Announcements

• HW 9 in
  • Due Mon 11/21 11:59pm EST

• HW 10 out
  • Due Mon 12/5 11:59pm EST
  • 2 week assignment

• No class Thursday. Happy Thanksgiving!
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 $B$.

**Step 1:** Show there is computable fn $f$ ... by creating a TM

**Step 2:** Prove the iff is true for $f$

**Step 2a:** "forward" direction ($\Rightarrow$): if $w \in A$ then $f(w) \in B$

**Step 2b:** "reverse" direction ($\Leftarrow$): if $f(w) \in B$ then $w \in A$

**Step 2b:** Equivalent (contrapositive): if $w \notin A$ then $f(w) \notin B$
Last Time: Using Mapping Reducibility

To prove decidability...

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

To prove undecidability...

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

Be careful with the direction of the reduction!
Flashback: \( E_{Q_{TM}} \) is undecidable

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

Proof by contradiction:

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

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

\[ S = \text{"On input } \langle M \rangle \text{, where } M \text{ is a TM:
1. Run } R \text{ on input } \langle M, M_1 \rangle \text{, where } M_1 \text{ is a TM that rejects all inputs.
2. If } R \text{ accepts, accept; if } R \text{ rejects, reject."} \]
Alternate Proof: $EQ_{TM}$ is undecidable

Proof by mapping reducibility: $E_{TM} \leq_m EQ_{TM}$

Step 1: create computable fn $f$, computed by TM $S$

$S = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:}$$
1. \text{Construct: } \langle M, M_1 \rangle, \text{ where } M_1 \text{ is a TM that rejects all inputs.}$$
2. \text{Output: } \langle M, M_1 \rangle$

Step 2: show iff requirements of mapping reducibility

Do for HW 10!

And use theorem ...

If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.
Flashback: $E_{TM}$ is undecidable

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

Proof, by contradiction:

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

$S =$ “On input $\langle M, w \rangle$, an encoding of a TM $M$ and a string $w$:
  
  1. Use the description of $M$ and $w$ to construct the TM $M_1$
     
     $M_1 =$ “On input $x$:
     
     1. If $x \neq w$, reject.
     2. If $x = w$, run $M$ on input $w$ and accept if $M$ halts.
  
  2. Run $R$ on input $\langle M_1 \rangle$.
  
  3. If $R$ accepts, reject; if $R$ rejects, accept.”

If $M$ accepts $w$, then $M_1$ not in $E_{TM}$! So do the opposite!

$M_1$:
- accepts $w$ if $M$ does not halt
- rejects everything else
Alternate Proof: \( E_{TM} \) is undecidable

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

**Proof**, by mapping reducibility?: \( A_{TM} \leq_m E_{TM} \)

**Step 1:** create computable fn \( f: \langle M, w \rangle \rightarrow \langle M_1 \rangle \), computed by \( S \)

\[
S = \text{“On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\n
1. Use the description of } M \text{ and } w \text{ to construct the TM } M_1
\]

\[
M_1 = \text{“On input } x: \\
1. \text{ If } x \neq w, \text{ reject.} \\
2. \text{ If } x = w, \text{ run } M \text{ on input } w \text{ and accept if } M \text{ does.”}
\]

\[
2. \text{ Output: } \langle M_1 \rangle.
3. \text{ If } R \text{ accepts, reject; if } R \text{ rejects, accept.”}
\]

**Step 2:** show iff requirements of mapping reducibility:

Do for HW 10!

- **This reduces** \( A_{TM} \) **to** \( E_{TM} \) !!
- **It’s good enough, if:** undecidable langs are **closed** under complement
Turing Unrecognizable?

Is there anything out here?

Where do these undecidable languages go?

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

\[ EQ_{CFG} = \{ \{G, H\} | G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \]

\[ EQ_{TM} = \{ \{M_1, M_2\} | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]
**Thm:** Some langs are not Turing-recognizable

**Proof:** requires 2 lemmas

- **Lemma 1:** The set of all languages is *uncountable*
  - **Proof:** Show there is a bijection with another uncountable set ...
    - ... The set of all infinite binary sequences

- **Lemma 2:** The set of all TMs is *countable*

- Therefore, some language is not recognized by a TM
  (pigeonhole principle)
Mapping a Language to a Binary Sequence

\[ \Sigma^* = \{ \varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \ldots \} \]
\[ A = \{ 0, 00, 01, 000, 001, \ldots \} \]
\[ \chi_A = 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 0 \ 1 \ 1 \ \ldots \]

Each digit represents one possible string:
- 1 if lang has that string,
- 0 otherwise
**Thm:** Some langs are not Turing-recognizable

**Proof:** requires 2 lemmas

- **Lemma 1:** The set of all languages is uncountable
  - **Proof:** Show there is a bijection with another uncountable set...
    - ... The set of all infinite binary sequences
    - Now just prove set of infinite binary sequences is uncountable (exercise)

- **Lemma 2:** The set of all TMs is countable
  - Because every TM $M$ can be encoded as a string $<M>$
  - And set of all strings is countable (from hw9)

- Therefore, some language is not recognized by a TM
Co-Turing-Recognizability

- A language is **co-Turing-recognizable** if ...
- ... it is the **complement** of a Turing-recognizable language.
Thm: Decidable $\iff$ Recognizable & co-Recognizable
Thm: Decidable $\iff$ Recognizable & co-Recognizable

$\Rightarrow$ If a language is \textbf{decidable}, then it is \textbf{recognizable} and \textbf{co-recognizable}
- Decidable $\Rightarrow$ Recognizable:
  - A decider is a recognizer (that always halts)
- Decidable $\Rightarrow$ Co-Recognizable:
  - To create co-decider from a decider ... switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

$\Leftarrow$ If a language is \textbf{recognizable} and \textbf{co-recognizable}, then it is \textbf{decidable}
**Thm:** Decidable $\Leftrightarrow$ Recognizable & co-Recognizable

$\Rightarrow$ If a language is **decidable**, then it is **recognizable** and **co-recognizable**
- Decidable $\Rightarrow$ Recognizable:
  - A decider is a recognizer (that always halts)
- Decidable $\Rightarrow$ Co-Recognizable:
  - To create co-decider from a decider ... switch reject/accept of all inputs
  - A co-decider is a co-recognizer, for same reason as above

$\Leftarrow$ If a language is **recognizable** and **co-recognizable**, then it is **decidable**
- Let $M_1 = \text{recognizer for the language}$,
- and $M_2 = \text{recognizer for its complement}$
  - **Decider** $M$:
    - Run 1 step on $M_1$,
    - Run 1 step on $M_2$,
    - Repeat, until one machine accepts. If it’s $M_1$, accept. If it’s $M_2$, reject

Termination Arg: Either $M_1$ or $M_2$ must accept and halt, so $M$ halts and is a decider
A Turing-unrecognizable language

• We’ve proved:

\[ A_{TM} \text{ is Turing-recognizable} \]

\[ A_{TM} \text{ is undecidable} \]

• So:

\[ \overline{A_{TM}} \text{ is not Turing-recognizable} \]

• Because: recognizable & co-recognizable \(\Rightarrow\) decidable
Is there anything out here?

Where do these undecidable languages go?

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

\[ EQ_{CFG} = \{ \langle G, H \rangle | G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \]

\[ EQ_{TM} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]
Using Mapping Reducibility to Prove ...

- Decidability
- Undecidability
- Recognizability
- Unrecognizability
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: $EQ_{TM}$ is neither Turing-recognizable nor co-Turing-recognizable.

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

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

If $A \leq_m B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.
Mapping Reducibility implies Mapping Red. of Complements

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: $EQ_{TM}$ is neither Turing-recognizable nor co-Turing-recognizable

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

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

Two Choices:
- Create Computable fn: $A_{TM} \rightarrow EQ_{TM}$
- Or Computable fn: $A_{TM} \rightarrow \overline{EQ_{TM}}$

And use theorem ...

If $A \leq_m B$ and $A$ is not Turing-recognizable, then $B$ is not Turing-recognizable.
**Thm:** $EQ_{TM}$ is not Turing-recognizable

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

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

  $\langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle$ $M_1$ 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 $M_2$.
     \[ M_1 = \text{"On any input:} \]
     \[ 1. \text{ Reject."} \]
     \[ M_2 = \text{"On any input:} \]
     \[ 1. \text{ Run } M \text{ on } w. \text{ If it accepts, accept."} \]
  2. Output $\langle M_1, M_2 \rangle$.

  - Accepts nothing
  - Accepts nothing or everything

**Step 2, iff:**

$\Rightarrow$ If $M$ accepts $w$, then $M_1 \neq M_2$

$\Leftarrow$ If $M$ does not accept $w$, then $M_1 = M_2$
**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: $\overline{A_{TM}} \rightarrow EQ_{TM}$
   - Or Computable fn: $A_{TM} \rightarrow \overline{EQ_{TM}}$
   - **DONE!**

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

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

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

Step 1

$\langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle$  $M_1$ 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:}$$
   1. \text{Reject.”}$

$M_2 = \text{“On any input:}$
   1. Run $M$ on $w$. If it accepts, accept.”

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

Accepts nothing

Accepts nothing or everything
Now: \( \overline{EQ_{TM}} \) is not Turing-recognizable

- Create Computable fn: \( A_{TM} \rightarrow \overline{EQ_{TM}} \)

Step 1: \( \langle M, w \rangle \rightarrow \langle M_1, M_2 \rangle \)  
\( M_1 \) 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 \( M_2 \).
   \[ M_1 = \text{“On any input:} \]
   \[ \begin{align*}
   \text{1. Accept.”} \\
   \text{M_2 = “On any input:} \\
   \text{1. Run } M \text{ on } w. \text{ If it accepts, accept.”}
   \end{align*} \]

2. Output \( \langle M_1, M_2 \rangle \).

Step 2, iff:
\( \Rightarrow \) If \( M \) accepts \( w \), then \( M_1 \equiv M_2 \)
\( \Leftarrow \) If \( M \) does not accept \( w \), then \( M_1 \not\equiv M_2 \)

DONE!
Unrecognizable Languages?

Where do these go?

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

\[ EQ_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \]

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

$A_{TM}$

Turing-recognizable

decidable

context-free

regular

$E_{TM} = \{\{M\} | M \text{ is a TM and } L(M) = \emptyset\}$

$EQ_{TM} = \{\{G, H\} | G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

Where do these go?

next
Thm: $EQ_{\text{CFG}}$ is not Turing-recognizable

Recognizable & co-recognizable implies decidable

• We’ve proved:
  $EQ_{\text{CFG}}$ is undecidable

• We now prove:
  $EQ_{\text{CFG}}$ is co-Turing recognizable

• And conclude that:
  • $EQ_{\text{CFG}}$ is not Turing recognizable
Thm: $EQ_{CFG}$ is co-Turing-recognizable

$EQ_{CFG} = \{ (G, H) | G$ and $H$ are CFGs and $L(G) = L(H) \}$

Recognizer for $\overline{EQ_{CFG}}$:

- On input $<G, H>$:
  - For every possible string $w$:
    - Accept if $w \in L(G)$ and $w \notin L(H)$
    - Or accept if $w \in L(H)$ and $w \notin L(G)$
  - Else reject

This is only a recognizer because it loops for ever when $L(G) = L(H)$
Unrecognizable Languages

Where do these go?

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

\[ EQ_{CFG} = \{ \{G, H\} \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \} \]
Unrecognizable Languages

Where do these go?

\[ E_{TM} = \{ \{M\} | M \text{ is a TM and } L(M) = \emptyset \} \]
Thm: $E_{TM}$ is not Turing-recognizable

Recognizable & co-recognizable implies decidable

• We’ve proved:
  • $E_{TM}$ is undecidable

• We now prove:
  $E_{TM}$ is co-Turing recognizable

• And then conclude that:
  • $E_{TM}$ is not Turing recognizable
Thm: $E_{TM}$ is co-Turing-recognizable

$E_{TM} = \{\langle M \rangle | \ M \text{ is a TM and } L(M) = \emptyset \}$

Recognizer for $\overline{E_{TM}}$:

Let $s_1, s_2, \ldots$ be a list of all strings in $\Sigma^*$

“On input $\langle M \rangle$, where $M$ is a TM:
1. Repeat the following for $i = 1, 2, 3, \ldots$
2. Run $M$ for $i$ steps on each input, $s_1, s_2, \ldots, s_i$.
3. If $M$ has accepted any of these, accept. Otherwise, continue.”

This is only a recognizer because it loops for ever when $L(M)$ is empty
Unrecognizable Languages

- $A_{TM}$
- Turing-recognizable
- Decidable
- Context-free
- Regular

$E_{TM}$
$E_{Q_{TM}}$
$E_{Q_{CFG}}$
Check-in Quiz 11/22
On gradescope