Mapping Reducibility

Wednesday, April 7, 2021
Announcements

- HW 8 due Sun 4/11 11:59pm EST

- HW 9 out
  - Due Sun 4/18 11:59pm EST
  - Ch5 material (starting today)
Last time: “Reduced” $A_{TM}$ to $HALT_{TM}$

Thm: $HALT_{TM}$ is undecidable

Proof, by contradiction:

• Assume $HALT_{TM}$ has decider $R$; use to create $A_{TM}$ decider:

  $S =$ “On input $\langle M, w \rangle$, an encoding of a TM $M$ and a string $w$:

  1. Run TM $R$ on input $\langle M, w \rangle$.
  2. If $R$ rejects, reject.
  3. If $R$ accepts, simulate $M$ on $w$ until it halts.
  4. If $M$ has accepted, accept; if $M$ has rejected, reject.”

• Contradiction: $A_{TM}$ is undecidable and has no decider!

Today: Formalize “reduction” and “reducibilty”
Last time: \( \text{REGULAR}_{\text{TM}} \) is undecidable

\[
\text{REGULAR}_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language} \}
\]

Proof, by contradiction:

• Assume \( \text{REGULAR}_{\text{TM}} \) has decider \( R \); use to create \( \text{A}_{\text{TM}} \) decider:

\[ S = \text{"On input } \langle M, w \rangle \text{, an encoding of a TM } M \text{ and a string } w: } \]

• First, construct \( M_2 \) (see below, and next slide)

• Run \( R \) on input \( \langle M \rangle_2 \) \hspace{1cm} \textbf{Important: } M_2 \text{ is never run; only used as an arg}

• If \( R \) accepts, accept; if \( R \) rejects, reject

Want: \( L(M_2) = \)

• regular, if \( M \) accepts \( w \)
• nonregular, if \( M \) does not accept \( w \)
Thm: $\text{REGULAR}_{TM}$ is undecidable (continued)

$\text{REGULAR}_{TM} = \{ \langle M \rangle | M$ is a TM and $L(M)$ is a regular language $\}$

$M_2 = \text{"On input } x:\n1. \text{ If } x \text{ has the form } 0^n1^n, \text{ accept.}\n2. \text{ If } x \text{ does not have this form, run } M \text{ on input } w \text{ and accept if } M \text{ accepts } w.\"$

- If $M$ does not accept $w$, $M_2$ accepts all strings (regular lang)
- if $M$ accepts $w$, $M_2$ accepts this non-regular lang
- Always accept strings $0^n1^n$
- $L(M_2) = \text{nonregular, so far}$
- If $M$ accepts $w$, accept everything else, so $L(M_2) = \Sigma^* = \text{regular}$

Want: $L(M_2) =$
- regular, if $M$ accepts $w$
- nonregular, if $M$ does not accept $w$
Reducing to non-\(A_{\text{TM}}\) language

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

**Thm:** \( EQ_{\text{TM}} \) is undecidable

**Proof, by contradiction:**

- Assume \( EQ_{\text{TM}} \) has decider \( R \); use to create \( A_{\text{TM}} \) decider.

\[ E_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \]

\( S \) = “On input \( \langle M \rangle \), where \( M \) is a TM:

1. Run \( R \) on input \( \langle M, M_1 \rangle \), where \( M_1 \) is a TM that rejects all inputs.

2. If \( R \) accepts, accept; if \( R \) rejects, reject.”
Reducing to non-$A_{TM}$ language

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

**Thm:** $EQ_{TM}$ is undecidable

**Proof, by contradiction:**

- Assume $EQ_{TM}$ has decider $R$; use to create $A_{TM}$ decider:

  $S =$ “On input $\langle M \rangle$, where $M$ is a TM:
  
  1. Run $R$ on input $\langle M, M_1 \rangle$, where $M_1$ is a TM that rejects all inputs.
  2. If $R$ accepts, accept; if $R$ rejects, reject.”

  

- **Contradiction:** $E_{TM}$ is undecidable!
Summary

- \( A_{\text{DFA}} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \} \) Decidable
- \( A_{\text{CFG}} = \{ \langle G, w \rangle | G \text{ is a CFG that generates string } w \} \) Decidable
- \( A_{\text{TM}} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ accepts } w \} \) Undecidable
- \( E_{\text{DFA}} = \{ \langle A \rangle | A \text{ is a DFA and } L(A) = \emptyset \} \) Decidable
- \( E_{\text{CFG}} = \{ \langle G \rangle | G \text{ is a CFG and } L(G) = \emptyset \} \) Decidable
- \( E_{\text{TM}} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \} \) Undecidable
- \( E_{\text{DFA}} = \{ \langle A, B \rangle | A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \) Decidable
- \( E_{\text{CFG}} = \{ \langle G, H \rangle | G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \) Undecidable
- \( E_{\text{TM}} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \) Undecidable

Observation: Can we decide anything about Turing Machines, i.e., about programs?
Can’t decide anything about TMs?

- **REGULAR_{TM} = \{<M> | M is a TM and L(M) is a regular language\}**  
  Undecidable

- **CONTEXTFREE_{TM} = \{<M> | M is a TM and L(M) is a CFL\}**  
  Undecidable

- **DECIDABLE_{TM} = \{<M> | M is a TM and L(M) is a decidable language\}**  
  Undecidable

- **FINITE_{TM} = \{<M> | M is a TM and L(M) is a finite language\}**  
  Undecidable

- ...  
  Undecidable:  
  **ANYTHING_{TM} = \{<M> | M is a TM and “something something” about L(M)\}**  
  Rice’s Theorem
Today: Computable Functions

• Needed to formalize the notion of “reducibility”
Flashback: $A_{\text{NFA}}$ is a decidable language

$A_{\text{NFA}} = \{\langle B, w \rangle | \; B \text{ is an NFA that accepts input string } w \}$

Decider (i.e., “run” function) for $A_{\text{NFA}}$:

$N =$ “On input $\langle B, w \rangle$, where $B$ is an NFA and $w$ is a string:

1. Convert NFA $B$ to an equivalent DFA $C$, using the procedure for this conversion given in Theorem 1.39.
2. Run TM $M$ on input $\langle C, w \rangle$.
3. If $M$ accepts, accept; otherwise, reject.”

More generally, we’ve been saying “programs = TMs”, but programs do more than accept/reject?

We said this NFA -> DFA algorithm is a TM, but it doesn’t accept/reject?
Computable Functions

• A TM that, instead of accept/reject, “outputs” final tape contents

**Definition 5.17**

A function $f : \Sigma^* \rightarrow \Sigma^*$ is a *computable function* if some Turing machine $M$, on every input $w$, halts with just $f(w)$ on its tape.

• **Example 1:** All arithmetic operations

• **Example 2:** Converting between machines, like DFA $\rightarrow$ NFA
  • E.g., adding states, changing transitions, wrapping TM in TM, etc.
Mapping Reducibility

**Definition 5.20**

Language $A$ is **mapping reducible** to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

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

The function $f$ is called the **reduction** from $A$ to $B$.

**Definition 5.17**

A function $f : \Sigma^* \rightarrow \Sigma^*$ is a **computable function** if some Turing machine $M$, on every input $w$, halts with just $f(w)$ on its tape.
Thm: $A_{TM}$ is mapping reducible to $HALT_{TM}$

- To show: $A_{TM} \leq_m HALT_{TM}$
- Want: computable fn $f : \langle M, w \rangle \rightarrow \langle M', w' \rangle$ where:

  $\langle M, w \rangle \in A_{TM}$ if and only if $\langle M', w' \rangle \in HALT_{TM}$

The following machine $F$ computes a reduction $f$.

- $F =$ “On input $\langle M, w \rangle$:
  1. Construct the following machine $M'$.
      $M'$ = “On input $x$:
          1. Run $M$ on $x$.
          2. If $M$ accepts, accept.
          3. If $M$ rejects, enter a loop.”
  2. Output $\langle M', w \rangle$.”

Language $A$ is mapping reducible to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

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

The function $f$ is called the reduction from $A$ to $B$.
How is mapping reducibility useful?
**Thm:** If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

**Proof**

We let $M$ be the decider for $B$ and $f$ be the reduction from $A$ to $B$. We describe a decider $N$ for $A$ as follows.

$N =$ “On input $w$:
1. Compute $f(w)$.
2. Run $M$ on input $f(w)$ and output whatever $M$ outputs.”

**Definition 5.20**

Language $A$ is *mapping reducible* to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

\[ w \in A \iff f(w) \in B. \]

The function $f$ is called the *reduction* from $A$ to $B$. 
**Coro:** If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

- Proof by contradiction.

- Assume $B$ is decidable.

- Then $A$ is decidable (by the previous thm).

- **Contradiction:** we already said $A$ is undecidable
Summary: Mapping Reducibility Theorems

- If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.
  Known

- If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.
  Unknown
Alternate Proof: The Halting Problem

$HALT_{TM}$ is undecidable

• If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

• $A_{TM} \leq_m HALT_{TM}$

• Since $A_{TM}$ is undecidable, then $HALT_{TM}$ is undecidable
Alternate Proof: $EQ_{TM}$ is undecidable

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

Flashback: proof by contradiction:

• Assume $EQ_{TM}$ has decider $R$; use to create $ET_{TM}$ decider:
  $ET_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \}$

$S = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$

1. \text{ Run } R \text{ on input } \langle M, M_1 \rangle, \text{ where } M_1 \text{ is a TM that rejects all inputs.}$
2. \text{ If } R \text{ accepts, accept; if } R \text{ rejects, reject.”}$

Alternate proof: Show: $ET_{TM} \leq_m EQ_{TM}$

• Computable fn $f: \langle M \rangle \rightarrow \langle M, M_1 \rangle$

**Definition 5.20**

Language $A$ is *mapping reducible* to language $B$, written $A \leq_m B$, if there is a computable function $f: \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

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

The function $f$ is called the *reduction* from $A$ to $B$. 
Reducing to complement: \( E_{TM} \) is undecidable

\[ E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \} \]

Proof, by contradiction:

- Assume \( E_{TM} \) has decider \( R \); use to create \( A_{TM} \) decider:

  \( S = \text{“On input } \langle M, w \rangle \text{, an encoding of a TM } M \text{ and a string } w:\text{ 1. Use the description of } M \text{ and } w \text{ to construct the TM } M_1 \text{ just described.} \text{ 2. Run } R \text{ on input } \langle M_1 \rangle. \text{ 3. If } R \text{ accepts, reject; if } R \text{ rejects, accept.”} \)

Alternate proof: computable fn: \( \langle M, w \rangle \rightarrow \langle M_1 \rangle \)

- So this only reduces \( A_{TM} \) to \( \overline{E_{TM}} \)
- Still proves \( E_{TM} \) is undecidable
  - HW9: show that undecidable langs are closed under complement
More Helpful Theorems

If $A \leq_m B$ and $B$ is Turing-recognizable, then $A$ is Turing-recognizable.

If $A \leq_m B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.

• Same proofs as:

If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.

If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.
Thm: $E_{\text{Turing}}$ is neither Turing-recognizable nor co-Turing-recognizable

$$E_{\text{Turing}} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$$

1. $E_{\text{Turing}}$ is not Turing-recognizable

$A_{\text{Turing}}$ is not Turing-recognizable, so $E_{\text{Turing}}$ is not Turing-recognizable.
Mapping Reducibility implies Mapping Red. of Complements

**Definition 5.20**

Language $A$ is *mapping reducible* to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$, $w \in A \iff f(w) \in B$.

The function $f$ is called the *reduction* from $A$ to $B$.

\[ A \leq_m B \]

implies

\[ \overline{A} \leq_m \overline{B} \]
Thm: $EQ_{\text{TM}}$ is neither Turing-recognizable nor co-Turing-recognizable.

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

1. $EQ_{\text{TM}}$ is not Turing-recognizable

Two Choices:

- Create Computable fn: $A_{\text{TM}} \rightarrow EQ_{\text{TM}}$

- Or Computable fn: $A_{\text{TM}} \rightarrow \overline{EQ_{\text{TM}}}$
Thm: $EQ_{TM}$ is not Turing-recognizable

$EQ_{TM} = \{(M_1, M_2) | M_1$ and $M_2$ are TMs and $L(M_1) = L(M_2)\}$

• Create Computable fn: $A_{TM} \rightarrow EQ_{TM}$

• $\langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle$ $M_1$ and $M_2$ are TMs and $L(M_1) \neq L(M_2)$

$F = \text{"On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ a string:}\$

1. Construct the following two machines, $M_1$ and $M_2$.
   $M_1 = \text{"On any input: }
   \begin{cases}
   1. \text{ Reject.} \\
   \end{cases}
   \text{ Accepts nothing}

   M_2 = \text{"On any input: }
   \begin{cases}
   1. \text{ Run } M \text{ on } w. \text{ If it accepts, accept.} \\
   \end{cases}
   \text{ Accepts nothing or everything}

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

• If $M$ accepts $w$, $M_1$ not equal to $M_2$
• If $M$ does not accept $w$, $M_1$ equal to $M_2$
**Thm:** $EQ_{TM}$ is neither Turing-recognizable nor co-Turing-recognizable

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

1. $EQ_{TM}$ is not Turing-recognizable
   - Create Computable fn: $A_{TM} \rightarrow EQ_{TM}$
   - Or Computable fn: $A_{TM} \rightarrow \overline{EQ_{TM}}$
   - DONE!

2. $\overline{EQ_{TM}}$ is not co-Turing-recognizable
   - (A lang is co-Turing-recog. if it is complement of Turing-recog. lang)
Prev: $EQ_{TM}$ is not Turing-recognizable

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

- Create Computable fn: $A_{TM} \rightarrow EQ_{TM}$
  - $\langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle$ $M_1$ and $M_2$ are TMs and $L(M_1) \neq L(M_2)$

$F = \text{“On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ a string:}\
  1. \text{Construct the following two machines, } M_1 \text{ and } M_2.\
     M_1 = \text{“On any input: Reject.”}\
     1. \text{Reject.”}\
     M_2 = \text{“On any input:}\
     1. \text{Run } M \text{ on } w. \text{ If it accepts, accept.”}\
  2. \text{Output } \langle M_1, M_2 \rangle.\text{”}\

DONE!
Now: $\overline{E_{TM}}$ is not Turing-recognizable

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

- Create Computable fn: $A_{TM} \rightarrow \overline{E_{TM}}$
- $\langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle$ $M_1$ and $M_2$ are TMs and $L(M_1) \neq L(M_2)$

$F = \text{“On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ a string:} \$

1. Construct the following two machines, $M_1$ and $M_2$.
   $M_1 = \text{“On any input:} \rightarrow \text{Accept.”}$
   $M_2 = \text{“On any input:} \rightarrow \text{Accept nothing or everything}$

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

\[ \overline{A_{TM}} \subseteq EQ_{TM} \]

- Turing-recognizable
- Decidable
- Context-free
- Regular
Check-in Quiz 4/7

On gradescope