CS622

Reducibility by
“Modifying the TM”

Friday, April 26, 2024
Announcements

- HW 10 out
  - Due Wed 5/1 12pm noon

- 5/1: HW 11 out
- 5/8: HW 11 in, HW 12 out
- 5/8: last lecture
- 5/15: HW 12 in (no exceptions)
Summary: The Limits of Algorithms

- \(A_{\text{DFA}} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \}\) 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
- \(\text{HALT}_{\text{TM}} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w \}\) Undecidable

It's straightforward to use hypothetical \(\text{HALT}_{\text{TM}}\) decider to create \(A_{\text{TM}}\) decider.
Summary: The Limits of Algorithms

- \( A_{DFA} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \} \)  
  - Decidable
- \( A_{CFG} = \{ \langle G, w \rangle | G \text{ is a CFG that generates string } w \} \)  
  - Decidable
- \( A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ accepts } w \} \)  
  - Undecidable
- \( \text{HALT}_TM = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w \} \)  
  - Undecidable
- \( E_{DFA} = \{ \langle A \rangle | A \text{ is a DFA and } L(A) = \emptyset \} \)  
  - Decidable
- \( E_{CFG} = \{ \langle G' \rangle | G \text{ is a CFG and } L(G') = \emptyset \} \)  
  - Decidable
- \( E_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \} \)  
  - Undecidable

How can we use a hypothetical \( E_{TM} \) decider to create \( A_{TM} \) or \( \text{HALT}_TM \) decider?
Reducibility: Modifying the TM

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

**Proof**, by contradiction:

- Assume $E_{TM}$ has _decider_ $R$; use it to create _decider_ for $A_{TM}$:

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

  - 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 _???_ ($\langle M \rangle$ accepts something, but is it $w$ ???)

Let $\langle M, w \rangle$ be a string where:
- $M$ is some TM
- $w$ is some string

<table>
<thead>
<tr>
<th>String</th>
<th>$M$ on $w$</th>
<th>$R$ on $\langle M \rangle$</th>
<th>$S$ on $\langle M, w \rangle$</th>
<th>In lang $A_{TM}$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle M, w \rangle$</td>
<td>Accept</td>
<td>Reject, $L(M)=$??</td>
<td>??</td>
<td>Yes</td>
</tr>
<tr>
<td>$\langle M, w \rangle$</td>
<td>Reject</td>
<td>Accept, $L(M)=${}</td>
<td>Reject</td>
<td>No</td>
</tr>
<tr>
<td>$\langle M, w \rangle$</td>
<td>Loop</td>
<td>Accept, $L(M)=${}</td>
<td>Reject</td>
<td>No</td>
</tr>
</tbody>
</table>

Example Table for $A_{TM}$ decider $S$
Thm: $E_{TM}$ is undecidable

Proof, by contradiction:

• Assume $E_{TM}$ has decider $R$; use it to create decider for $A_{TM}$:

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

• 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 ??? (\(\langle M \rangle\) accepts something, but is it $w$???)

• Idea: Wrap $\langle M \rangle$ in a new TM that can only (maybe) accept $w$.

\[
M_1 = \text{"On input } x:\n\begin{align*}
1. & \text{If } x \neq w, \text{ reject.} \\
2. & \text{If } x = w, \text{ run } M \text{ on input } w \text{ and accept if } M \text{ does."
}\]

Input is $w$, maybe accept
1. Input not $w$, always reject
2. $M_1$ accepts $w$ if $M$ does
Reducibility: Modifying the TM

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

**Proof, by contradiction:**

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

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

<table>
<thead>
<tr>
<th>String $x$</th>
<th>$M$ on $w$</th>
<th>$M_1$ on $x$</th>
<th>In lang ${w} \cap L(M)$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td>Accept</td>
<td>Accept</td>
<td>Yes (lang = ${w}$)</td>
</tr>
<tr>
<td>$w$</td>
<td>Reject</td>
<td>Reject</td>
<td>No (lang = ${}$)</td>
</tr>
<tr>
<td>not $w$</td>
<td>-</td>
<td>Reject</td>
<td>No (lang = ${}$ or ${w}$)</td>
</tr>
</tbody>
</table>

**Idea:** Wrap $\langle M \rangle$ in a new TM that **can only** (maybe) accept $w$.

$M_1 = \text{"On input } x:\$

1. If $x \neq w$, reject.
2. If $x = w$, run $M$ on input $w$ and accept if $M$ does.

$L(M_1)$ depends on $M$ and $w$!
If $M$ accepts $w$, $L(M_1) = \{w\}$
else $L(M_1) = \{\}$
Reducibility: Modifying the TM

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

**Proof, by contradiction:**

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

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

<table>
<thead>
<tr>
<th>String $x$</th>
<th>$M$ on $w$</th>
<th>$M_1$ on $x$</th>
<th>In lang ${w} \cap L(M)$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td>Accept</td>
<td>Accept</td>
<td>Yes ($\text{lang} = {w}$)</td>
</tr>
<tr>
<td>$w$</td>
<td>Reject</td>
<td>Reject</td>
<td>No ($\text{lang} = {}$)</td>
</tr>
<tr>
<td>not $w$</td>
<td>-</td>
<td>Reject</td>
<td>No ($\text{lang} = {}$ or ${w}$)</td>
</tr>
</tbody>
</table>

**Idea:**

| String | $M$ on $w$ | $R$ on $\langle M \rangle$ | $S$ on $\langle M, w \rangle$ | In lang $A_{TM}$? |
|--------|------------|---------------------------|-----------------------------|----------------
| $\langle M, w \rangle$ | Accept | **Reject, $L(M)=??$** | ?? | Yes |
| $\langle M, w \rangle$ | Reject | Accept, $L(M)=\{}$ | Reject | No |
| $\langle M, w \rangle$ | Loop | Accept, $L(M)=\{}$ | Reject | No |

**Example Table for $A_{TM}$ decider $S$**

$L(M_1)$ depends on $M$ and $w$!
If $M$ accepts $w$, $L(M_1) = \{w\}$
else $L(M_1) = \{}$
Reducibility: Modifying the TM

Thm: $E_{TM}$ is undecidable

Proof, by contradiction:

• Assume $E_{TM}$ has decider $R$; use it to create decider for $A_{TM}$:

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

<table>
<thead>
<tr>
<th>String $x$</th>
<th>$M$ on $w$</th>
<th>$M_1$ on $x$</th>
<th>In lang ${w} \cap L(M)$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w$</td>
<td>Accept</td>
<td>Accept</td>
<td>Yes ($\text{lang} = {w}$)</td>
</tr>
<tr>
<td>$w$</td>
<td>Reject</td>
<td>Reject</td>
<td>No ($\text{lang} = {}$)</td>
</tr>
<tr>
<td>not $w$</td>
<td>-</td>
<td>Reject</td>
<td>No ($\text{lang} = {}$ or ${w}$)</td>
</tr>
</tbody>
</table>

Idea:

<table>
<thead>
<tr>
<th>String</th>
<th>$M$ on $w$</th>
<th>$R$ on $\langle M_1 \rangle$</th>
<th>$S$ on $\langle M, w \rangle$</th>
<th>In lang $A_{TM}$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle M, w \rangle$</td>
<td>Accept</td>
<td><strong>Reject, $L(M_1) = {w}$</strong></td>
<td>Accept</td>
<td>Yes</td>
</tr>
<tr>
<td>$\langle M, w \rangle$</td>
<td>Reject</td>
<td>Accept, $L(M_1) = {}$</td>
<td>Reject</td>
<td>No</td>
</tr>
<tr>
<td>$\langle M, w \rangle$</td>
<td>Loop</td>
<td>Accept, $L(M_1) = {}$</td>
<td>Reject</td>
<td>No</td>
</tr>
</tbody>
</table>

Example Table for $M_1$:

$L(M_1)$ depends on $M$ and $w$!
If $M$ accepts $w$, $L(M_1) = \{w\}$
else $L(M_1) = \{}$
Reducibility: Modifying the TM

Thm: $E_{TM}$ is undecidable

Proof, by contradiction:

• Assume $E_{TM}$ has decider $R$; use it to create decider for $A_{TM}$:

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

  1. Run $R$ on input $\langle M \rangle$
  2. If $R$ accepts, reject (because it means $\langle M \rangle$ doesn’t accept $w$)
  3. If $R$ rejects, then accept ($\langle M \rangle$ accepts $w$)

• Idea: Wrap $\langle M \rangle$ in a new TM that can only (maybe) accept $w$.

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

Undecidability Proof Technique #2

$L(M_1) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \{} & \text{else } \end{cases}$
Reducibility: Modifying the TM

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

**Proof,** by contradiction:

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

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

  This decider for $A_{TM}$ cannot exist!

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

  - First, construct $M_1$
  - Run $R$ on input $\langle M_1 \rangle$
  - If $R$ accepts, *reject* (because it means $\langle M \rangle$ doesn’t accept $w$)
  - If $R$ rejects, then *accept* ($\langle M \rangle$ accepts $w$)

  \[ \text{• Idea: Wrap } \langle M \rangle \text{ in a new TM that can only (maybe) accept } w:} \]

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

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

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

$S = “On \text{ input } \langle M \rangle, \text{ where } M \text{ 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.”
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 it to create *decider* for \( E_{TM} \):
  
  \[ = \{ \langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \} \]

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

1. 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.”

- But \( E_{TM} \) is undecidable!
Summary: Undecidability Proof Techniques

- **Proof Technique #1:**
  - Use hypothetical decider to implement impossible $A_{TM}$ decider
  - Example Proof: $\text{HALT}_{TM} = \{\langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w \}$

- **Proof Technique #2:**
  - Use hypothetical decider to implement impossible $A_{TM}$ decider
  - But first modify the input $M$
  - Example Proof: $E_{TM} = \{\langle M \rangle | M \text{ is a TM and } L(M) = \emptyset \}$

- **Proof Technique #3:**
  - Use hypothetical decider to implement non-$A_{TM}$ impossible decider
  - Example Proof: $E_{Q_{TM}} = \{\langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$
Summary: Decidability and Undecidability

1. $A_{DFA} = \{\langle B, w \rangle \mid B \text{ is a DFA that accepts input string } w \}$  
   Decidable

2. $A_{CFG} = \{\langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$  
   Decidable

3. $A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \}$  
   Undecidable

4. $E_{DFA} = \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \}$  
   Decidable

5. $E_{CFG} = \{\langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$  
   Decidable

6. $E_{TM} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) = \emptyset \}$  
   Undecidable

7. $E_{DFA} = \{\langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$  
   Decidable

8. $E_{CFG} = \{\langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$  
   Undecidable

9. $E_{TM} = \{\langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$  
   Undecidable
Also Undecidable ...

- $\text{REGULAR}_{\text{TM}} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}$
Thm: \( \text{REGULAR}_{TM} \) is undecidable

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

Proof, by contradiction:

- **Assume**: \( \text{REGULAR}_{TM} \) has decider \( R \); use it to create decider for \( A_{TM} \):
  
  \[
  S = \text{“On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:}
  \]
  
  - 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)

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

\[ M_2 = \text{“On input } x:\]
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 \( w \)

Always accept strings \( 0^n1^n \)
\( L(M_2) = \text{nonregular, so far} \)

If \( M \) accepts \( w \), accept everything else, so \( L(M_2) = \Sigma^* = \text{regular} \)

if \( M \) does not accept \( w \), \( M_2 \) accepts all strings (regular lang)

All strings

\( 0^n1^n \)

if \( M \) accepts \( w \), \( M_2 \) accepts this nonregular lang
Also Undecidable ...

• $REGULAR_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a regular language}\}$

• $CONTEXTFREE_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a CFL}\}$

• $DECIDABLE_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a decidable language}\}$

• $FINITE_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is a finite language}\}$

Seems like no algorithm can compute anything about the language of a Turing Machine, i.e., about the runtime behavior of programs ...
An Algorithm About Program Behavior?

```c
main()
{
    printf("hello, world\n");
}
```

Write a program that, given another program as its argument, returns TRUE if that argument prints "Hello, World!"

TRUE
Write a program that, given another program as its argument, returns TRUE if that argument prints "Hello, World!"

```
main()
{
    If \( x^n + y^n = z^n \), for any integer \( n > 2 \)
    printf("hello, world\n");
}
```

Fermat's Last Theorem (unknown for ~350 years, solved in 1990s)
Also Undecidable ...

- $\text{REGULAR}_{\text{TM}} = \{<M> | M \text{ is a TM and } L(M) \text{ is a regular language}\}$
- $\text{CONTEXTFREE}_{\text{TM}} = \{<M> | M \text{ is a TM and } L(M) \text{ is a CFL}\}$
- $\text{DECIDABLE}_{\text{TM}} = \{<M> | M \text{ is a TM and } L(M) \text{ is a decidable language}\}$
- $\text{FINITE}_{\text{TM}} = \{<M> | M \text{ is a TM and } L(M) \text{ is a finite language}\}$
- ...
- $\text{ANYTHING}_{\text{TM}} = \{<M> | M \text{ is a TM and “… anything …” about } L(M)\}$

Rice’s Theorem

Seems like no algorithm can compute anything about the language of a Turing Machine, i.e., about the runtime behavior of programs...
Rice’s Theorem: \( \text{ANYTHING}_{\text{TM}} \) is Undecidable

\[ \text{ANYTHING}_{\text{TM}} = \{<M> | \text{M is a TM and ... anything ... about } L(M) \} \]

• “... Anything ...”, more precisely:
  For any \( M_1, M_2 \),
  • if \( L(M_1) = L(M_2) \)
  • then \( M_1 \in \text{ANYTHING}_{\text{TM}} \iff M_2 \in \text{ANYTHING}_{\text{TM}} \)

• Also, “... Anything ...” must be “non-trivial”:
  • \( \text{ANYTHING}_{\text{TM}} \neq \{\} \)
  • \( \text{ANYTHING}_{\text{TM}} \neq \text{set of all TMs} \)
Rice’s Theorem: \( \text{ANYTHING}_{\text{TM}} \) is Undecidable

\( \text{ANYTHING}_{\text{TM}} = \{<M> | M \text{ is a TM and ... anything ... about } L(M)\} \)

Proof by contradiction

- **Assume** some language satisfying \( \text{ANYTHING}_{\text{TM}} \) has a decider \( R \).
  - Since \( \text{ANYTHING}_{\text{TM}} \) is non-trivial, then there exists \( M_{\text{ANY}} \in \text{ANYTHING}_{\text{TM}} \)
  - Where \( R \) accepts \( M_{\text{ANY}} \)
- **Use** \( R \) to create decider for \( A_{\text{TM}} \):

  On input \( <M, w> \):
  - **Create** \( M_w \):
    - \( M_w = \) on input \( x \):
      - Run \( M \) on \( w \)
      - If \( M \) rejects \( w \): reject \( x \)
      - If \( M \) accepts \( w \): Run \( M_{\text{ANY}} \) on \( x \) and accept if it accepts, else reject
  - **Run** \( R \) on \( M_w \)
    - If it accepts, then \( M_w = M_{\text{ANY}} \), so \( M \) accepts \( w \), so accept
    - Else reject

**Wait!** What if the TM that accepts nothing is in \( \text{ANYTHING}_{\text{TM}} \)?

These two cases must be different, (so \( R \) can distinguish when \( M \) accepts \( w \))

Proof still works! Just use the complement of \( \text{ANYTHING}_{\text{TM}} \) instead!
Rice’s Theorem Implication

\{<M> | M is a TM that installs malware\}

Undecidable!
(by Rice’s Theorem)