CS420
Reducibility

Monday, March 28, 2022

```c
DEFINE DOESITHALT(PROGRAM):
{
    RETURN TRUE;
}
```

THE BIG PICTURE SOLUTION TO THE HALTING PROBLEM
Announcements

• HW 7 in
  • Due Sun 3/27 11:59pm

• HW 8 out
  • Due Sun 4/3 11:59pm
Last Time: Undecidability Proofs

- We proved \( A_{TM} = \{\langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \) undecidable ...
- ... by contradiction:
  - Use hypothetical \( A_{TM} \) decider to create an impossible decider “\( D \)”!

- Step # 1: coming up with “\( D \)” --- hard!
  - Need to invent diagonalization

- Step # 2: “reduce” \( A_{TM} \) to the “\( D \)” problem --- easier!

- From now on: undecidability proofs only need to do step # 2!
  - And we now have two “impossible” problems to choose from
Last Time: The Halting Problem

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

Thm: \text{HALT}_{TM} is undecidable

Proof, by contradiction:

- Assume: \text{HALT}_{TM} has decider \( R \); use it to create decider for \( A_{TM} \):
  \[ A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ accepts } w \} \]

- ...

- But \( A_{TM} \) is undecidable and has no decider!
Last Time: The Halting Problem

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

Thm: \( \text{HALT}_{TM} \) is undecidable

Proof, by contradiction: 

- **Assume:** \( \text{HALT}_{TM} \) has **decider** \( R \); use it to create decider for \( A_{TM} \):

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

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

1. Run TM \( R \) on input \( \langle M, w \rangle \).
2. If \( R \) rejects, reject.  
   - This means \( M \) loops on input \( w \)
3. If \( R \) accepts, simulate \( M \) on \( w \) until it halts.  
   - This step always halts
4. If \( M \) has accepted, accept; if \( M \) has rejected, reject.”

Termination argument:

**Step 1:** \( R \) is a decider so always halts

**Step 3:** \( M \) always halts because \( R \) said so
Last Time: The Halting Problem

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

Thm: \( \text{HALT}_{TM} \) is undecidable

Proof, by contradiction:

- **Assume:** \( \text{HALT}_{TM} \) has *decider* \( R \); use it to create decider for \( A_{TM} \):

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

  \[ S = \text{“On input } \langle M, w \rangle \text{, an encoding of a TM } M \text{ and a string } w:\n  1. \text{ Run TM } R \text{ on input } \langle M, w \rangle. 
  2. \text{ If } R \text{ rejects, reject.}
  3. \text{ If } R \text{ accepts, simulate } M \text{ on } w \text{ until it halts.}
  4. \text{ If } M \text{ has accepted, accept; if } M \text{ has rejected, reject.”} \]

- But \( A_{TM} \) is undecidable! i.e., this decider does not exist!
  - So \( \text{HALT}_{TM} \) is also undecidable!
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
- \( HALT_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w \} \) 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
Reducibility: Modifying the TM

Thm: \( E_{TM} \) is undecidable

Proof, by contradiction:

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

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

  - 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 \( w \))

- Idea: Wrap \( \langle M \rangle \) in a new TM that can only 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.

Input is \( w \), maybe accept

Input not \( w \), always reject

\( M_1 \) 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 $A_{TM}$:

  $S =$ “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 $w$)
  - If $R$ rejects, then $\langle M \rangle$ accepts $w$

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

  $M_1 =$ “On input $x$:
  1. If $x \neq w$, reject.
  2. If $x = w$, run $M$ on input $w$ and accept if $M$ does.”
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
- $E_{DFA} = \{ \langle A \rangle | \ A \text{ is a DFA and } L(A) = \emptyset \}$ 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
- $EQ_{DFA} = \{ \langle A, B \rangle | \ A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$ Decidable
- $EQ_{CFG} = \{ \langle G, H \rangle | \ G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$ Undecidable
- $EQ_{TM} = \{ \langle M_1, M_2 \rangle | \ M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$ Undecidable
Reduce to something else: $EQ_{TM}$ is undecidable

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

Proof, by contradiction:

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

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

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

  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 $E_{TM}$:

  $S = \text{"On input } \langle M \rangle, \text{ where } M \text{ is a TM:}
  \begin{enumerate}
  \item Run $R$ on input $\langle M, M_1 \rangle$, where $M_1$ is a TM that rejects all inputs.
  \item If $R$ accepts, accept; if $R$ rejects, reject."
  \end{enumerate}$

• But $E_{TM}$ is undecidable!
Summary: Undecidability Proof Techniques

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

- **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: $EQ_{TM} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$
Summary: Decidability and Undecidability

- \( A_{\text{DFA}} = \{ \langle B, w \rangle | \ B \text{ is a DFA that accepts input string } w \} \)  
  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

- \( EQ_{\text{DFA}} = \{ \langle A, B \rangle | \ A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)  
  Decidable

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

- \( EQ_{\text{TM}} = \{ \langle M_1, M_2 \rangle | \ M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \)  
  Undecidable
Also Undecidable ...

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

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

Proof, by contradiction:

- **Assume:** \( REGULAR_{TM} \) 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 \( w \):
  
  - 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: $REGULAR_{TM}$ is undecidable (continued)

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

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

- Always accept strings $0^n1^n$; $L(M_2)$ = nonregular, so far
- If $M$ accepts $w$, accept everything else, so $L(M_2) = \Sigma^* = \text{regular}$
- Want: $L(M_2) =$\begin{itemize}
  \item regular, if $M$ accepts $w$\item nonregular, if $M$ does not accept $w$\end{itemize}
Also Undecidable ...

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

- $\text{CONTEXTFREE}_\text{TM} = \{ <M> \mid M \text{ is a TM and } L(M) \text{ is a CFL} \}$

- $\text{DECIDABLE}_\text{TM} = \{ <M> \mid M \text{ is a TM and } L(M) \text{ is a decidable language} \}$

- $\text{FINITE}_\text{TM} = \{ <M> \mid M \text{ is a TM and } L(M) \text{ is a finite language} \}$

Seems like no algorithm can compute anything about language of TMs, i.e., about 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!"
Also Undecidable ...

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

- $\text{CONTEXTFREE}_\text{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a CFL}\}$

- $\text{DECIDABLE}_\text{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a decidable language}\}$

- $\text{FINITE}_\text{TM} = \{<M> \mid M \text{ is a TM and } L(M) \text{ is a finite language}\}$

- ...

- $\text{ANYTHING}_\text{TM} = \{<M> \mid M \text{ is a TM and “… anything …” about } L(M)\}$

 Seems like no algorithm can compute anything about Turing Machines, i.e., about programs!

 Rice’s Theorem
Rice’s Theorem: \( ANYTHING_{\text{TM}} \) is Undecidable

\[
ANYTHING_{\text{TM}} = \{<M> \mid M \text{ is a TM and … anything … about } L(M)\}
\]

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

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

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

Proof by contradiction

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

   **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

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

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

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

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

Undecidable!
(by Rice’s Theorem)
\[ A_{DFA} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \} \]
\[ A_{CFG} = \{ \langle G, w \rangle | G \text{ is a CFG that generates string } w \} \]
\[ A_{TM} = \{ \langle M, w \rangle | M \text{ is a TM and } M \text{ accepts } w \} \]

Decidable
Decidable
Undecidable

• In hindsight, of course a restricted TM (a **decider**) shouldn’t be able to simulate unrestricted TM (a **recognizer**)

• But could a restricted TM simulate an even more restricted TM?
  • Next time
Check-in Quiz 3/28
On gradescope