Lecture 12: Oracles and Self-Reference
Midterms back at end of lecture!

Thanks for your feedback

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mean</td>
<td>72.81632653</td>
</tr>
<tr>
<td>stddev</td>
<td>14.7939648</td>
</tr>
<tr>
<td>median</td>
<td>75</td>
</tr>
</tbody>
</table>
## 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>1</td>
</tr>
<tr>
<td>Fewer proofs in lecture (more concepts)</td>
<td>4</td>
</tr>
<tr>
<td>Too fast recitation/lecture</td>
<td>7</td>
</tr>
<tr>
<td>Likes Ryan/slides/lectures</td>
<td>33</td>
</tr>
<tr>
<td>Dislikes Ryan/slides/lectures</td>
<td>0</td>
</tr>
<tr>
<td>Likes TAs and recitations</td>
<td>16</td>
</tr>
<tr>
<td>More lecture notes (besides lecs)</td>
<td>3</td>
</tr>
<tr>
<td>More Office Hours</td>
<td>3</td>
</tr>
<tr>
<td>More examples/practice in lec/rec</td>
<td>7</td>
</tr>
<tr>
<td>Likes HW</td>
<td>7</td>
</tr>
<tr>
<td>I dislike that I don’t have any criticism</td>
<td>3</td>
</tr>
<tr>
<td>Faster pset solutions/grades</td>
<td>4</td>
</tr>
<tr>
<td>Likes piazza+office hours</td>
<td>12</td>
</tr>
<tr>
<td>“Yeet ❤️ this class”</td>
<td>1</td>
</tr>
</tbody>
</table>
Recognizability via Logic

Def. A decidable predicate $R(x,y)$ is a logical proposition about input strings $x, y \in \Sigma^*$ that is implementable by some TM $M$. 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

Think of $R$ as a function: $R: \Sigma^* \times \Sigma^* \rightarrow \{\text{True}, \text{False}\}$

EXAMPLES: $R(x,y) =$ “$xy$ has at most 100 zeroes”
$R(N,y) =$ “TM $N$ halts on $y$ in at most 99 steps”
are both decidable predicates
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

Let $M$ be a TM implementing $R$.

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

If $M(x,y)$ accepts, 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:  \( L = \{ M \mid M \text{ accepts at least one string} \} \) is recognizable.

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

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

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 \(A_{TM}\)?

Yes!

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 $q_? \in Q$ 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 so that it can always check the condition $(z \in B)$ 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 \textit{oracle TM} M with oracle B that accepts strings in A and rejects strings not in A.

Language A "Turing-Reduces" to B

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

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 REJECT.
HALT\textsubscript{TM} is decidable with A\textsubscript{TM} (HALT\textsubscript{TM} \leq_T A\textsubscript{TM})

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

1. If (M,w) is in A\textsubscript{TM} then ACCEPT

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

3. REJECT
\[ \leq_T \text{ versus } \leq_m \]

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

**Proof (Sketch):**

\( A \leq_m B \) means there is a computable function 
\[ f : \Sigma^* \rightarrow \Sigma^*, \text{ where for every } w, \]
\[ w \in A \iff f(w) \in B \]

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

**Theorem:** \( \neg \text{HALT}_{TM} \leq_T \text{HALT}_{TM} \)

\( D(M,w) : \text{ If } ((M,w) \text{ in } \text{HALT}_{TM}) \text{ then reject else accept} \)

**Theorem:** \( \neg \text{HALT}_{TM} \not\leq_m \text{HALT}_{TM} \quad \text{Why?} \)
Limitations on Oracle TMs!

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

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

*We can use the original proof by diagonalization!*

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

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

\( (D \text{ uses a HALT oracle to simulate } H) \)

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

*(by assumption on \( H) \quad (by \text{def of } D) \)*
Limitations on Oracle TMs!

There is an infinite hierarchy of unsolvable problems!

*Given ANY oracle A, there is always a harder problem that cannot be decided with that oracle A*

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

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

\[
\text{SUPERHALT}^n = \{ (M,x) \mid M, \text{ with an oracle for } \text{SUPERHALT}^{n-1}, \text{ halts on } x \}
\]
ORACLE
A Puzzle About Oracles

Given three instances
\((M_1, w_1), (M_2, w_2), (M_3, w_3)\)
of the Halting Problem,

It’s easy to decide all three of them,
using three oracle calls to HALT.

Can you decide \((M_i, w_i) \in \text{HALT}\) for all \(i\),
with only \text{TWO} oracle calls to 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 $Q$:

Actual TM

String encoding a TM
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
Another Way of Looking At It

Suppose in general we want to design a program that prints its own description. How?

“Print this sentence.”

Print two copies of the following, the second copy in quotes:

“Print two copies of the following, the second copy in quotes:”

\[ P_B \xrightarrow{w} B \xrightarrow{B} B \]

\[ = B \]

\[ = P_B \]
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 $M$:

$\text{Define } R$: What is $S$?

$\text{What is } S?$

$t(S,w)$
Proof: \((a,b) \rightarrow T \rightarrow t(a,b)\)

Define \(M\):

\[
\begin{align*}
N & \rightarrow \quad B \\
\text{C} \quad & \rightarrow \quad T \\
\text{w} \quad & \rightarrow \quad M
\end{align*}
\]

Define \(R\):

\[
\begin{align*}
w & \rightarrow \quad P_M \\
M & \rightarrow \quad B \\
S & \rightarrow \quad T \\
t(S,w) & \rightarrow \quad t(S,w)
\end{align*}
\]

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

Define \(R\):

\(S = C = R.\)  \(\therefore\) QED
$\text{FOO}_x(y) := \text{Output } x \text{ and halt.}$

$\text{BAR}(M) := \text{Output } \text{“} N(w) = \text{Run } \text{FOO}_M \text{ outputting } M. \text{ Run } M \text{ on } (M, w) \text{”}$

$\text{Q}(N, w) := \text{Run } \text{BAR}(N) \text{ outputting } S.$

Run $T$ on $(S, w)$

$\text{R}(w) := \text{Run } \text{FOO}_Q \text{ outputting } Q.$

Run $\text{BAR}(Q)$ outputting $S$.

Run $T$ on $(S, x)$

**Claim:** $S$ is a description of $R$ itself!

$S(w) = \text{Run } \text{FOO}_Q \text{ outputting } Q.$

Run $Q$ on $(Q, w)$
$\text{FOO}_x(y) := \text{Output } x \text{ and halt.}$

$\text{BAR}(M) := \text{Output } \text{``N}(w) = \text{Run } \text{FOO}_M \text{ outputting } M. \text{ Run } M \text{ on } (M, w)\text{''}$

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

$R(w) := \text{Run } \text{FOO}_Q \text{ outputting } Q. \text{ Run } \text{BAR}(Q) \text{ outputting } S. \text{ Run } T \text{ on } (S, x)$

Claim: $S$ is a description of $R$ itself!

$S(w) = \text{Run } \text{FOO}_Q \text{ outputting } Q. \text{ Run } \text{BAR}(Q) \text{ outputting } S. \text{ Run } T \text{ on } (S, w)$

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 a TM computing $r$.

**Moral:** 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 always find a TM $R$ with the behavior:

"On input $w$, do bla bla bla with code of $R$, do bla bla bla bla with $w$, etc. etc."

We can use the operation:

"Obtain your own description" in Turing machine pseudocode!