Lecture 12: Oracles, Self-Reference, Foundations of Mathematics
Midterms back after lecture today!

Thanks for your feedback
### Some interesting feedback

<table>
<thead>
<tr>
<th>Feedback</th>
<th>Number of students</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW is too hard</td>
<td>0</td>
</tr>
<tr>
<td>HW is too easy</td>
<td>1</td>
</tr>
<tr>
<td>Pace is too fast</td>
<td>4</td>
</tr>
<tr>
<td>Likes Ryan/slides/lectures</td>
<td>54</td>
</tr>
<tr>
<td>Dislikes Ryan/slides/lectures</td>
<td>4</td>
</tr>
<tr>
<td>Likes TAs and recitations</td>
<td>9</td>
</tr>
<tr>
<td>Wants fewer websites</td>
<td>5</td>
</tr>
<tr>
<td>More Office Hours (evenings?)</td>
<td>4</td>
</tr>
<tr>
<td>Want more examples/practice</td>
<td>11</td>
</tr>
<tr>
<td>Dislikes guest lectures</td>
<td>4</td>
</tr>
<tr>
<td>I dislike that I don’t have any criticism</td>
<td>2</td>
</tr>
<tr>
<td>More videos!</td>
<td>5</td>
</tr>
<tr>
<td>HW returns earlier?</td>
<td>7</td>
</tr>
</tbody>
</table>
Recognizability via Logic

Def. 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 $x, y$, $R(x,y)$ is TRUE $\Rightarrow$ $M(x,y)$ accepts

$R(x,y)$ is FALSE $\Rightarrow$ $M(x,y)$ rejects

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

EXAMPLES: $R(x,y) = \text{"xy has at most 100 zeroes"}$

$R(N,y) = \text{"TM N halts on y in at most 99 steps"}$
Theorem: A language $A \subseteq \Sigma^*$ is *recognizable* if and only if there is a decidable predicate $R(x, y)$ such that:

$$A = \{ x \mid \exists y \in \Sigma^* \; R(x, y) \}$$

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

Define the TM $M(x)$: For all strings $y \in \Sigma^*$,

If $R(x, y)$ is true, accept.

Then, $M$ accepts exactly those $x$ s.t. $\exists y \; R(x, y)$ is true

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

Suppose TM $M$ recognizes $A$.

Define $R(x, y)$ to be TRUE iff $M$ accepts $x$ in $|y|$ steps.

Then, $M$ accepts $x \iff \exists y \; R(x, y)$
Example: \[ L = \{ M \mid M \text{ accepts at least one string}\} \text{ is recognizable.} \]

Want: decidable predicate \( R \) such that
\[ L = \{ M \mid \exists y \in \Sigma^* \ R(M, y) \text{ is true} \} \]

Define \( R(M,(x,y)) = \text{“TM M accepts string } x \text{ in } |y| \text{ steps”} \)
Note that \( R(M,(x,y)) \) is decidable!

Then: \[ L = \{ M \mid \exists (x,y) \in \Sigma^* \ R(M, (x,y)) \text{ is true}\} \]

So \( L \) is recognizable!
Computability With Oracles

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

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

\(q_{YES}\)

yes!

\[\text{INPUT}\]

\[\text{INFINITE TAPE}\]

Now leaving reality for a moment....
An oracle Turing machine $M$ is equipped with a set $B \subseteq \Gamma^*$ to which a TM $M$ may ask membership queries on a special “oracle tape”
[Formally, $M$ enters a special state $q_?$]

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:
  if the string $y$ written on the oracle tape is in $B$, then state $q_?$ is changed to $q_{YES}$, otherwise $q_{NO}$]

This notion makes sense even if $B$ is not decidable!

How to Think about Oracles?

Think 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 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 $HALT_{TM}$ ($A_{TM} \leq_T HALT_{TM}$)

We can decide if $M$ accepts $w$ using an ORACLE for the Halting Problem:

On input $(M, w)$,

If $(M, w)$ is in $HALT_{TM}$ 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 $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$,

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

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

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

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

*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 HALT oracle) decides \( \text{SUPERHALT} \)

Define \( D(X) := \text{“if } H(X,X) \text{ (with HALT oracle) accepts then LOOP, else ACCEPT.”} \)

\( D \) uses a HALT oracle to simulate \( H \)

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

(by assumption on \( H \)) \quad \text{(by def of } D \text{)}
“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

\[
\text{SUPERHALT}^0 = \text{HALT} = \{ (M,x) \mid M \text{ halts on } x \}.
\]

\[
\text{SUPERHALT}^1 = \{ (M,x) \mid M, \text{ with an oracle for } \text{HALT}^\text{TM}, \text{ halts on } x \}
\]

\[
\text{SUPERHALT}^n = \{ (M,x) \mid M, \text{ with an oracle for } \text{SUPERHALT}^{n-1}, \text{ halts on } x \}
\]
Given three instances
\((M_1, w_1), (M_2, w_2), (M_3, w_3)\)
of the Halting Problem,

Can you decide \((M_i, w_i) \in \text{HALT}\) for all \(i\),
with only TWO oracle calls to \(\text{HALT}\)?
Self-Reference and the Recursion Theorem
Lemma: There is a computable function $q : \Sigma^* \rightarrow \Sigma^*$ such that for every string $w$, $q(w)$ is the *description* of a TM $P_w$ that on every input, prints out $w$ and then accepts.

"Proof" Define a TM $Q$:
Theorem: There is a Self-Printing TM

Proof: First define a TM $B$ which does this:

Now consider the TM that looks like this:

No explicit self-reference here!

QED
The Recursion Theorem

**Theorem:** For every TM $T$ computing a function $t : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ there is a Turing machine $R$ computing a function $r : \Sigma^* \rightarrow \Sigma^*$, such that for every string $w$,

$$r(w) = t(R, w)$$
Proof: \((a,b) \rightarrow T \rightarrow t(a,b)\)

Define \(M = \)

Define \(R:\)

What is \(S?\)
Proof: \((a,b) \rightarrow T \rightarrow t(a,b)\)

Define \(M = \)

Define \(R:\)

What is \(M(M,w)\)?
Proof: \((a,b) \rightarrow T \rightarrow t(a,b)\)

Define \(R\):

\[ S = Y = R. \quad \text{QED} \]
\begin{align*}
  \text{FOO}_x(y) & := \text{Output } x \text{ and halt.} \\
  \text{BAR}(M) & := \text{Output } \texttt{"N(w) = Run FOO}_M \text{ outputting } M. \\
  & \quad \text{Run } M \text{ on } (M, w) \texttt{"} \\
  \text{Q}(N, w) & := \text{Run BAR}(N) \text{ outputting } S. \\
  & \quad \text{Run } T \text{ on } (S, w) \\
  \text{R}(w) & := \text{Run FOO}_Q \text{ outputting } Q. \\
  & \quad \text{Run BAR}(Q) \text{ outputting } S. \\
  & \quad \text{Run } T \text{ on } (S, x) \\
  \text{Claim: } S & \text{ is a description of } R \text{ itself!} \\
  \text{S}(w) & := \text{Run FOO}_Q \text{ outputting } Q. \\
  & \quad \text{Run } Q \text{ on } (Q, w)
\end{align*}
\[ FOO_x(y) := \text{Output } x \text{ and halt.} \]
\[ BAR(M) := \text{Output } \text{“} N(w) = \text{Run } FOO_M \text{ outputting } M. \text{ Run } M \text{ on } (M, w) \text{”} \]

\[ Q(N, w) := \text{Run } BAR(N) \text{ outputting } S. \]
\[ \text{Run } T \text{ on } (S, w) \]

\[ R(w) := \text{Run } FOO_Q \text{ outputting } Q. \]
\[ \text{Run } BAR(Q) \text{ outputting } S. \]
\[ \text{Run } T \text{ on } (S, w) \]

**Claim:** \( S \) is a description of \( R \) itself!

\[ S(w) = \text{Run } FOO_Q \text{ outputting } Q. \]
\[ \text{Run } BAR(Q) \text{ outputting } S. \]
\[ \text{Run } T \text{ on } (S, w) \]

\[ \text{Therefore } R(w) = T(R, w) \]
For every computable \( t \), there is a computable \( r \) such that \( r(w) = t(R,w) \) where \( R \) is a description of \( r \).

Suppose we can design a TM \( T \) of the form:

“On input \((x,w)\), do bla bla with \( x \),
    do bla bla bla bla with \( w \), etc. etc.”

We can then find a TM \( R \) with the behavior:

“On input \( w \), do bla bla with \( R \),
    do bla bla bla bla with \( w \), etc. etc.”

We can use the operation:

“Obtain your own description”

in Turing machine pseudocode!