BU CS 332 – Theory of Computation

https://forms.gle/GmpaSseFf4DfeHw78

Lecture 15:

- Undecidable and Unrecognizable Languages
- Reductions

Reading:

Sipser Ch 4.2, 5.1

Mark Bun

March 24, 2025

Undecidability

Last time: Countability, uncountability, and diagonalization

Existential proof that there are undecidable and unrecognizable languages

Today: An explicit undecidable language

Reductions: Relate decidability / undecidability of different problems

Specializing the proof

Theorem: Let X be the set of all TM deciders. Then there exists an undecidable language in $P(\{0,1\}^*)$

1) Consider the function $L: X \to P(\{0,1\}^*)$

2) "Flip the diagonal" to construct a language $UD \in P(\{0,1\}^*)$ such that $L(M) \neq UD$ for every $M \in X$

3) Conclude that UD is undecidable, hence L is not onto

An explicit undecidable language

Does TM M_2 accept on input $\langle M_3 \rangle$?
If accepts, write Y
If rejects, write N

TM M	$M(\langle M_1 \rangle)$?	$M(\langle M_2 \rangle)$?	$M(\langle M_3 \rangle)$?	$M(\langle M_4 \rangle)$?		$D(\langle D \rangle)$?
M_1	~~Y~~ N	N	Y	Y	…	
M_2	N	~~N~~ Y	$\boxed{\text{Y}}$	Y		
M_3	Y	Y	~~Y~~ N	N		
M_4	N	N	Y	~~N~~ Y		
⋮					⋱	
D						~~Y~~ N | ~~N~~ Y

$UD = \{\langle M \rangle \mid M \text{ is a TM that does not accept on input } \langle M \rangle\}$

Claim: UD is undecidable

Assume FTSOC UD were decidable by some TM D

Case 1: $D(\langle D \rangle)$ accepts $\Rightarrow$ by def of UD, $\langle D \rangle \notin UD$ $\Rightarrow$ D messed up on input $\langle D \rangle$

Case 2: $D(\langle D \rangle)$ rejects $\Rightarrow$ by def of UD, $\langle D \rangle \in UD$ $\Rightarrow$ D messed up on input $\langle D \rangle$

D decides UD means: • ∀ ⟨M⟩ ∈ UD, D accepts ⟨M⟩
- ∀ ⟨M⟩ ∉ UD, D rejects ⟨M⟩

An explicit undecidable language

Theorem: $UD = \{\langle M\rangle \mid M$ is a TM that does **not** accept on input $\langle M\rangle\}$ is undecidable

Proof: Suppose for contradiction that some TM D decides UD

D always halts by assumption

Two cases:

M = D

1) D accepts on input ⟨D⟩ ⇒ ⟨D⟩ ∉ UD (by def'n of UD)
⇒ D made a mistake on ⟨D⟩

2) D rejects on input ⟨D⟩ ⇒ ⟨D⟩ ∈ UD (by def'n of UD)
⇒ D made a mistake on ⟨D⟩

In either case, D messes up on input ⟨D⟩, so D does not decide UD

Since D was an arbitrary TM decider, conclude UD undecidable.

A more useful undecidable language

$A_{\mathrm{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$

Theorem: A_{TM} is undecidable

Computational problem: Given TM M, input w, does M accept w?

Proof: Assume for the sake of contradiction that TM H decides A_{TM}:

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

Idea: Show that H can be used to construct a decider for the (undecidable) language UD -- a contradiction.

A more useful undecidable language

$A_{\text{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM that accepts input } w\}$

Proof (continued):

$\langle M, w\rangle \in A_{\text{TM}} \iff$ TM M accepts input w
$\langle M, w\rangle \notin A_{\text{TM}} \iff$ TM M does not accept input w

Suppose, for contradiction, that H decides A_{TM}

Consider the following TM D:

> "On input $\langle M\rangle$ where M is a TM:
> 1. Run H on input $\langle M, \langle M\rangle\rangle$
> 2. If H accepts, reject. If H rejects, accept."

Claim: D decides $UD = \{\langle M\rangle \mid \text{TM } M \text{ does not accept } \langle M\rangle\}$

- $\langle M\rangle \in UD \Rightarrow M$ does not accept input $\langle M\rangle$ "w" (by def. of UD)
  $\Rightarrow \langle M, \langle M\rangle\rangle \notin A_{\text{TM}}$
  $\Rightarrow H$ rejects on input $\langle M, \langle M\rangle\rangle$ since H decides A_{TM}
  $\Rightarrow D$ accepts on input $\langle M\rangle$. ✓
- $\langle M\rangle \notin UD \Rightarrow M$ accepts input $\langle M\rangle$ (by def. of UD)
  $\Rightarrow \langle M, \langle M\rangle\rangle \in A_{\text{TM}}$
  $\Rightarrow H$ accepts on input $\langle M, \langle M\rangle\rangle$ since H decides A_{TM}
  $\Rightarrow D$ rejects on input $\langle M\rangle$. ✓

…but this language is undecidable!

Unrecognizable Languages

Theorem: A language L is decidable if and only if L and $\bar{L}$ are both Turing-recognizable.

recognizable by universal TM

Corollary: $\overline{A_{\text{TM}}}$ is unrecognizable

Proof: know A_{TM} is undecidable $\Rightarrow$ either A_{TM} or $\overline{A_{TM}}$ unrecognizable (by Thm)

$\Rightarrow$ $\overline{A_{TM}}$ unrecognizable

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

Proof of Theorem:

$\Rightarrow$) Assume L decidable $\Rightarrow$ L is recognizable

$\hookrightarrow$ $\bar{L}$ is decidable $\Rightarrow$ $\bar{L}$ is recognizable

Unrecognizable Languages

Theorem: A language L is decidable if and only if L and $\overline{L}$ are both Turing-recognizable.

Proof continued: $\Leftarrow$ Suppose both L and $\overline{L}$ recognizable, by TMs M_1 and M_2 resp.

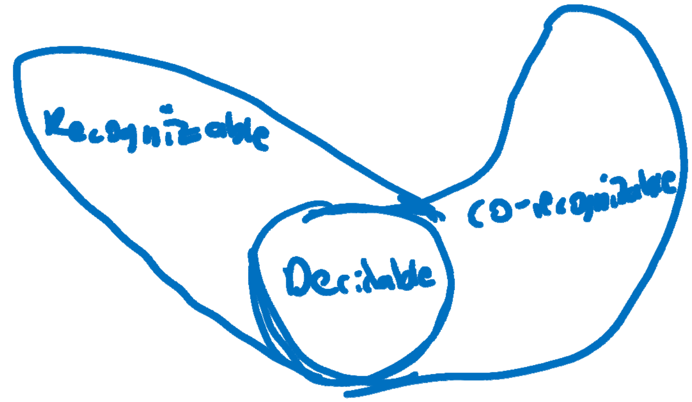
Goal: Use M_1, M_2 to construct a decider N for L

TM N:

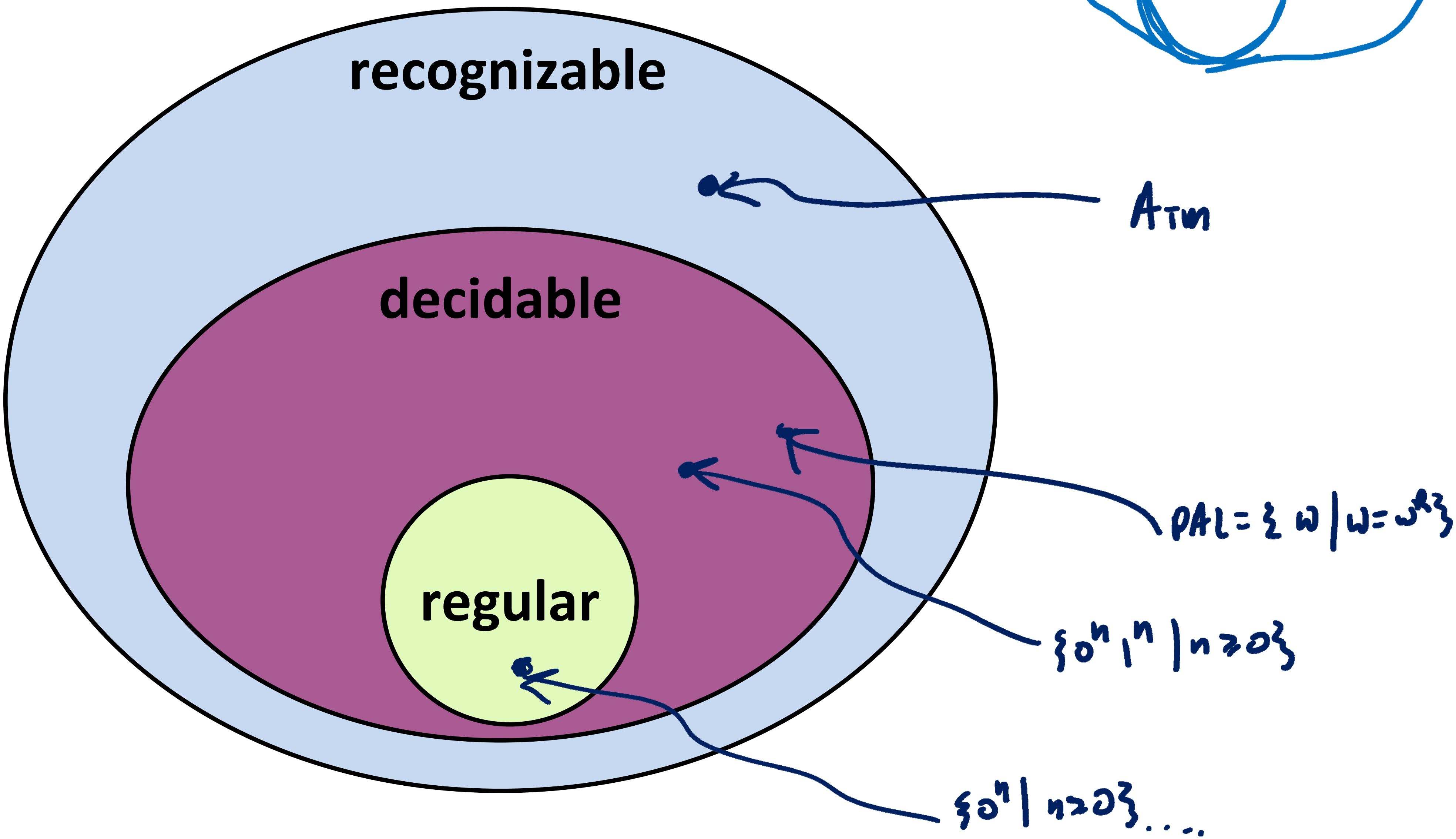
On input w:

1) Repeat until termination:
   a) Run M_1 for one additional step on input w. If accepts, accept.
   b) Run M_2 for one step on input w. If accepts, reject.

Classes of Languages



• $\overline{A_{TM}}$



Reductions

Scientists vs. Engineers

A computer scientist and an engineer are stranded on a desert island. They find two palm trees with one coconut on each. The engineer climbs a tree, picks a coconut and eats.

The computer scientist climbs the second tree, picks a coconut, climbs down, climbs up the first tree and places it there, declaring success.

“Now we’ve reduced the problem to one we’ve already solved.”

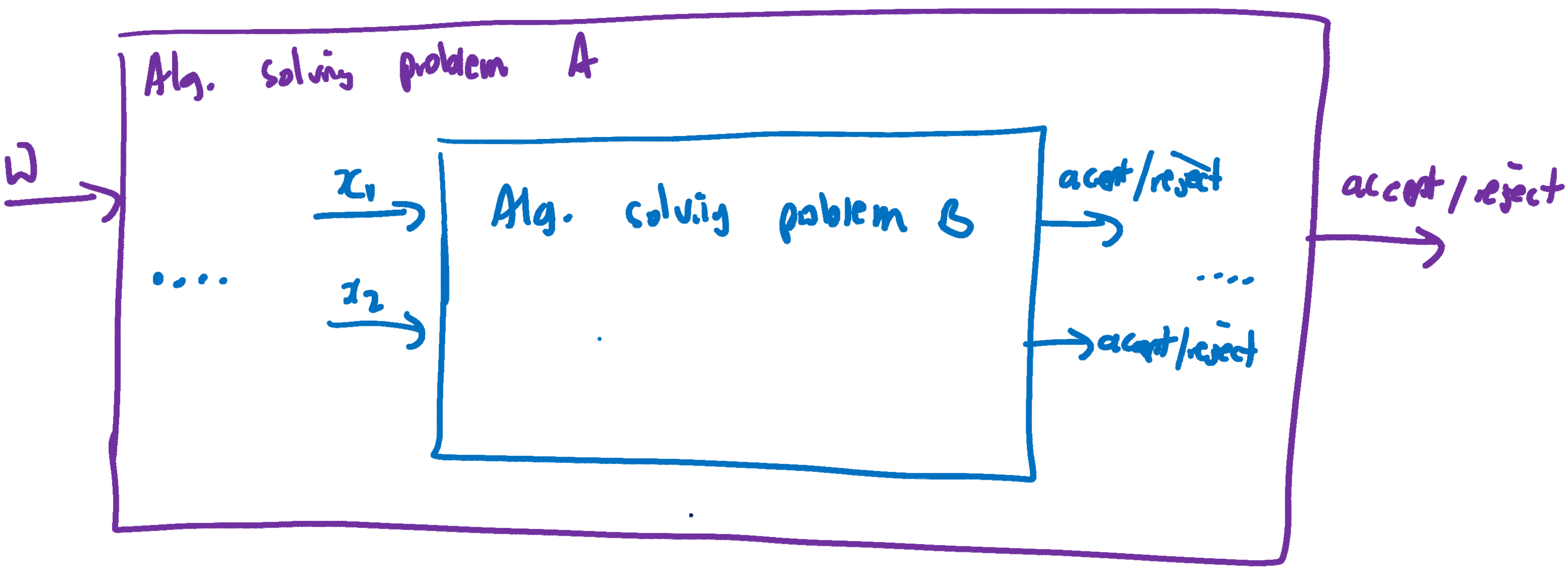
(Please laugh)

Reductions

Eat coconut from Tree 2   Eat coconut from Tree 1

A reduction from problem A to problem B is an algorithm solving problem A which uses an algorithm solving problem B as a subroutine

If such a reduction exists, we say "A reduces to B"



Two uses of reductions

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

$E_{DFA} = \{\langle D \rangle \mid D$ is a DFA where $L(D) = \emptyset\}$

$EQ_{\mathrm{DFA}} = \{\langle D_1, D_2\rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2)\}$

Theorem: EQ_{DFA} is decidable

Proof: The following TM decides EQ_{DFA}

Reduction from EQ_{DFA} to E_{DFA}

On input $\langle D_1, D_2\rangle$, where $\langle D_1, D_2\rangle$ are DFAs:

1. Construct a DFA D that recognizes the symmetric difference $L(D_1) \triangle L(D_2)$
2. Run the decider for E_{DFA} on $\langle D \rangle$ and return its output

Two uses of reductions

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

$UD = \{\langle M \rangle \mid \text{TM } M \text{ does not accept input } \langle M \rangle\}$

$A_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM that accepts input } w\}$

Suppose H decides A_{TM}

Reduction from UD to A_{TM}

Consider the following TM D.

On input $\langle M \rangle$ where M is a TM:

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

Claim: If H decides A_{TM} then D decides

$$UD = \{\langle M \rangle \mid M \text{ is a TM that does not accept input } \langle M \rangle\}$$

Two uses of reductions

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

Template for undecidability proof by reduction:

1. Suppose to the contrary that B is decidable
2. Using a decider for B as a subroutine, construct an algorithm deciding A
3. But A is undecidable. Contradiction!

Halting Problem

Computational problem: Given a program (TM) and input w, does that program halt (either accept or reject) on input w?

Formulation as a language:

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

Ex. M = "On input x (a natural number written in binary):

For each $y = 1, 2, 3, \ldots$:

If $y^2 = x$, accept. Else, continue."

Is $\langle M, \overbrace{101}^{5}\rangle \in HALT_{\mathrm{TM}}$?

a) Yes, because M accepts on input 101

b) Yes, because M rejects on input 101

c) No, because M rejects on input 101

d) No, because M loops on input 101

Halting Problem

Computational problem: Given a program (TM) and input w, does that program halt (either accept or reject) on input w?

Formulation as a language:

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

Ex. M = "On input x (a natural number in binary):
For each $y = 1, 2, 3, \ldots$:
If $y^2 = x$, accept. Else, continue."

M' = "On input x (a natural number in binary):
For each $y = 1, 2, 3, \ldots, x$:
If $y^2 = x$, accept. Else, continue.
Reject."

$\langle M', 101 \rangle \in HALT_{TM}$.

Halting Problem

Reduce from A_{TM} to $HALT_{TM}$

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

Theorem: $HALT_{\text{TM}}$ is undecidable

Proof: Suppose for contradiction that there exists a decider H for $HALT_{\text{TM}}$. 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.

→ Use H to test whether M will make a decision on w or not

→ Run M on w

- $\langle M, w\rangle \in A_{TM} \Rightarrow$ M accepts input w
  $\Rightarrow$ H accepts input $\langle M, w\rangle$ ✓
  $\Rightarrow$ proceed to line 3, V accepts in line 4
- $\langle M, w\rangle \notin A_{TM}$
  a) M loops on w $\Rightarrow$
  H rejects input $\langle M, w\rangle$
  $\Rightarrow$ V rejects in line 2 ✓
  b) M rejects on w $\Rightarrow$
  H accepts $\langle M, w\rangle$
  $\Rightarrow$ proceed to line 3, V rejects in line 4 ✓

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