Undecidability II

Lecture 26
Example of Undecidable Language

\[ \text{SELFREJECT} = \{ \langle M \rangle \mid M \text{ rejects } \langle M \rangle \} \]

\( M \) = Turing Machine (piece of executable code)

\( \langle M \rangle \) = encoding of M as a string (source code for M)

\( \langle M \rangle \) is what you would feed to a universal TM, that would allow it to simulate M.

(e.g. TM that rejects everything. TM that rejects every description of a TM are in that language)
Showing Undecidability

To show $L$ is undecidable, reduce some undecidable language to $L$

$$SELFHALT = \{ <M> \mid M \text{ halts on } <M> \}$$

Claim: $SELFHALT$ is undecidable

More general looking problem:

$$HALT = \{ <M,w> \mid M \text{ halts on } w \}$$

Claim: $HALT$ is acceptable

Claim: $HALT$ is undecidable

The halting problem
Showing Undecidability

\[ HALT = \{ <M,w> | M \text{ halts on } w \} \]

**Claim**: \( HALT \) is undecidable

**Proof:**
Suppose (towards contradiction) that there is a TM H that decides HALT. Reduce from SELFHALT.
NEVERACCEPT = \{ <M> \mid \text{ACCEPT}(M) = \emptyset \}

(is a TM useless or not?)

Claim: NEVERACCEPT is undecidable
How many Turing Machines?

- Fix a TM M and an input w.
- Build a new TM M’ with the following behavior:
  - M’ accepts its input iff M accepts w. (toss input out the window)
- Pseudocode:
  ```
  M’(x) = Run M(w)
  ```
Fix a TM $M$ and an input $w$.

Build a new TM $M'$ with the following behavior:

- $M'$ accepts its input iff $M$ accepts $w$. (toss input out the window)

Pseudocode:

$M'(x)$

Run $M(w)$
How many Turing Machines?

• Fix a TM $M$ and an input $w$.

• Build a new TM $M'$ with the following behavior:

• $M'$ accepts its input iff $M$ accepts $w$. (toss input out the window)

• Pseudocode:

  $M'(x)$

  Run $M(w)$

  $w$ hardcoded and $M$ hardcoded in $M'$

$M'$

$w$

acc

rej

$M$
• Build M’?

Write a program

**Input** $<M,w>$: $M$ - Turing Machine,

$w$ - string

**Output** $<M'>$: $M'$ - Turing Machine,

s.t. for any string $x$, $M'$ accepts $x$ iff $M$ accepts $w$.

• could produce $M'$ ourselves (write pseudocode).

• So far, when we talk about reduction, WE are doing the reduction

Now, we need to describe how to do this transformation by writing code that performs the transformation
NEVERACCEPT = \{ <M> | ACCEPT(M) = \emptyset \} (M accepts nothing)

**Claim**: NEVERACCEPT is undecidable

We will assume we know the following:

\[
\text{ACCEPT} = \{ <M,w> | M \text{ accepts } w \} \text{ is undecidable}
\]

**Proof:**

Suppose (towards contradiction) that there is a TM NA that decides NEVERACCEPT.
\[ \text{NEVERACCEPT} = \{ \langle M \rangle \mid \text{ACCEPT}(M) = \emptyset \} \]

**Claim:** NEVERACCEPT is undecidable

**Proof:**

Suppose (towards contradiction) that there is a TM NA that decides NEVERACCEPT.
NEVERACCEPt = \{ <M> | ACCEPT(M) = \emptyset \} \\

Claim: NEVERACCEPt is undecidable

Proof:

Suppose (towards contradiction) that there is a TM NA that decides NEVERACCEPt.

How many TMs?
when I design a compiler for a piece of code, I can’t worry about the input that this code will be fed many many years from now.

x and w not related!
NEVERACCEPT = \{ <M> \mid ACCEPT(M) = \emptyset \} \\

**Claim:** A decides ACCEPT

- **Case 1:** M accepts w.

  Implies M' accepts everything (by def. of M').

  Implies M' not in NEVERACCEPT (by def of NEVERACCEPT)

  Implies NA rejects <M'> (by def of NA)

  Implies A accepts <M,w> (by def of A)
NEVERACCEPT = \{ <M> \mid ACCEPT(M) = \emptyset \}

**Claim:** A decides ACCEPT

- **Case 2:** M doesn’t accept w.

  Implies M’ doesn’t accept anything (by def. of M’).

  Implies M’ in NEVERACCEPT (by def of NEVERACCEPT)

  Implies NA accepts <M’> (by def of NA)

  Implies A rejects <M,w> (by def of A)

  These two cases are exhaustive and imply A decides ACCEPT, contradiction
Rice’s Theorem

- We want to answer questions of the form “does the language this machine accepts have some interesting property?”

- \( L = \{ \text{set of acceptable languages that is not empty and is not the set of all languages} \} \)

  - e.g. \( L = \text{set of all languages containing the word “surfing”} \)

- Define \( \text{ACCEPTIN}(L) = \{ <M> | \text{ACCEPT}(M) \text{ is in } L \} \)

- \( L = \emptyset : \text{ACCEPTIN}(\emptyset) \) is decidable (always say no, no language is element of \( \emptyset \))

- \( L = \text{everything} : \text{ACCEPTIN}(\text{all}) \) is decidable (always say yes: does this TM accept a language?)

- For every other \( L \) \( \text{ACCEPTIN}(L) \) is undecidable
Rice’s Theorem

Rice’s Theorem. Let $\mathcal{L}$ be any set of languages that satisfies the following conditions:

- There is a Turing machine $Y$ such that $\text{Accept}(Y) \in \mathcal{L}$.
- There is a Turing machine $N$ such that $\text{Accept}(N) \notin \mathcal{L}$.

The language $\text{AcceptIn}(\mathcal{L}) := \{ \langle M \rangle \mid \text{Accept}(M) \in \mathcal{L} \}$ is undecidable.

To Show $\text{ACCEPTIN}(\mathcal{L})$ is undecidable

Reduce from $\text{HALT} = \{ \langle M, w \rangle \mid M \text{ halts on } w \}$
Rice’s Theorem

- ACCEPTIN(L) = \{ <M> | ACCEPT(M) is in L \}
- \( HALT = \{ <M,w> \mid M \text{ halts on } w \} \)

\[ M \text{ halts on } W \text{ iff } ACCEPT(WTF) \text{ is in } L \]
Rice’s Theorem

- ACCEPTIN(L) = \{<M>|\text{ACCEPT}(M) \text{ is in } L\}

\[\text{HALT} = \{<M,w>| M \text{ halts on } w\}\]

\[M \text{ halts on } w \iff \text{ACCEPT}(WTF) \text{ is in } L\]
Rice’s Theorem

- ACCEPTIN(L) = \{<M>|\text{ACCEPT}(M) \text{ is in } L\}
- HALT = \{<M,w>|M \text{ halts on } w\}

\[M \text{ halts on } w \iff \text{ACCEPT}(WTF) \text{ is in } L\]

Assume \(\emptyset \text{ not in } L\). Let \(Y\) be a TM so that \(\text{ACCEPT}(Y) \text{ in } L\)
Rice’s Theorem

- \( \text{ACCEPTIN}(L) = \{<M>|\text{ACCEPT}(M) \text{ is in } L\} \)

\[
\begin{align*}
\text{HALT} \\
<\text{M}> &\rightarrow \text{WTF}(x) \\
\text{M}(w) &\rightarrow \text{Y}(x) \\
w &\rightarrow <\text{WTF}> \\
\text{AIL} &\rightarrow \text{acc} \\
\text{rej} &\rightarrow \phi \text{ not in } L \\
\text{ACCEPT}(Y) &\text{ in } L \\
\text{acc} &\rightarrow \phi \text{ not in } L \\
\text{rej} &\rightarrow \phi \text{ not in } L \\
\end{align*}
\]

- if \( M \) halts on \( w \) then \( \text{WTF}(x) \) is \( \text{Y}(x) \) and
  \[ \text{ACCEPT}(\text{WTF}) = \text{ACCEPT}(Y) \text{ in } L, \text{ AIL accepts} \]
- if \( M \) doesn't halt on \( w \) then \( \text{WTF}(x) \) never halts
  \[ \text{so } \text{ACCEPT}(\text{WTF}) = \phi, \text{ not in } L, \text{ AIL rejects} \]
Rice’s Theorem

• $\text{ACCEPTIN}(L) = \{<M>|\text{ACCEPT}(M) \text{ is in } L\}$

$H$ accepts $<M,w>$ iff $H$ halts on $w$!

contradiction
Rice’s Theorem

- example: \{<M>| M accepts the empty string\}
- \(M_1\) accepts nothing : empty string is not in \(\emptyset\)
- \(M_2\) accepts everything: empty string is in \(S^*\)
- example: \{<M>| M accepts regular language\}
  - \(M_1\) accepts \(0^n1^n: n \geq 0\)
  - \(M_2\) accepts \(0^n1^n: n \geq 0\)