CS420
Chapter 5: Reducibility

Monday, April 5, 2021
Announcements

• HW 7 due date past

• HW 8 due Sun 4/11 11:59pm EST

• HW9 out soon
  • Due Sun 4/18 11:59pm EST
  • Ch5-6 material (starting Wed)

```c
#define DOESITHALT_program:
{
  return true;
}
```

The big picture solution to the halting problem.
## Last time: Diagonalization of TMs

### Diagonal: Result of giving a TM itself as input

| $M_1$ | $M_2$ | $M_3$ | $M_4$ | ... | $D$ | ...
|-------|-------|-------|-------|-----|-----|-----
| accept | reject | accept | reject | ... | accept | ...
| accept | accept | accept | accept | ... | accept | ...
| reject | reject | reject | reject | ... | reject | ...
| accept | accept | reject | reject | ... | accept | ...

### All TMs

### Opposite

### "Opposite" machine

### Contradiction: Needs to both reject and accept

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

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

Proof, by contradiction.

• Assume \( A_{TM} \) is decidable. Then there exists a decider:

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

• If \( H \) exists, then we can create \( D \):

\[
D = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \\
1. \text{ Run } H \text{ on input } \langle M, \langle M \rangle \rangle. \\
2. \text{ Output the opposite of what } H \text{ outputs. That is, if } H \text{ accepts, reject; and if } H \text{ rejects, accept.”}
\]

“Opposite” machine

Result of giving a TM itself as input
Last time: $A_{TM}$ is undecidable

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 $D$:

\[
D = \text{“On input } \langle M \rangle, \text{ where } M \text{ is a TM:} \\
1. \text{ Run } H \text{ on input } \langle M, \langle M \rangle \rangle. \\
2. \text{ Output the opposite of what } H \text{ outputs. That is, if } H \text{ accepts, reject; and if } H \text{ rejects, accept.”}
\]

• But $D$ does not exist! Therefore we have a contradiction!
Last time: Unrecognizability

• We’ve proved:
  \[ A_{TM} \text{ is Turing-recognizable} \]
  \[ A_{TM} \text{ is undecidable} \]

• And:
  **Theorem 4.22**
  A language is decidable iff it is Turing-recognizable and co-Turing-recognizable.

• So:
  \[ \overline{A_{TM}} \text{ is not Turing-recognizable} \]
Today: Easier Undecidability Proofs!

• We proved \( A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ accepts } w \} \) undecidable by ...

• ... showing that its decider could be used to implement an impossible “D” decider.

• In other words, we reduced \( A_{TM} \) to the “D” problem.
  • That was hard (needed to invent diagonalization)

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

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

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

Proof, by contradiction:

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

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

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

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

• But \( A_{TM} \) is undecidable!
  • I.e., this decider that we just created cannot exist! So \( \text{HALT}_{TM} \) is undecidable
Easier Undecidability Proofs

In general, to prove the undecidability of a language:
• Use proof by contradiction:

• Assume the language is decidable,

• Show that its decider can be used to create a decider for ...

• ... a known undecidable language ...

• ... which doesn’t have a decider!
Summary: Languages About Machines

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

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

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

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

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

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

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

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

- $EQ_{TM} = \{ \langle M_1, M_2 \rangle \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \}$  
  Undecidable
Reducibility: Modifying the TM

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

Proof, by contradiction:

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

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

  - First, construct \( M_1 \)
  - 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 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. \text{ If } x \neq w, \text{ reject.}
  
  \[
  2. \text{ If } x = w, \text{ run } M \text{ on input } w \text{ and accept if } M \text{ does.}"
  \]
Reducibility: Modifying the TM

Thm: $E_{TM}$ is undecidable

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 $w$:
  
  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 $w$:

  $M_1 = “$ 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 $M$: $\text{REGULAR}_{\text{TM}}$ 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 $A_{\text{TM}}$ 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 $\langle M \rangle_2$
  • 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)

$REGULAR_{TM} = \{ \langle M \rangle \mid M$ 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.\"

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 $w$
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 to create $A_{TM}$ decider:

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

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.”
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 = \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

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

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

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

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

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

We can’t compute anything about Turing Machines, i.e., about programs!
Also Undecidable ... 

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

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

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

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

- ... 

- $\textit{ANYTHING}_{\text{TM}} = \{<M> \mid M \text{ is a TM and “something something” about } L(M)\}$
Formalizing Reducibility, i.e., Mapping Reducibility
Flashback: $A_{\text{NFA}}$ is a decidable language

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

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.”

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:** Machine conversion algorithms, like DFA -> 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 $\langle M', w \rangle$.”

$M$ accepts $w \iff M'$ halts on $w$

$M'$ halts on $w$
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 $B$. 
**Corollary:** 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).
- So we have a contradiction.
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

$\text{HALT}_{\text{TM}}$ is undecidable

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

- $A_{\text{TM}} \leq_m \text{HALT}_{\text{TM}}$
Check-in Quiz 4/5

On gradescope