Rightarrow L(M_1) = L(M_2)$ $(= \Sigma^*)$

M does not accept $w \Rightarrow L(M_1) \neq L(M_2)$ $(= \Sigma^*)$

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"

M_2 = "On input x,
accept"

$L(M_1) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$

$L(M_2) = \Sigma^*$

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

Consequences of $A_{\text{TM}} \leq_{\text{m}} EQ_{\text{TM}}$

1. Since A_{TM} is undecidable, EQ_{TM} is also undecidable

2. $A_{\text{TM}} \leq_{\text{m}} EQ_{\text{TM}}$ implies $\overline{A_{\text{TM}}} \leq_{\text{m}} \overline{EQ_{\text{TM}}}$
   Since $\overline{A_{\text{TM}}}$ is unrecognizable, $\overline{EQ_{\text{TM}}}$ is unrecognizable

EQ_{TM} itself is also unrecognizable

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

Theorem: $\overline{A_{\text{TM}}} \leq_{\text{m}} EQ_{\text{TM}}$ (Hence EQ_{TM} is unrecognizable)

Proof: The following TM computes the reduction:

$\langle M, w\rangle \in \overline{A_{TM}} \Rightarrow M$ does not accept w
$\Rightarrow L(M_1) = \emptyset = L(M_2) \Rightarrow \langle M_1, M_2\rangle \in EQ_{TM}$

$\langle M, w\rangle \notin \overline{A_{TM}} \Rightarrow$
M accepts $w \Rightarrow$
$L(M_1) = \Sigma^* \neq \emptyset = L(M_2) \Rightarrow$
$\langle M_1, M_2\rangle \notin EQ_{TM}$.

$L(M_2) = \emptyset$

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~~ If M rejects, reject."

M_2 = "On input x,
1. Ignore x and reject"

$$L(M_1) = \begin{cases} \Sigma^* & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

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

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

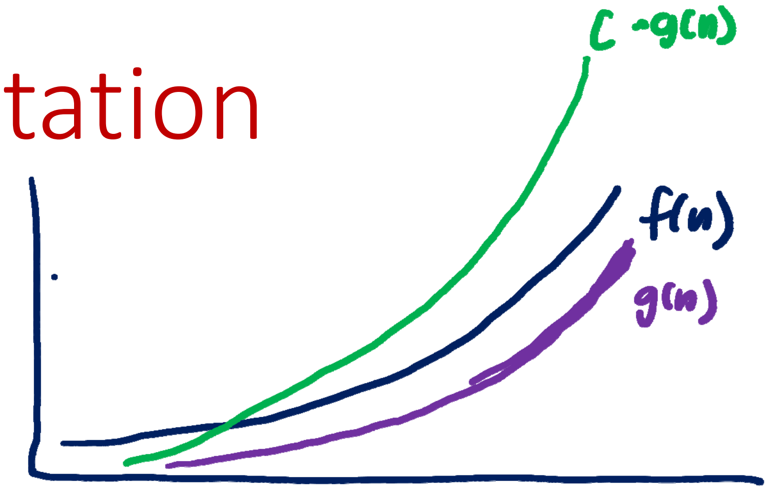
= Max number of steps as a <u>function</u> of input length n

Space complexity of a TM = Memory usage of algorithm

= Max number of tape cells as a <u>function</u> 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: $2n^2 + 12 = O(n^3)$ $(c = 3, n_0 = 4)$

$2n^2 + 12 \leq 2n^2 + n^2$ for all $n \geq n_0 = 4$

(why?. $n \geq 4 \Rightarrow n^2 \geq 16 \geq 12$)

$\leq 3n^2$

Properties of asymptotic notation:

Transitive:

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

Ex: $10n^2 + 17 = O(n^2)$, $n^2 = O(n^3)$ $\Rightarrow$ $10n^2+17 = O(n^3)$

Not reflexive:

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

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

$2n^2 = O(n^3)$ but $n^3 \neq O(n^2)$

Alternative (better) notation: $f(n) \in O(g(n)) = \{ h(n) \mid \exists\, c, n_0 \text{ s.t. } h(n) \le c \cdot g(n) \}$

Examples

- $\underline{10^6\,n^3} + 2n^2 - n + 10 = O(n^3)$

  $= O(n^{27})$ [also true, but not tight]

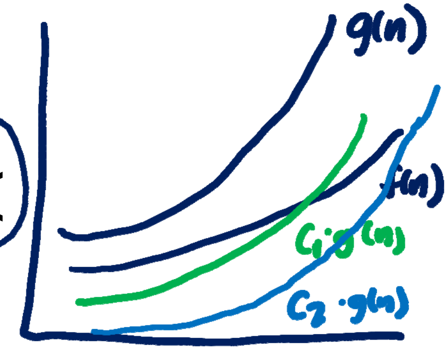
- $\underline{\sqrt{n}} + \underbrace{\log n} = O(\sqrt{n}) = O(n^{1/2})$

$c = 2$ $n_0 = 2$	$\log n \le \sqrt{n} \;\; \forall n \ge n_0$

  logs grow more slowly than all polynomials

- $n(\log n + \sqrt{n}) = n \log n + n\sqrt{n} = O(n\sqrt{n}) = O(n^{3/2})$

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 $n \geq n_0$

$\iff \forall c > 0 \ \exists n_0 > 0$ s.t. $\frac{f(n)}{g(n)} \leq c$

$\iff \lim_{n \to \infty} \frac{f(n)}{g(n)} = 0$

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

Let $c > 0$ be arbitrary

Choose $n_0 = \max\{4/c, 3\}$

$$
\begin{aligned}
2n^2 + 12 &\leq 2n^2 + n^2 \quad \forall n \geq 3 \\
&= 3n^2 \\
&\leq (cn)n^2 \quad \forall n \geq \tfrac{4}{c} \\
&= cn^3
\end{aligned}
$$

$$\frac{2n^2+12}{n^3} = \underbrace{\frac{2}{n}}_{\to 0} + \underbrace{\frac{12}{n^3}}_{\to 0}$$

True facts about asymptotic expressions

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

$2^n \neq O(n^2+2)$

a) $f(n) = O(3^n)$ $\forall n \quad 2^n \leq 3^n$

b) $f(n) = o(3^n)$ $\frac{2^n}{3^n} = \left(\frac{2}{3}\right)^n \to 0$

c) $f(n) = O(n^2)$ Exponentials grow faster than polynomials

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:

- $n^{O(1)}$ means there exists some $f(n) = O(1)$ s.t. the expression is $n^{f(n)}$

  $\Leftrightarrow \leq n^c$ for some constant c

  Ex: n^2, $n^{17{,}000{,}000} = n^{O(1)}$

- $n^2 + O(n)$

  means $n^2 + f(n)$ for some $f(n) = O(n)$

- $(1 + o(1))n$ = $n +$ $o(1) \cdot n$ = $n +$ $f(n)$ where

  $f(n) = o(n)$

FAABs: Frequently asked asymptotic bounds

- **Polynomials.** $a_0 + a_1 n + \ldots + a_d n^d$ is $O(n^d)$ if $a_d > 0$
- **Logarithms.** $\log_a n = O(\log_b n)$ for all constants $a, b > 0$

$\log_a(n) = \dfrac{\log_b n}{\log_b a}$

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,

$\approx n^{O(n)}$.

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