Lecture 11: More on Undecidability and Reductions
Thursday (3/23)

Midterm: 2:35-4:00pm,
Walker Memorial (50-340)

All pset and practice midterm solutions are on piazza!
Thursday (3/23)

Midterm: 2:35-4:00pm, Walker Memorial (50-340)

All pset and practice midterm solutions are on piazza!

FAQ: What material is on the midterm?
Everything up to this lecture (Lectures 1-11)
But we’ll focus more on earlier material

FAQ: Can I bring notes?
Yes, one single-sided sheet of notes, letter paper
Theorem: $L$ is decidable iff both $L$ and $\overline{L}$ are recognizable.

$L \in \Sigma^*$

$w \in L \ ?$

- **yes**: accept
- **no**: reject or loop

$L$ is decidable (recursive)

$L$ is recognizable (recursively enumerable)
The Acceptance Problem for TMs

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

Given: code of a Turing machine M and an input w for that Turing machine,
Decide: Does M accept w?

\( A_{TM} \) decidable \( \Rightarrow \) There is an algorithm ALG which, given any code and input,
ALG determines in finite time if the code will stop and output 1 when executed

Theorem [Turing]:
\( A_{TM} \) is recognizable, but NOT decidable!
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: $A \leq_m B$
Another Example

\[
E_{Q_{DFA}} = \{ (D_1, D_2) \mid D_1 \text{ and } D_2 \text{ are DFAs, } L(D_1) = L(D_2) \}
\]
\[
E_{Q_{REGEX}} = \{ (R_1, R_2) \mid R_1 \text{ and } R_2 \text{ are regexps, } L(R_1) = L(R_2) \}
\]

Theorem: \( E_{Q_{REGEX}} \leq_m E_{Q_{DFA}} \)

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

\( f \): 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, \( (R_1, R_2) \in E_{Q_{REGEX}} \iff L(D_1) = L(R_1) = L(R_2) = L(D_2) \)

\[ \iff L(D_1) = L(D_2) \iff (D_1, D_2) \in E_{Q_{DFA}} \]

So \( f \) is a mapping reduction from \( E_{Q_{REGEX}} \) to \( E_{Q_{DFA}} \)
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
A mapping reduction from $A_{TM}$ to $HALT_{TM}$

Theorem: $A_{TM} \leq_m HALT_{TM}$

$f(z) :=$ Decode $z$ into a pair $(M, w)$

Construct a TM $M'$ with the specification:

“$M'(w) =$ Simulate $M$ on $w$.

if the sim accepts then accept

else loop forever”

Output $(M', w)$

We have $z \in A_{TM} \iff (M', w) \in HALT_{TM}$

Corollary: $HALT_{TM}$ is undecidable
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: $A_{TM} \leq_m \neg A_{TM}$?
Theorem: $\text{HALT}_{TM} \leq_m A_{TM}$

Proof: Define the computable function $f$:

$$f(z) := \text{Decode } z \text{ into a pair } (M, w)$$
Construct a TM $M'$ with the specification:
“$M'(w) = \text{Simulate } M \text{ on } w.$
If the simulation halts, then accept”
Output $(M', w)$

Claim: $(M, w) \in \text{HALT}_{TM} \iff (M', w) \in A_{TM}$
Corollary: $\text{HALT}_{\text{TM}} \equiv_{\text{m}} \text{A}_{\text{TM}}$

Yo, T.M.! I can give you the magical power to either solve the halting problem, or the acceptance problem. Which do you want?

Wow, hm, so hard to choose...

I can’t decide!
The Emptiness Problem for TMs

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

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

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

Proof: Show that \( \neg A_{\text{TM}} \leq_m \text{EMPTY}_{\text{TM}} \)

\[ f(z) := \text{Decode } z \text{ into } (M, w). \text{ Output code of the TM:} \]
\[ "M'(x) := \text{if } (x = w) \text{ then output answer of } M(w), \]
\[ \text{else reject}" \]

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

\( z=(M,w) \notin A_{\text{TM}} \iff M \text{ doesn’t accept } w \)
\[ \iff L(M') = \emptyset \]
\[ \iff M' \in \text{EMPTY}_{\text{TM}} \iff f(z) \in \text{EMPTY}_{\text{TM}} \]
The Emptiness Problem for Other Models

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

Given a DFA, does it reject every input?

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

Why?

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

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

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

*Do two programs compute the same function?*

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

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

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

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

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

Problem 1  Undecidable
{ (M, w) | M is a TM that on input w, tries to move its head past the left end of the input } 

Problem 2  Decidable
{ (M, w) | M is a TM that on input w, moves its head left at least once, at some point}
Problem 1  Undecidable

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

Proof:  Reduce \( A_{\text{TM}} \) 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 \( A_{\text{TM}} \) 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 least once, at some point} \}

On input \((M, w)\), run \(M\) on \(w\) for 
\(|Q| + |w| + 1\) 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 \mid M \text{ is a TM with the property: for all } w, M(w) \text{ accepts } \iff M(w^R) \text{ 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 $M_{\text{YES}}$ and $M_{\text{NO}}$ where $P(M_{\text{YES}}) = 1$ and $P(M_{\text{NO}}) = 0$
2. (Semantic) For all TMs $M_1$ and $M_2$, if $L(M_1) = L(M_2)$ then $P(M_1) = P(M_2)$

Then, $L = \{M \mid P(M) = 1\}$ is undecidable.

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

**Semantic Properties \( P(M) \)**

- \( M \) accepts 0
- \( L(M) = \{0\} \)
- \( L(M) \) is empty
- \( L(M) = \Sigma^* \)
- \( M \) accepts 6045 strings
- for all \( w \), \( M(w) \) accepts \( \iff M(w^R) \) accepts

\[ L = \{ M \mid P(M) = 1 \} \]

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 \( M_1 \) and \( M_2 \) such that \( L(M_1) = L(M_2) \) and \( P(M_1) \neq P(M_2) \)
Rice’s Theorem: If P is nontrivial and semantic, then \( L = \{ M \mid P(M) = 1 \} \) is undecidable.

Proof: Either reduce \( A_{TM} \) or \( \neg A_{TM} \) to the language \( L \)

Define \( M_\emptyset \) to be a TM such that \( L(M_\emptyset) = \emptyset \)

Case 1: Suppose \( P(M_\emptyset) = 0 \)

Since \( P \) is nontrivial, there’s \( M_{YES} \) such that \( P(M_{YES}) = 1 \)

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

\[ M_w(x) := \text{If } ((M \text{ accepts } w) \& (M_{YES} \text{ accepts } x)) \text{ then ACCEPT, else REJECT} \]

If \( M \) accepts \( w \), then \( L(M_w) = L(M_{YES}) \)

Since \( P(M_{YES}) = 1 \), we have \( P(M_w) = 1 \) and \( M_w \in L \)

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 \( M_w \notin L \)
Rice’s Theorem: If P is nontrivial and semantic, then \( L = \{M \mid P(M) = 1\} \) is undecidable.

Proof: Either reduce \( A_{TM} \) or \( \neg A_{TM} \) to the language \( L \)
Define \( M_\emptyset \) to be a TM such that \( L(M_\emptyset) = \emptyset \)

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

Since P is nontrivial, there’s \( M_{\text{NO}} \) such that \( P(M_{\text{NO}}) = 0 \)

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

“\( M_w(x) := \text{If } ((M \text{ accepts } w) \& (M_{\text{NO}} \text{ accepts } x)) \text{ then ACCEPT, else REJECT} \)”

If M does not accept w, then \( L(M_w) = L(M_\emptyset) = \emptyset \)
Since \( P(M_\emptyset) = 1 \), we have \( P(M_w) = 1 \) and \( M_w \in L \)

If M accepts w, then \( L(M_w) = L(M_{\text{NO}}) \)
Since \( P(M_{\text{NO}}) = 0 \), we have \( M_w \notin L \)
The Regularity Problem for Turing Machines

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

Given a program, is it equivalent to some DFA?

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

**Proof:** Use Rice’s Theorem!

$P(M) := \text{“}L(M) \text{ is regular”}$ 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

$\neg \text{A}_{TM} \leq_m \text{REGULAR}_{TM}$
Next Episode:

Your Midterm... Good Luck!