CS420
Chapter 5: Reducibility

Wed, November 4, 2020
HW 6/7 Questions?
HW announcements

• HW4 grades released

• HW7 released

• New partner required starting from hw7
Last time: Diagonalization of TMs

<table>
<thead>
<tr>
<th></th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>$\langle D \rangle$</th>
<th>$\ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>reject</td>
<td>accept</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_2$</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>accept</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_3$</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>reject</td>
<td>$\ldots$</td>
</tr>
<tr>
<td>$M_4$</td>
<td>accept</td>
<td>accept</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>reject</td>
<td>reject</td>
<td>accept</td>
<td>accept</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Contradiction: Needs to be both reject and accept

TM $D$ can’t exist!
Last time: $A_{TM}$ is undecidable

$$A_{TM} = \{\langle M, w \rangle | \ M \ is \ a \ TM \ and \ M \ accepts \ w \}$$

- Proof by contradiction.
- Assume $A_{TM}$ is decidable. Then there exists a decider:

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

- If $H$ exists, then we can create:

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

1. Run $H$ on input $\langle M, \langle M \rangle \rangle$.
2. Output the opposite of what $H$ outputs. That is, if $H$ accepts, reject; and if $H$ rejects, accept.”

- But $D$ does not exist! Contradiction!
Reducibility

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \]

- We proved \( A_{TM} \) undecidable by showing that its decider ...
- ... could be used to implement an impossible “D” decider.
  - Was hard to prove (diagonalization)

- In other words, we **reduced** \( A_{TM} \) to the “D” problem.

- But now we can just reduce things to \( A_{TM} \): much easier!
The Halting Problem

\[ \text{HALT}_{\text{TM}} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \} \]

- Thm: \( \text{HALT}_{\text{TM}} \) is undecidable
- Proof, by contradiction:
- Assume \( \text{HALT}_{\text{TM}} \) has decider \( R \); use to create \( A_{\text{TM}} \) decider:

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

- But \( A_{\text{TM}} \) has no decider!

\[ U = \text{“On input } \langle M, w \rangle, \text{ where } M \text{ is a TM and } w \text{ is a string:}
1. Simulate } M \text{ on input } w.
2. If } M \text{ ever enters its accept state, accept; if } M \text{ ever enters its reject state, reject.”}

Recall \( A_{\text{TM}} \)’s recognizer (which might loop):
Might need to change $M$: $E_{\text{TM}}$ is undecidable

Proof, by contradiction:
Assume $E_{\text{TM}}$ has decider $R$; use to create $A_{\text{TM}}$ decider:

First, construct $M_1$

- On input $\langle M, w \rangle$, an encoding of a TM $M$ and a string $w$:
  - Run $R$ on input $\langle M \rangle$
    - If $R$ accepts, reject (because it means $\langle M \rangle$ doesn’t accept anything)
    - if $R$ rejects, then $\text{accept} \langle M \rangle$ accepts $w$

Idea: Wrap $\langle M \rangle$ in a TM that only accepts $w$:

$M_1$ = “On input $x$:
1. If $x \neq w$, reject.
2. If $x = w$, run $M$ on input $w$ and accept if $M$ does.”
One more, modify $M$: \( R_{\text{TM}} \) is undecidable

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

• Proof, by contradiction:

• Assume \( R_{\text{TM}} \) has decider \( R \); use to create \( A_{\text{TM}} \) decider:

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

• First, construct \( M_2 \)

• Run \( R \) on input \( \langle M_2 \rangle \)

• 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: \( \text{REGULAR}_{\text{TM}} \) is undecidable (continued)

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

\( M_2 \) = “On input \( x \):

1. If \( x \) has the form \( 0^n1^n \), accept.
2. If \( x \) does not have this form, run \( M \) on input \( w \) and accept if \( M \) accepts \( w \).

Want: \( L(M_2) = \)

- regular, if \( M \) accepts \( w \)
- nonregular, if \( M \) does not accept \( w \)
Reduce to something else: \( EQ_{TM} \) is undecidable

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

- 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.”
Turing Unrecognizable?

Is there anything out here?
Thm: Some langs are not Turing-recognizable

• **Lemma 1**: The **set of all strings** in $\Sigma^*$ is **countable**
  • Count strings of length 0, then
  • Count strings of length 1, ...

• **Lemma 2**: The **set of all TMs** is **countable**
  • Because every TM $M$ can be encoded as a string $\langle M \rangle$
  • And set of all strings is countable (Lemma 1)

• **Lemma 3**: The **set of all infinite binary sequences** $B$ is **uncountable**
  • Diagonalization proof (HW7)

• **Lemma 4**: The **set of all languages** is **uncountable**
  • There is a mapping to $B$
Mapping a Lang to a Binary Sequence

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

1 if lang has this string, 0 otherwise
Thm: Some langs are not Turing-recognizable

• **Lemma 1:** The set of all strings in $\Sigma^*$ is countable
  • Count strings of length 0, then
  • Count strings of length 1, ...

• **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 (Lemma 1)

• **Lemma 3:** The set of all infinite binary sequences $B$ is uncountable
  • Diagonalization proof (HW7)

• **Lemma 4:** The set of all languages is uncountable
  • There is a mapping to $B$

• **Corollary 5:**
  • TMs countable, langs uncountable $\Rightarrow$ some langs are not Turing-recognizable
Turing Unrecognizable?

Is there anything out here?

\[ A_{TM} \]

Turing-recognizable

decidable

context-free

regular
Co-Turing-Recognizability

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

- $\Rightarrow$ If a language is decidable, then it is Turing-recognizable and co-Turing-recognizable.
  - Decidable langs $\subseteq$ recognizable langs
    - decidable $\Rightarrow$ Turing-recognizable
  - Complement closed for decidable langs
    - decidable $\Rightarrow$ co-Turing-recognizable
**Thm:** Decidable ⇔ Turing & co-Turing-recognizable

• => If a language is decidable, then it is Turing-recognizable and co-Turing-recognizable.
  • Decidable langs ⊂ recognizable langs
    • decidable \(\rightarrow\) Turing-recognizable
  • Complement closed for decidable langs
    • decidable \(\rightarrow\) co-Turing-recognizable

• <= If a language is Turing- and co-Turing recognizable, then it is decidable.
  • Let \(M_1\) = recognizer for the lang, \(M_2\) = recognizer for complement
  • Decider \(M\):
    • Run 1 step on \(M_1\), and 1 step on \(M_2\),
    • Repeat until one machine accepts. If it’s \(M_1\), accept. If it’s \(M_2\), reject
  • \(M_1\) 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:
  \[ \overline{A_{TM}} \text{ is not Turing-recognizable} \]
Is there anything out here?
Check-in Quiz 11/4
On gradescope
End of Class Survey 11/4
See course website