Lecture 11: Undecidability, Reductions, Rice’s Theorem
The Acceptance Problem for TMs

\[ A_{\text{TM}} = \{ (M, w) \mid M \text{ is a TM that accepts string } w \} \]

Given: code of a Turing machine \( M \) and an input \( w \) for that Turing machine,

Decide: Does \( M \) accept \( w \)?

\( A_{\text{TM}} \) decidable \( \Rightarrow \) There is an algorithm \( \text{ALG} \) which, given any code and input, \( \text{ALG} \) determines in finite time if the code will stop and accept the input

\[ \text{Theorem [Turing]}: \quad A_{\text{TM}} \text{ is recognizable, but NOT decidable!} \]
Theorem: L is decidable iff both L and \( \neg L \) are recognizable
Theorem: \( L \) is decidable
iff both \( L \) and \( \neg L \) are recognizable

Theorem: \( A_{TM} \) is recognizable but NOT decidable

Corollary: \( \neg A_{TM} \) is not recognizable!
Reducing One Problem to Another

\( f : \Sigma^* \rightarrow \Sigma^* \) is a **computable function** if there is a Turing machine \( M \) that halts with just \( f(w) \) written on its tape, for every input \( w \)

A language \( A \) is **mapping reducible** to language \( B \), written as \( A \leq_m B \), if there is a computable \( f : \Sigma^* \rightarrow \Sigma^* \) such that for every \( w \in \Sigma^* \),

\[
w \in A \iff f(w) \in B
\]

\( f \) is called a mapping reduction (or many-one reduction) from \( A \) to \( B \)
Let $f : \Sigma^* \rightarrow \Sigma^*$ be a computable function such that $w \in A \iff f(w) \in B$.

Say: “A is mapping reducible to B”
Write: $A \subseteq_m B$
Theorem: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable

Corollary: If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable

Theorem: If $A \leq_m B$ and $B$ is recognizable, then $A$ is recognizable

Corollary: If $A \leq_m B$ and $A$ is unrecognizable, then $B$ is unrecognizable
Theorem: $A_{TM} \leq_m \text{HALT}_{TM}$

Corollary: $\neg A_{TM} \leq_m \neg \text{HALT}_{TM}$

Corollary: $\neg \text{HALT}_{TM}$ is unrecognizable!

Proof: If $\neg \text{HALT}_{TM}$ were recognizable, then $\neg A_{TM}$ would also be recognizable, because $\neg A_{TM} \leq_m \neg \text{HALT}_{TM}$. But it’s not!

Question: $A_{TM} \leq_m \neg A_{TM}$? (NO!)

Theorem: $\text{HALT}_{TM} \leq_m A_{TM}$
Another Reduction Example

\[ EQ_{DFA} = \{ (D_1, D_2) \mid D_1 \text{ and } D_2 \text{ are DFAs, } L(D_1) = L(D_2) \} \]

\[ EQ_{REGEX} = \{ (R_1, R_2) \mid R_1 \text{ and } R_2 \text{ are regexps, } L(R_1) = L(R_2) \} \]

Theorem: \( EQ_{REGEX} \leq_m EQ_{DFA} \)

Proof: Mapping reduction \( f \) from \( EQ_{REGEX} \) to \( EQ_{DFA} \):

\( f \): On input \( z \), decode \( z \) into a pair \( (R_1, R_2) \),

Convert \( R_1, R_2 \) into NFAs \( N_1, N_2 \),

Convert NFAs \( N_1, N_2 \) into DFAs \( D_1, D_2 \). Output \( (D_1, D_2) \)

Then, \( (R_1, R_2) \in EQ_{REGEX} \iff L(D_1) = L(R_1) = L(R_2) = L(D_2) \)

\( \iff L(D_1) = L(D_2) \iff (D_1, D_2) \in EQ_{DFA} \)

So \( f \) is a mapping reduction from \( EQ_{REGEX} \) to \( EQ_{DFA} \)
The Emptiness Problem for TMs

EMPTY\textsubscript{TM} = \{ M \mid M is a TM such that L(M) = \emptyset \}

Given a program, does it reject or loop on all inputs?

**Theorem:** EMPTY\textsubscript{TM} is not recognizable

**Proof:** Show that \neg A_{TM} \leq_m EMPTY\textsubscript{TM}

f(z) := Decode z into (M, w). Output code of the TM:

"M'(x) := if (x = w) then output answer of M(w), else reject"

Observe: EITHER L(M') = \emptyset OR L(M') = \{w\}

z=(M,w) \notin A_{TM} \iff M doesn't accept w

\iff L(M') = \emptyset

\iff M' \in EMPTY\textsubscript{TM} \iff f(z) \in EMPTY\textsubscript{TM}
The Emptiness Problem for Other Models

\( \text{EMPTY}_{\text{DFA}} = \{ M \mid M \text{ is a DFA such that } L(M) = \emptyset \} \)

*Given a DFA, does it reject every input?*

**Theorem:** \( \text{EMPTY}_{\text{DFA}} \) is decidable

**Why?**

\( \text{EMPTY}_{\text{NFA}} = \{ M \mid M \text{ is a NFA such that } L(M) = \emptyset \} \)

\( \text{EMPTY}_{\text{REX}} = \{ R \mid M \text{ is a regexp such that } L(M) = \emptyset \} \)
The Equivalence Problem

\( EQ_{\text{TM}} = \{(M, N) \mid M, N \text{ are TMs and } L(M) = L(N)\} \)

*Do two programs accept exactly the same strings?*

**Theorem:** \( EQ_{\text{TM}} \) is *not recognizable*

**Proof:** Reduce \( \text{EMPTY}_{\text{TM}} \) to \( EQ_{\text{TM}} \)

Let \( M_\emptyset \) be a TM that always rejects immediately, so \( L(M_\emptyset) = \emptyset \)

Define \( f(M) := (M, M_\emptyset) \)

Then \( M \in \text{EMPTY}_{\text{TM}} \iff L(M) = L(M_\emptyset) \iff (M, M_\emptyset) \in EQ_{\text{TM}} \)
Moral: Analyzing Programs is Really, Really Hard for Programs to Do.
Two Problems

Problem 1  Undecidable

\{ (M, w) \mid M \text{ is a TM that on input } w, \text{ tries to move its head past the left end of the tape at some point} \}

Problem 2  Decidable

\{ (M, w) \mid M \text{ is a TM that on input } w, \text{ moves its head left at some point} \}
Problem 1  Undecidable

L’ = \{ (M, w) \mid M \text{ is a TM that on input } w, \text{ tries to move its head past the left end of the tape } \}

Proof:  Reduce $A_{TM}$ to L’

On input $(M,w)$,
make a TM N that shifts w over one cell,
puts a special symbol # on the leftmost cell,
then simulates M(w) on its tape.
If M’s head moves to the cell with # but has not yet accepted, N moves the head back to the right.
If M accepts, N tries to move its head past the #.

$(M,w)$ is in $A_{TM}$ if and only if $(N,w)$ is in L’
Problem 2 Decidable

\{ (M, w) \mid M \text{ is a TM that on input } w, \text{ moves its head left at some point} \}

On input \((M, w)\), run \(M\) on \(w\) for

\[ |Q| + |w| + 1 \text{ steps}, \]

where \(|Q| = \text{number of states of } M\)

Accept If M’s head moved left at all
Reject Otherwise

(Why does this work?)
Moral: Analyzing Programs is Really, Really Hard for Programs to Do.

How can we more easily tell when some “program analysis” problem is undecidable?
Problem 3

REVERSE = \{ M | M is a TM with the property: for all w, M(w) accepts \iff M(w^R) accepts \}. 

Decidable or not?

REVERSE is undecidable.
Rice’s Theorem: *Program Analysis is Hard*

Let $P : \{\text{Turing Machines}\} \rightarrow \{0,1\}$. (Think of 0=false, 1=true) Suppose $P$ satisfies:

1. **(Nontrivial)** There are TMs $M_1$ and $M_0$ where $P(M_1) = 1$ and $P(M_0) = 0$

2. **(Semantic)** For all TMs $M$ and $M'$, if $L(M) = L(M')$ then $P(M) = P(M')$

Then, $\{M \mid P(M) = 1\}$ is undecidable. In other words, function $P$ is undecidable.

A Huge Hammer for Undecidability!
Some Examples and Non-Examples

Semantic Properties $P(M)$
- $M$ accepts 0
- $L(M)$ is empty
  - $L(M) = \Sigma^*$
- $M$ accepts 6045 strings
- for all $w$, $M(w)$ accepts $\iff M(w^R)$ accepts

Rice says: $\{M \mid P(M) = 1\}$ is undecidable

Not Semantic!
- $M$ halts and rejects 0
- $M$ has at least 6045 states
- $M$ halts on all inputs
- $M$’s head tries to move off the left end of the tape on some input

$P$ is not semantic: There are $M$ and $M'$ such that $L(M) = L(M')$ but $P(M) \neq P(M')$
Rice’s Theorem: If P is nontrivial and semantic, then \( L_P := \{M \mid P(M) = 1\} \) is undecidable.

Proof: Either reduce \( \overline{A_{TM}} \) or \( A_{TM} \) to \( L_P \)

Let \( M_\emptyset \) be any TM such that \( L(M_\emptyset) = \emptyset \)

Case 1: Suppose \( P(M_\emptyset) = 0 \) (\( M_\emptyset \not\in L_P \))

Since \( P \) is nontrivial, there’s \( M_1 \) such that \( P(M_1) = 1 \)

Reduction from \( A_{TM} \) to \( L_P \) On input \((M, w)\), output:

“\( M_w(x) := \) Run \( M \) on \( w \). If \( M \) accepts then run \( M_1 \) on \( x \) and output its answer, else reject.”

If \( M \) accepts \( w \), then \( L(M_w) = L(M_1) \)

Since \( P(M_1) = 1 \), we have \( P(M_w) = 1 \), so \( M_w \in L_P \)

If \( M \) does not accept \( w \), then \( L(M_w) = L(M_\emptyset) = \emptyset \)

Since \( P(M_\emptyset) = 0 \), we have \( P(M_w) = 0 \) and \( M_w \not\in L_P \)
Rice’s Theorem: If \( P \) is nontrivial and semantic, then \( L_P := \{M \mid P(M) = 1\} \) is undecidable.

Proof:

Either reduce \( A_{TM} \) or \( \neg A_{TM} \) to \( L_P \)

Let \( M_\emptyset \) be any TM such that \( L(M_\emptyset) = \emptyset \)

Case 2: Suppose \( P(M_\emptyset) = 1 \) (\( M_\emptyset \in L_P \))

Since \( P \) is nontrivial, there’s \( M_0 \) such that \( P(M_0) = 0 \)

Reduction from \( \neg A_{TM} \) to \( L_P \) On input \((M, w)\), output:

“\( M_w(x) := \) Run \( M \) on \( w \). If \( M \) accepts then run \( M_0 \) on \( x \) and output its answer, else reject.”

If \( M \) does not accept \( w \), then \( L(M_w) = L(M_\emptyset) = \emptyset \)

Since \( P(M_\emptyset) = 1 \), we have \( P(M_w) = 1 \), so \( M_w \in L_P \)

If \( M \) accepts \( w \), then \( L(M_w) = L(M_0) \)

Since \( P(M_0) = 0 \), we have \( M_w \notin L_P \)
The Regularity Problem for Turing Machines

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

*Given a program, is it equivalent to some DFA?*

**Theorem:** \( \text{REGULAR}_{\text{TM}} \) is *not recognizable*

**Proof:** Use Rice’s Theorem!

\[ P(M) = 1 \text{ iff } L(M) \text{ is regular} \text{ is nontrivial:} \]

- there’s an \( M_\emptyset \) which never halts: \( P(M_\emptyset) = 1 \)
- there’s an \( M' \) deciding \( \{0^n1^n \mid n \geq 0\} \): \( P(M') = 0 \)

\( P \) is also *semantic*:

If \( L(M) = L(M') \) then \( L(M) \) is regular \( \iff \) \( L(M') \) is regular, so \( P(M) = 1 \iff P(M') = 1 \), so \( P(M) = P(M') \)

By Rice’s Thm (case 2), we have

\[ \neg A_{\text{TM}} \leq_m \text{REGULAR}_{\text{TM}} \]
Next Episode:

Your Midterm… Good Luck!