BU CS 332 – Theory of Computation

Lecture 15:

- Undecidability
- Reductions

Reading:

Sipser Ch. 4, 5.1

Ran Canetti

October 29, 2020

An Undecidable Language

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

Theorem: A_{TM} is undecidable

An 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}:

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

An Undecidable Language

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

Theorem: A_{TM} is undecidable

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

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

Define

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

Consider $H(\langle \overline{H}, w\rangle)$

An 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}:

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

Define

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

Consider $H(\langle \overline{H}, w\rangle)$: Has to run forever…

➔ H is not a decider.

An unrecognizable Language

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

($L \in \boldsymbol{R}$ if and only if both $L \in \boldsymbol{RE}$ and $\overline{L} \in \boldsymbol{RE}$)

$R = RE \cap \text{co-}RE$

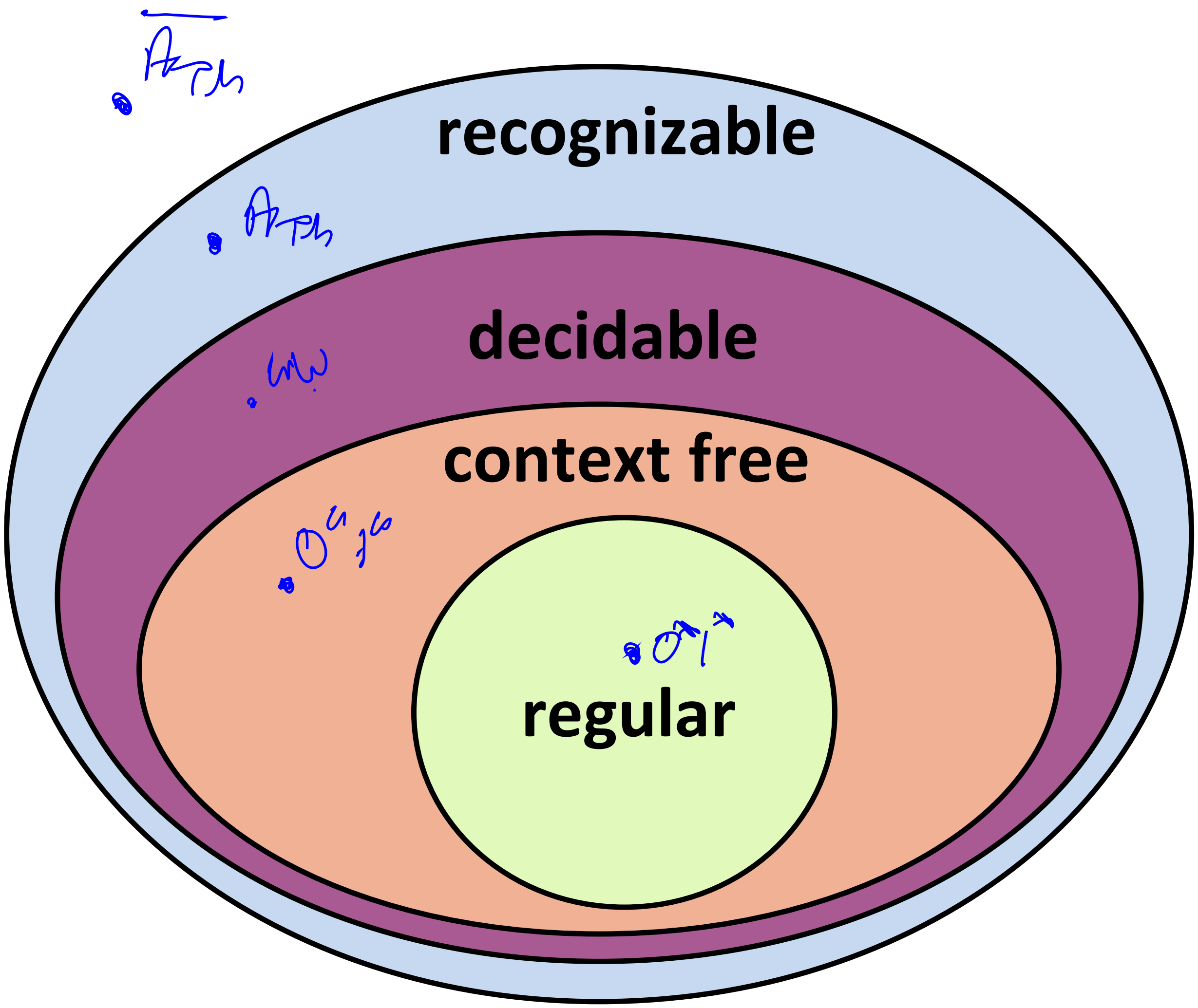
Corollary: If L is Turing-recognizable and undecidable then $\overline{L}$ is not Turing-recognizable.

(If $L \in \boldsymbol{RE}$ and $L \notin \boldsymbol{R}$ then $\overline{L} \notin \boldsymbol{RE}$)

$\overline{A_{TM}}$ is not T-recognizable

$\overline{A_{TM}} \notin RE$

Classes of Languages: updated view



A specific unrecognizable Language

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

Corollary: If L is Turing-recognizable and undecidable then $\overline{L}$ is not Turing-recognizable.

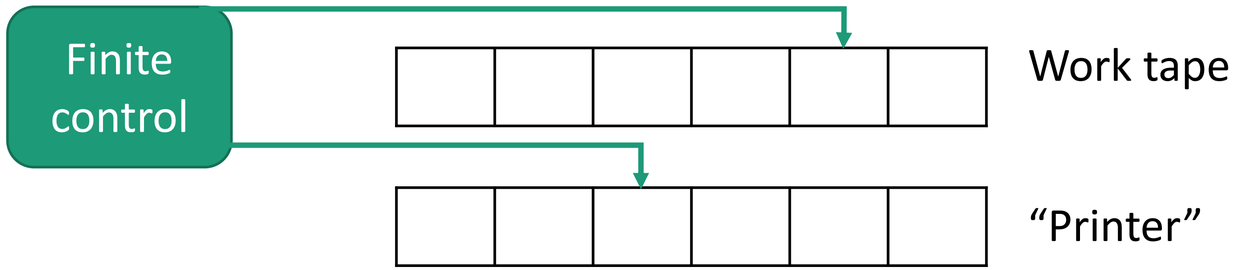
Define:

$\boldsymbol{R}$ = decidable languages

$\boldsymbol{RE}$ = Turing-recognizable languages

co$\boldsymbol{RE}$ $= \{\mathrm{L} \mid \overline{L}$ is Turing recognizable$\}$

Enumerators



- Starts with two blank tapes
- Prints strings to printer

$L(E) = \{\text{strings eventually printed by } E\}$

- May never terminate (even if language is finite)
- May print the same string many times

Enumerable = Turing-Recognizable = RE

Theorem: A language is Turing-recognizable $\Leftrightarrow$ some enumerator enumerates it

Reductions

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”

to solve $x \stackrel{?}{\in} A$

$x \xrightarrow{T} \boxed{x' \stackrel{?}{\in} B}$

y/n

12hrs

Two uses of reductions

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

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

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}

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, accept. If H rejects, reject.

Claim: D decides
$SA_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM that accepts on input } \langle M \rangle\}$

Two uses of reductions

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

Proof template:

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

Halting Problem

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

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 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, simulate M on w
4. If M accepts, accept. Otherwise, reject

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

Proof of Validity of reduction:

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

we know that $M(w)$ stops and accepts

$\Rightarrow H(\langle M, w \rangle) = accept$

$\Rightarrow$ algorithm accepts.

$\langle M, w \rangle \notin A_{TM}$: two cases,

case 1: M stops and rejects w.

$\Rightarrow$ algorithm will run $M(w)$ and will reject.

case 2: $M(w)$ never stops. $\Rightarrow H$ will reject $\langle M, w \rangle$ and our algorithm reject

Empty language 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 for A_{TM} as follows:

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

1. Construct a TM M' as follows:

   $M'(x)$:
   run M on w. if accept then accept.
   if rejects then reject.

   $M(0) = $ acc
   $M(1) = $ rej

2. Run R on input $\langle M' \rangle$
3. If R rejects, accept. Otherwise, reject

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

Context-free language testing for TMs

$$CFL_{\text{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is context} - \text{free}\}$$

Theorem: CFL_{TM} is undecidable

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

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

1. Construct a TM M' as follows:

$M'(X)$:
run $M(w)$. if reject, then reject. if accept; then accept X if $x = ww$, else reject.

if $M(w)$ = accept then $L(M') \in CFL$; else $L(M') \notin CFL$

2. Run R on input $\langle M' \rangle$
3. If R accepts, accept. Otherwise, reject

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

Context-free language testing for TMs

$$CFL_{\text{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is context} - \text{free}\}$$

Theorem: CFL_{TM} is undecidable

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

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

1. Construct a TM M' as follows:

   M' = "On input x,

   1. If $x \in \{0^n 1^n 2^n \mid n \geq 0\}$, accept
   2. Run TM M on input w
   3. If M accepts, accept."

2. Run R on input $\langle M'\rangle$
3. If R accepts, accept. Otherwise, reject

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