Lecture 11: Fun With Undecidability!
Announcements

- If MIT is denying you resources you need and you’re running out of options, please contact me personally.

- Pset now due **Monday March 30**: we will release pset solutions immediately after that
- For now, midterm is still **Thursday April 2**
- Practice midterm + solutions out tonight!
  - There is candy!
The Acceptance Problem for TMs

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

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

Decide: Does M accept w?

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

**Theorem [Turing]:**

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

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

Corollary: \( \neg A_{TM} \) is not recognizable!

Theorem: \( \text{HALT}_{TM} \) is not decidable
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 $B$. 
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: If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable

\[ w \in \Sigma^* \quad \xrightarrow{f} \quad f(w) \in \Sigma^* \quad \xrightarrow{\text{accept/reject}} \quad B \]

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

A recipe for proving undecidability!
To prove $B$ is undecidable, find undecidable $A$ and a mapping reduction from $A$ to $B$. 
Theorem: $A_{TM} \leq_m HALT_{TM}$

$f(z) :=$ Decode $z$ into a pair $\langle M, w \rangle$. Write down the description of a TM $M'$ with the spec:

“$M'(w) =$ Run $M$ on $w$. If $M$ accepts, then accept, else loop forever”

Output the encoding $\langle M', w \rangle$

Then, $z=\langle M, w \rangle \in A_{TM} \iff M$ accepts $w 
\iff M'$ halts on $w \iff \langle M', w \rangle \in HALT_{TM}$

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 \( \neg A_{TM} \) is not!

Question: \( A_{TM} \leq_m \neg A_{TM} \)?

Theorem: \( \text{HALT}_{TM} \leq_m A_{TM} \)
Theorem: \( \text{HALT}_{\text{TM}} \leq_m \text{A}_{\text{TM}} \)

Proof: Define a mapping reduction \( f \):

\[ f(z) := \text{Decode } z \text{ into a pair } \langle M, w \rangle \]

Write down a TM \( M' \) with the specification:

“\( M'(w) = \text{Run } M \text{ on } w. \text{ If } M \text{ halts, accept} \)”

Output \( \langle M', w \rangle \)

Observe \( z = (M, w) \in \text{HALT}_{\text{TM}} \iff (M', w) \in \text{A}_{\text{TM}} \)
Corollary: $\text{HALT}_{TM} \equiv_m A_{TM}$

Yo, T.M.! I can give you the magical power to either compute 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

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

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

**Theorem:** \( \text{EMPTY}_{\text{TM}} \) is *unrecognizable*

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

\[ f(z) := \text{Decode } z \text{ into } \langle M, w \rangle. \text{ Output code of the TM: } \]

\[ "M'(x) := \text{if } (x = w) \text{ then run } M(w) \text{ and output answer, else reject"} \]

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

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

\[ \iff L(M') = \emptyset \]

\[ \iff \langle M' \rangle \in \text{EMPTY}_{\text{TM}} \iff f(z) \in \text{EMPTY}_{\text{TM}} \]
The Emptiness Problem for Other Models

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

Given a DFA, does it reject every input?

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

Why?

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

\[ \text{EMPTY}_{\text{REX}} = \{ \langle R \rangle \mid M \text{ is a regexp such that } L(M) = \emptyset \} \]
Moral: Analyzing Programs is Really, Really Hard for Programs to Do. (Sometimes)
Computing With Oracles: Another Kind of Reduction

*We do not condone smoking. Don’t do it. It’s bad. Kthxbye
Oracle Turing Machines

Is \((M, w)\) in \(A_{TM}\)?

\[ q_{YES} \]

\[ \text{yes!} \]

INPUT

INFINITE TAPE

Now leaving reality for a moment....
Oracle Turing Machines

An oracle Turing machine $M$ is equipped with a set $B \subseteq \Gamma^*$ and a special oracle tape, on which $M$ may ask membership queries about $B$

Formally, $M$ enters a special state $q_?$ to ask a query

and the TM receives a query answer in one step

(Formally, the transition function on $q_?$ is defined in terms of the entire oracle tape:

State $q_?$ changes to $q_{\text{YES}}$
if the string $y$ written on the oracle tape is in $B$,
else $q_?$ changes to $q_{\text{NO}}$]

This notion makes sense even if $B$ is not decidable!
Think in terms of Turing Machine pseudocode!

An oracle Turing machine $M$ with oracle $B \subseteq \Gamma^*$ lets you include the following kind of if-then statement:

```
“if (z in B) then <do something>
else <do something else>”
```

where $z$ is some string defined earlier in pseudocode.

We define the oracle TM to that it can always check the condition $(z \text{ in } B)$ in one step.

This notion makes sense even if $B$ is not decidable!
Deciding one problem with another

Definition: A is decidable with B if there is an oracle TM M with oracle B that accepts strings in A and rejects strings not in A

Language A “Turing-Reduces” to B

A \leq_T B
A_{TM} is decidable with \text{HALT}_{TM} \quad (A_{TM} \leq_T \text{HALT}_{TM})

We can decide if \( M \) accepts \( w \) using an ORACLE for the Halting Problem:

On input \((M, w)\),

If \((M, w)\) is in \text{HALT}_{TM} then
    run \( M(w) \) and output its answer.
else \text{REJECT}.

(This is exactly like our proof that \text{HALT}_{TM} is undecidable, from last lecture!)
HALT_{TM} is decidable with A_{TM} (HALT_{TM} \leq_T A_{TM})

On input (M,w), decide if M halts on w as follows:

1. If (M,w) is in A_{TM} then ACCEPT

2. Else, swap the accept and reject states of M to get a machine M'. If (M',w) is in A_{TM} then ACCEPT

3. REJECT
Theorem: If $A \leq_T B$ and $B$ is decidable,
then $A$ is decidable

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

Proof: Exactly the same proof
as the one for mapping reductions!

If $A \leq_T B$ then there is a TM $M$ with oracle $B$
that decides $A$. If $B$ is decidable, then we can
replace every oracle call to $B$ with a TM
that decides $B$. Now $M$ is a TM with no oracle!
\[ \leq_T \text{ versus } \leq_m \]

**Theorem:** If \( A \leq_m B \) then \( A \leq_T B \)

**Proof (Sketch):**

A \( \leq_m B \) means there is a computable function \( f : \Sigma^* \rightarrow \Sigma^* \), where for every \( w \),

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

To decide A on an input \( w \) with oracle B, just compute \( f(w) \), then call B on \( f(w) \) and return answer

**Theorem:** \( \neg A_{TM} \leq_T A_{TM} \)

D (\( \langle M, w \rangle \)): If (\( \langle M, w \rangle \) in \( A_{TM} \)) then reject else accept

**Theorem:** \( \neg A_{TM} \not\leq_m A_{TM} \)
The Busy Beaver Function

How much work can a little TM do?
The Busy Beaver Function

Define a simple Turing machine to be one with input alphabet \{1\}, tape alphabet \{1,□\}, and a “halt state”. Besides the “halt state”, our TMs have \(n\) other states.

Define \(\text{BB}(n)\) to be the maximum number of steps taken on input \(\varepsilon\) by any \(n\)-state simple TM that halts.

\(\text{BB}(1) = 1\): For a 1-state TM running on blank tape, it either halts in the first step, or it runs forever!

\(\text{BB}(2) = 6\)
\(\text{BB}(3) = 21\)
\(\text{BB}(4) = 107\)
\(\text{BB}(5) \geq 47,176,870\)
\(\text{BB}(6) > 10^{36,534}\)
\(\text{BB}(7) > 10^{10^{10^{10^{18,000,000}}}}\)
The Busy Beaver Function

Theorem: $BB(n)$ is not computable!

$BB(n)$ grows so ridiculously fast that no computable function whatsoever (no function you have ever seen) can even upper bound it!!

First Idea: If you could compute $BB(n)$, then you could solve the Halting problem for simple TMs running on blank tape!

Second Idea: It is impossible to decide that Halting problem
The Busy Beaver Function

Theorem: \( \text{BB}(n) \) is not computable!

Theorem: Assuming there is a TM computing \( \text{BB}(n) \), we can solve the Halting problem for simple TMs on \( \varepsilon \).

Proof: Here’s pseudocode for the Halting problem:

On the input \( \langle M \rangle \) [code of a TM \( M \)]

Count the number of states in \( M \), call it \( n \)
Compute \( t = \text{BB}(n) \) [in binary or unary]
Run \( M \) on blank tape for \( t \) steps.
If it halts, then accept. Otherwise, reject!

Theorem: There is NO computable function \( f : \mathbb{N} \rightarrow \mathbb{N} \) such that for all \( n \), \( f(n) \geq \text{BB}(n) \).
The Busy Beaver Function

You can encode arbitrary math conjectures in simple TMs!

**Theorem:** There is a 1919-state simple TM that halts iff ZFC (set theory) is inconsistent!

There is a 744-state simple TM that halts iff the Riemann hypothesis is false.

There is a 43-state simple TM that halts iff Goldbach's conjecture is false

Good luck verifying if those halt!
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 \( A_{\text{TM}} \) 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 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?)
Thank you all ... 

... see you in June, hopefully?

Be safe!