Announcements

1. Prelim 2 solutions posted on CMS. Re-grade requests due a week from now.
2. Problem Set 9 will be released Tuesday, 5/4, due Thursday, 5/13. This will be the last 4820 homework this spring.

Theorem. Let \( M_0, M_1 \) be two Turing machines such that \( L(M_0) \subseteq L(M_1) \). Then there is no Turing machine that accepts every \( x \) such that \( L(M_x) = L(M_0) \) or does not accept any \( x \) such that \( L(M_x) = L(M_1) \).

Proof. By contradiction. Suppose \( M \) accomplishes a and b. We need to show how to use \( M \) to solve co-HP.
Accordingly, given an input $x \# y$, we want to write down a description of a TM $T$ such that:

- if $M_x$ halts on $y$ then $L(M_x) = L(M_1)$;
- if $M_x$ doesn't halt on $y$ then $L(M_x) = L(M_0)$.

Idea: $M_x$ processes its input $w$ while simultaneously, on an extra set of working tapes, running a universal TM on $x \# y$.

As long as the UTM $(x \# y)$ has not halted, $M_x$ simulates $M_0$ processing $w$.

The moment UTM $(x \# y)$ halts, $M_x$ switches to simulating $M_1$ processing $w$.

Pseudocode:

1. On Tape 2, write the string $x \# y$.
2. Set status = false ("status" indicates if UTM $(x \# y)$ has halted)
3. Copy input from Tape 1 to Tape 3.
4. while status = false
   run one transition of $M_0$ on Tape 3
   run one transition of UTM on Tape 2.
if UTN halted, set status = true
if $M_0$ accepted on Tape 3
   Enter accept state
5. Clear Tape 3, Make fresh copy of Tape 1.
6. repeat forever:
   run one transition of $M_1$ on Tape 3.
   if $M_1$ accepts, enter accept state.

Some observations
1. If $M_x$ halts on $y$, then $UTN(x \# y)$ halts, so status is eventually set to "true"
   That means $M_x$ accepts its input if and only if one of the following happens:
   - $M_0$ accepts input before UTN halts.
   - $M_1$ accepts input.
   That means $L(M_x)$ is the union of $L(M_1)$ with a subset of $L(M_0)$. But $L(M_0) \subseteq L(M_1)$
   so $L(M_x) = L(M_1)$. 
2. If \( M_x \) doesn't halt on \( y \), then \( \text{UTM}(x \# y) \) loops, so the while-loop in step 4 never exits.
That means \( M_2 \) accepts its input if and only if \( M_0 \) accepts it.
\[ \implies L(M_2) = L(M_0). \]

Now assume there exists a \( M \) that accepts all \( z \) s.t. \( L(M_2) = L(M_0) \) and doesn't accept any \( z \) s.t. \( L(M_2) \neq L(M_1) \).

Then \( \text{co-HP} \) would be r.e. because the following Turing machine, \( M' \), would have \( L(M') = \text{co-HP} \).

\( M' \) on input \( x \# y \), constructs a description of the Turing machine \( M_2 \) defined above, and then runs machine \( M \) on this description, \( z \).

We reached a contradiction, hence no \( M \) that satisfies (5) and (6) above.
Recall, a **language** is a set of finite strings. A **property** of languages is just a function from languages to \{TRUE, FALSE\}. A property is **non-trivial** if it is true for at least one r.e. language and false for at least one r.e. language.

**Rice's Theorem.** For any non-trivial property of r.e. languages, it is undecidable to test, given \( \mathcal{X} \), whether \( L(M_\mathcal{X}) \) has that property.

**Proof.** Assume WLOG \( \emptyset \) has the property. (Otherwise replace the property with its negation.) Let \( L(M_\emptyset) = \emptyset \). Let \( L(M_\mathcal{X}) \) be any r.e. language that doesn't satisfy the property. If the property is decidable then \( \exists \) a Turing machine \( M \) that accepts \( \mathcal{X} \) if and only if \( L(M_\mathcal{X}) \) has the property. In particular \( M \) accepts all \( x \) s.t. \( L(M_\mathcal{X}) = L(M_0) = \emptyset \).
and \( M \) doesn't accept any \( x \)

such that \( L(M_x) = L(M) \).

Since \( L(M_0) \neq \emptyset \subseteq L(M) \),

the theorem above tells us

there is no such \( M \).

Examples: The following properties of a

Turing machine \( M_x \) with description \( \omega \)

are undecidable.

1. \( M_x \) accepts the string "M".
2. \( M_x \) accepts at least one string.
3. \( M_x \) accepts more strings of

   length 4820 than of length 3110.
4. Every string accepted by \( M_x \)

   has at most 4820 1's in it.

Some of these are r.e., others are not.

"Undecidable" just means

\( \{ x \mid L(M_x) \text{ has the property} \} \)

is not recursive.

Rice's Theorem alone can't tell you

if that set is r.e.

The first theorem in this lecture

often can tell you about r.e., as well.
Approx Algorithms: coping with NP-Hard optimization problems (like minimum vertex cover) by designing algorithms that may produce suboptimal solutions, but they are probably not too much worse than optimal.

Example: Minimum Vertex Cover is the problem: given undirected $G = (V, E)$ find a vertex set $C \subseteq V$ that contains an endpoint of each edge and has as few vertices as possible.

This is NP-Hard.

Algorithm for approximate the min VC.

1. Initialize $\tilde{E} = E$ (edge set of G).
2. Initialize $C = \emptyset$
3. While $\tilde{E}$ contains an edge $(u, v)$:
   
   $C = C \cup \{u, v\}$
   
   Delete from $\tilde{E}$ every edge having
4. output $C$. 

**Theorem.** This outputs a vertex cover $C$, s.t.

$$|C| \leq 2 \cdot |C^*|$$

where $C^*$ is a minimum vertex cover.

**Proof.** Let $e_1 = (u_1, v_1), \ldots, e_k = (u_k, v_k)$ be the edges chosen in the $k$ iterations of line 3. 

$$|C| = 2k.$$ 

Since $C^*$ is a vertex cover, it contains an endpoint of $e_i$ for $i = 1, \ldots, k$. The endpoints of these edges are all distinct. So 

$$|C^*| \geq k.$$ 

Therefore 

$$|C| \leq 2 \cdot |C^*|.$$ 

$C$ is a vertex cover because an edge isn't deleted from $E$ until it has an endpoint in $C$, and the algo doesn't until $E$ is empty.