CS154, Lecture 10: Rice’s Theorem, Oracle Machines
Moral: Analyzing Programs is Really, Really Hard

But can we more easily tell when some “program analysis” problem is undecidable?
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 input } \}

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}\}
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 $A_{TM}$ to $L'$

On input $(M, w)$, make a TM $N$ that shifts $w$ over one cell, marks a special symbol $\$$ on the leftmost cell, then simulates $M(w)$ on the 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 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?)
Problem 3

REVERSE = \{ M \mid M \text{ is a TM with the property:}
for all $w$, $M(w)$ accepts $\Leftrightarrow M(w^R)$ accepts\}.

Decidable or not?

REVERSE is undecidable.
Rice’s Theorem

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

<table>
<thead>
<tr>
<th>Semantic Properties $P(M)$</th>
<th>Not Semantic!</th>
</tr>
</thead>
<tbody>
<tr>
<td>• M accepts 0</td>
<td>• M halts and rejects 0</td>
</tr>
<tr>
<td>• for all $w$, $M(w)$ accepts iff $M(w^R)$ accepts</td>
<td>• M tries to move its head off the left end of the tape, on input 0</td>
</tr>
<tr>
<td>• $L(M) = {0}$</td>
<td>• M never moves its head left on input 0</td>
</tr>
<tr>
<td>• $L(M)$ is empty</td>
<td>• M has exactly 154 states</td>
</tr>
<tr>
<td>• $L(M) = \Sigma^*$</td>
<td>• M halts on all inputs</td>
</tr>
<tr>
<td>• M accepts 154 strings</td>
<td>There are $M_1$ and $M_2$ such that $L(M_1) = L(M_2)$ and $P(M_1) \neq P(M_2)$</td>
</tr>
</tbody>
</table>

$L = \{M \mid P(M) \text{ is true}\}$ is undecidable
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: $P(M_{\emptyset}) = 0$

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

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

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

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

Since $P(M_{\text{YES}}) = 1$, 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 $M_w \not\in L$
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: $P(M_\emptyset) = 1$

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

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

"$M_w(x) := \text{If } ((M \text{ accepts } w) \& (M_{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 $M_w \in L$

If $M$ accepts $w$, then $L(M_w) = L(M_{NO})$

Since $P(M_{NO}) = 0$, we have $M_w \notin L$
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!

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

- there’s an \( M_{\emptyset} \) such that \( L(M_{\emptyset}) = \emptyset \): \( \text{P}(M_{\emptyset}) = 1 \)
- there’s an \( M' \) deciding \( \{0^n1^n \mid n \geq 0\} \): \( \text{P}(M') = 0 \)

\( \text{P} \) is also semantic:

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

By Rice’s Thm, we have \( \neg A_{\text{TM}} \leq_m \text{REGULAR}_{\text{TM}} \)
Definition: A decidable predicate \( R(x,y) \) is a proposition about the input strings \( x \) and \( y \), such that some TM \( M \) implements \( R \). That is, for all \( x, y \),  
\[
\text{\( R(x,y) \) is TRUE } \Rightarrow \text{\( M(x,y) \) accepts}
\]
\[
\text{\( R(x,y) \) is FALSE } \Rightarrow \text{\( M(x,y) \) rejects}
\]

Can think of \( R \) as a function from \( \Sigma^* \times \Sigma^* \rightarrow \{T,F\} \)

Examples:  
\( R(x,y) = \text{“} xy \text{ has at most 100 zeroes”} \)  
\( R(N,y) = \text{“} TM N \text{ halts on } y \text{ in at most 99 steps”} \)
Theorem: A language $A$ is \textit{recognizable} if and only if there is a decidable predicate $R(x, y)$ such that: $A = \{ x \mid \exists y \ R(x, y) \}$

Proof:

(1) If $A = \{ x \mid \exists y \ R(x,y) \}$ then $A$ is recognizable

Define the TM $M(x)$: Enumerate all finite-length strings $y$, If $R(x,y)$ is true, accept $\Rightarrow M$ accepts exactly those $x$ s.t. $\exists y \ R(x,y)$ is true

(2) If $A$ is recognizable, then there is a decidable predicate $R(x, y)$ such that: $A = \{ x \mid \exists y \ R(x,y) \}$

Suppose TM $M$ recognizes $A$. Let $R(x,y)$ be TRUE iff $M$ accepts $x$ in $|y|$ steps $\Rightarrow M$ accepts $x \Leftrightarrow \exists y \ R(x,y)$
Oracle Turing Machines, Turing Reductions and Hierarchies
Oracle Turing Machines

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

yes

FINITE STATE CONTROL

INFINITE REWRITABLE TAPE
Oracle Turing Machines

An oracle Turing machine $M$ that can ask membership queries in a set $B \subseteq \Gamma^*$ on a special “oracle tape” [Formally, $M$ enters a special state $q_b$]

The TM receives an answer to the query in one step[Formally, the transition function on $q_b$ is defined in terms of the entire oracle tape: if the string $y$ written on the oracle tape is in $B$, then state $q_b$ is changed to $q_{\text{YES}}$, otherwise $q_{\text{NO}}$]

This notion makes sense even if $B$ is not decidable!
How to Think about Oracles?

A black-box subroutine. In terms of Turing Machine pseudocode: An oracle Turing machine $M$ with oracle $B \subseteq \Gamma^*$ lets you include the following kind of branching instructions:

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

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

By definition, the oracle TM can always check the condition $(z \text{ in } B)$ in one step

This notion makes (mathematical) sense even if $B$ is not decidable.
Definition: A is recognizable with B if there is an oracle TM $M$ with oracle $B$ that recognizes $A$.

Definition: A is decidable with B if there is an oracle TM $M$ with oracle $B$ that decides $A$.

Language $A$ “Turing-Reduces” to $B$.

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

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 REJECT.
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, switch 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 $A \leq_m B$ then $A \leq_T B$

Proof (Sketch):

If $A \leq_m B$ then there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every $w$,

$$w \in A \iff f(w) \in B$$

To decide $A$ on the string $w$, just compute $f(w)$ and “call the oracle” for $B$

Theorem: $\neg\text{HALT}_{TM} \leq_T \text{HALT}_{TM}$

Theorem: $\neg\text{HALT}_{TM} \not\leq_m \text{HALT}_{TM}$

Why?
Limitations on Oracle TMs!

The following problem cannot be decided by any TM with an oracle for the Halting Problem:

\[ \text{SUPERHALT} = \{ (M,x) \mid M, \text{ with an oracle for the Halting Problem, halts on } x \} \]

We can use the proof by diagonalization!

Assume \( H \) (with \( \text{HALT} \) oracle) decides \( \text{SUPERHALT} \)

Define \( D(X) := \text{"if } H(X,X) \text{ (with } \text{HALT} \text{ oracle) accepts then LOOP, else ACCEPT."} \) (\( D \) uses a \( \text{HALT} \) oracle to simulate \( H \))

But \( D(D) \) halts \( \iff \) \( H(D,D) \) accepts \( \iff \) \( D(D) \) loops...

(by assumption) \hspace{1cm} (by def of \( D \))
Limits on Oracle TMs

“Theorem” There is an infinite hierarchy of unsolvable problems!

Given ANY oracle $O$, there is always a harder problem that cannot be decided with that oracle $O$

$\text{SUPERHALT}^0 = \text{HALT} = \{ (M,x) | M \text{ halts on } x \}.$

$\text{SUPERHALT}^1 = \{ (M,x) | M, \text{ with an oracle for } \text{HALT}_{\text{TM}}, \text{ halts on } x \}$

$\text{SUPERHALT}^n = \{ (M,x) | M, \text{ with an oracle for } \text{SUPERHALT}^{n-1}, \text{ halts on } x \}$
\[
\sum_1^0 \Delta_1^0 \subseteq \sum_2^0 \cap \Pi_2^0 \subseteq \Lambda_{TM} \subseteq \Delta_2^0 \cap \Pi_2^0 \subseteq \Delta_3^0 \subseteq \Pi_3^0
\]

Decidable languages

Co-R.E. Languages