BU CS 332 – Theory of Computation

https://forms.gle/LMB5MR8hSc5mxVt4A

Lecture 14:

- Undecidability
- Reductions

Reading:

Sipser Ch 4.2, 5.1

HW 6 deadline extended Wed, 11:59PM

Mark Bun

October 26, 2021

Where we are and where we're going

Church-Turing thesis: TMs capture all algorithms

Consequence: studying the limits of TMs reveals the limits of computation

{TM deciders} is countable
{languages over {0,1}} is uncountable

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

An Explicit Undecidable Language

Last time:

Theorem: Let X be any set. Then the power set $P(X)$ does **not** have the same size as X.

1) Assume, for the sake of contradiction, that there is a bijection $f: X \to P(X)$
2) "Flip the diagonal" to construct a set $S \in P(X)$ such that $f(x) \neq S$ for every $x \in X$

" $\{x \mid x \notin f(x)\}$

x \ y	Is $x_1 \in f(x)$?	Is $x_2 \in f(x)$?	...
x_1	~~Y~~ N	N	Y
x_2	N	~~Y~~ N	Y
x_3	N	N	~~N~~ Y
⋮			

3) Conclude that f is not onto, contradicting assumption that f is a bijection

Specializing the proof

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

1) Assume, for the sake of contradiction, that $L: X \to P(\{0,1\}^*)$ is onto

   Mapping from TM to the language it recognizes

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 L is not onto, a contradiction

An explicit undecidable language

TM M					
M_1					
M_2					
M_3					
M_4					
$\vdots$					

Why is it possible to enumerate all TMs like this?

a) The set of all TMs is finite
b) The set of all TMs is countably infinite
c) The set of all TMs is uncountable

An explicit undecidable language

{ Y if M_2 accepts input $\langle M_3 \rangle$; N if M_2 does not accept $\langle M_3 \rangle$ }

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	Y	Y		
M_3	Y	Y	~~Y~~ N	N		
M_4	N	N	Y	~~N~~ Y		
⋮					⋱	
D						~~Y~~ N, ~~N~~ Y

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

Claim: UD is undecidable

Assume for contradiction $\exists$ TM D deciding UD

Case 1: If D accepts $\langle D \rangle$, then by definition of UD, $\langle D \rangle \notin UD$ ✗

Case 2: If D does not accept $\langle D \rangle$, then by definition of UD, $\langle D \rangle \in UD$ ✗

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 TM D decides UD

Either:

1) D accepts $\langle D \rangle \Rightarrow \langle D \rangle \notin UD$ (by def. of UD)

   $\Rightarrow D$ does the wrong thing on input $\langle D \rangle$ ⨳

2) D does not accept $\langle D \rangle \Rightarrow \langle D \rangle \in UD$ (by def. of UD)

   $\Rightarrow D$ does the wrong thing. ⨳

A more useful undecidable language

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

Theorem: A_{TM} is undecidable

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

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

(either M rejects w or M loops on w)

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

"Reduction"

A more useful undecidable language

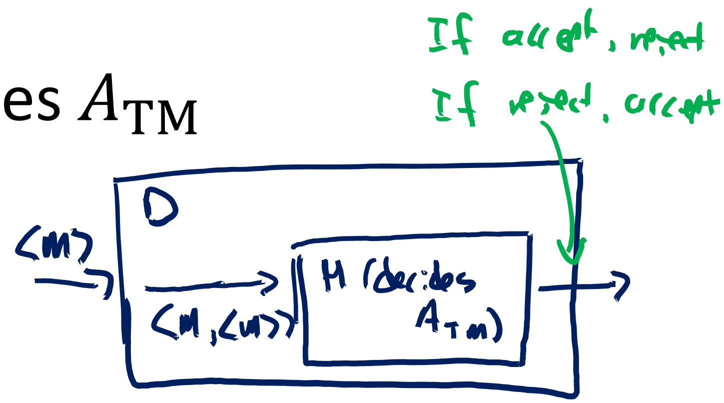
$UD = \{\langle M \rangle \mid$ TM M does not accept on input $\langle M \rangle\}$

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

Proof (continued):

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

Case 1: If $\langle M \rangle \in UD \Rightarrow M$ does not accept $\langle M \rangle \Rightarrow \langle M, \langle M \rangle\rangle \notin A_{TM}$
$\Rightarrow H$ rejects $\Rightarrow D$ accepts

Case 2: If $\langle M \rangle \notin UD \Rightarrow M$ accepts $\langle M \rangle \Rightarrow \langle M, \langle M \rangle\rangle \in A_{TM}$
$\Rightarrow H$ accepts $\Rightarrow D$ rejects

...but this language is undecidable

Unrecognizable Languages

A_{TM} is undecidable

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

A_{TM} is recognizable (by UTM)

Proof: $\Rightarrow$ |

L is decidable $\Rightarrow$ L is recognizable

L is decidable $\Rightarrow$ $\overline{L}$ is decidable (closure of decidable langs. under complement)

$\Rightarrow$ $\overline{L}$ is recognizable

Application: A_{TM} is "co-unrecognizable" meaning $\overline{A_{TM}}$ is unrecognizable.

Proof. By Thm, L is undecidable $\Leftrightarrow$ at least one of L, $\overline{L}$ unrecognizable

A_{TM} undecidable $\Rightarrow$ either ~~A_{TM}~~ or $\overline{A_{TM}}$ unrecognizable

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

Unrecognizable Languages

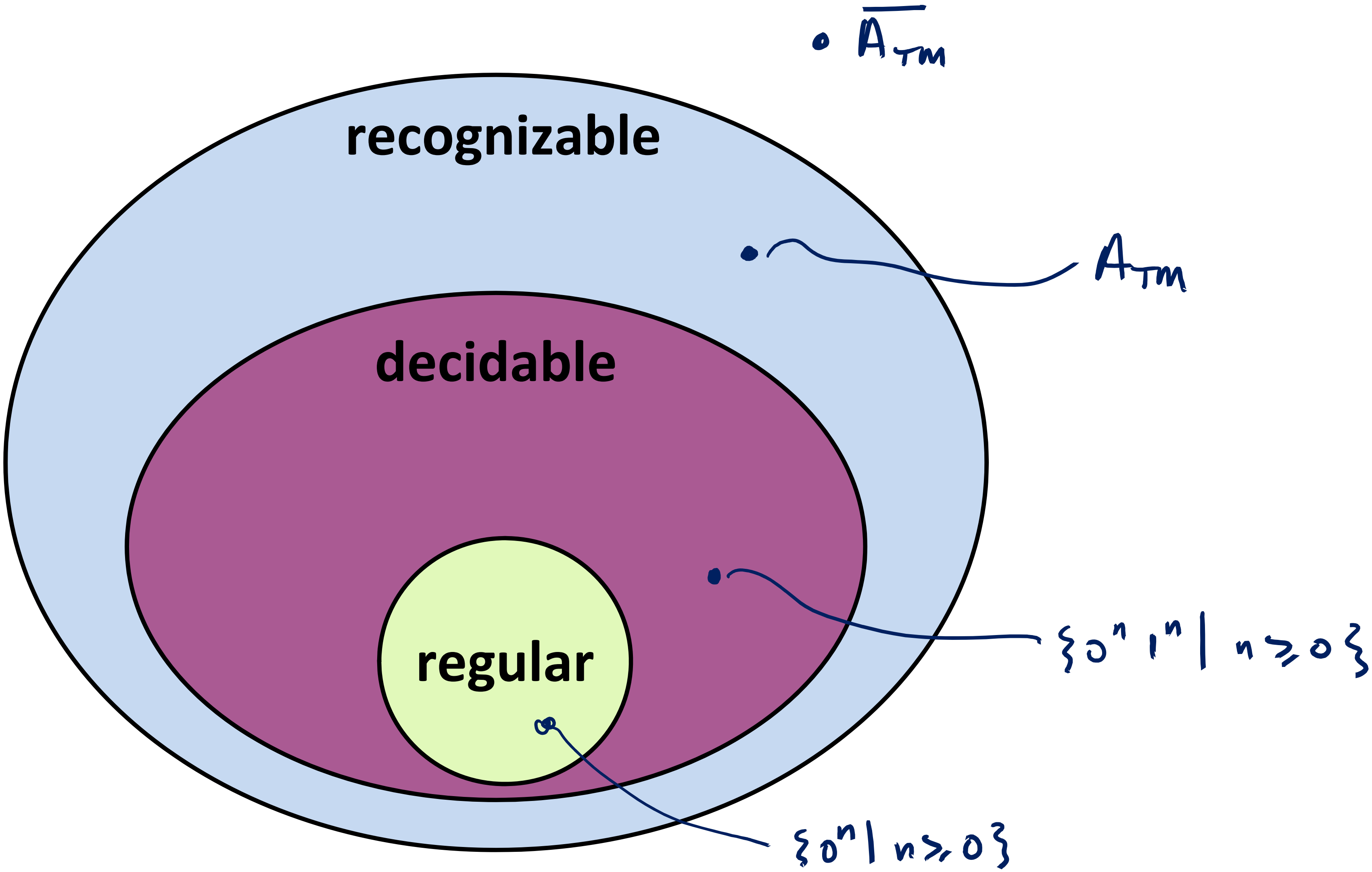
Theorem: A language L is decidable if and only if L and $\bar{L}$ are both Turing-recognizable.

Proof: $\Leftarrow$ | Suppose L is recognized by TM M
$\bar{L}$ is recognized by TM N

Goal: Construct a decider V for L (using M and N)

V = On input w:
Repeat the following forever:
1. Run M for one step on w
2. Run N for one step on w
3. If M accepts, accept; if N accepts, reject."

Classes of Languages



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.

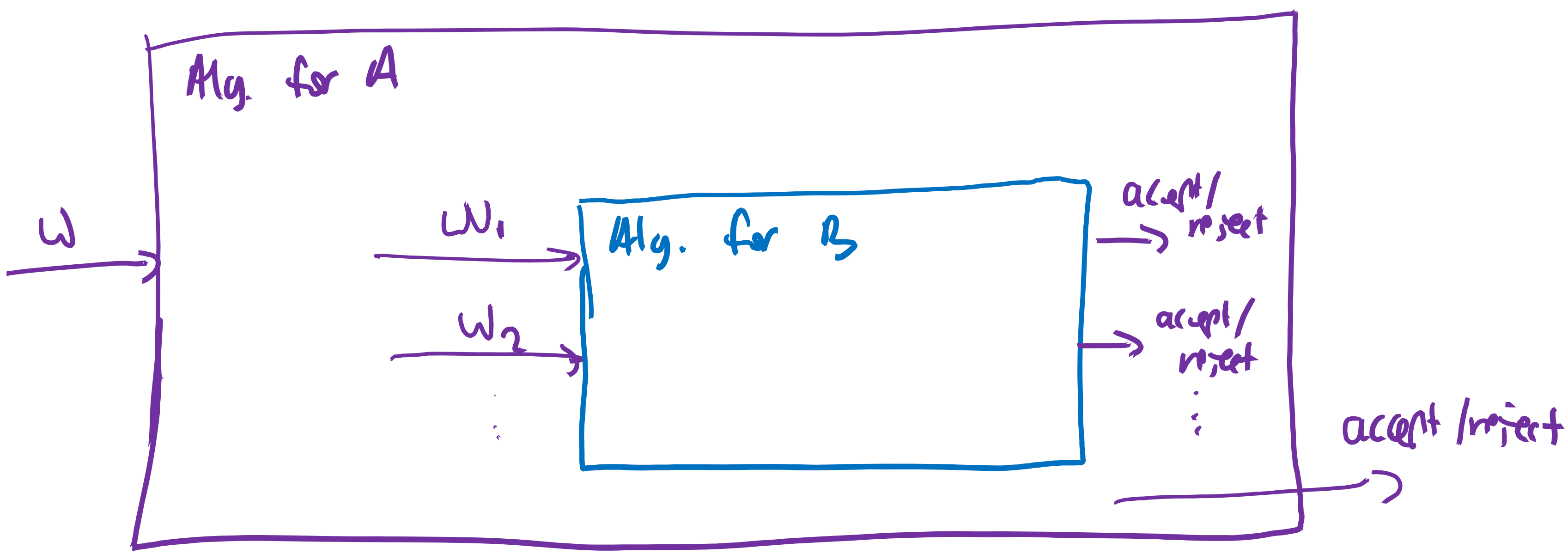
"Now we've reduced the problem to one we've already solved." (Please laugh)

Reductions

eating coconut from tree 1 — eating a coconut from tree 2

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"



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"

If A reduces to B, and B is decidable, what can we say about A?

a) A is decidable

b) A is undecidable

c) A might be either decidable or undecidable

Two uses of reductions

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

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

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

Theorem: EQ_{DFA} is decidable

Proof: The following TM decides EQ_{DFA}

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

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

Reduction

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

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

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:

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

M(x)

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."

binary for 5

Is $\langle M, 101\rangle \in HALT_{\text{TM}}$?

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 $y = 1, 2, 3, \dots$:
If $y^2 = x$, accept. Else, continue."

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

Halting Problem

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

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

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

On input $\langle M, w\rangle$: Input to A_{TM}

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.

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