Harvard CS 121 and CSCI E-207
Lecture 17: Undecidability

Salil Vadhan

November 1, 2012

• Reading: Sipser §4.2, §5.1.
Motivation

- **Goal**: to find an explicit undecidable language

- By the Church–Turing thesis, such a language has a membership problem that cannot be solved by any kind of algorithm

- We know such languages exist, by a counting argument.
  - Every recursive language is decided by a TM
  - There are only countably many TMs
  - There are uncountably many languages

∴ Most languages are not recursive (or even r.e.)
Is every Turing-recognizable set decidable?

This would be true if there were an algorithm to solve

The Acceptance Problem:

Given a TM $M$ and an input $w$, does $M$ accept input $w$?

Formally, $A_{TM} = \{\langle M, w \rangle : M \text{ accepts } w \}$. 
Completeness of $A_{TM}$

**Proposition:** If $A_{TM}$ is recursive, then every r.e. language is recursive.

“A$_{TM}$ is the hardest r.e. language.”

- $A_{TM}$ is said to be *r.e.-complete*, that is, it is a problem
  (a) that is r.e. and
  (b) to which every r.e. problem is reducible

**Proof:**
A simplifying detail: every string represents some TM

• Let $\Sigma$ be the alphabet over which TMs are represented (that is, $\langle M \rangle \in \Sigma^*$ for any TM $M$)

• Let $w \in \Sigma^*$

• if $w = \langle M \rangle$ for some TM $M$ then $w$ represents $M$

• Otherwise $w$ represents some fixed TM $M_0$ (say the simplest possible TM).
**Thm: $A_{TM}$ is not recursive**

- Look at $A_{TM}$ as a table answering every question:

<table>
<thead>
<tr>
<th>$w_0$</th>
<th>$w_1$</th>
<th>$w_2$</th>
<th>$w_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_0$</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>
  | $M_1$ | Y    | Y    | N    | N    | (WLOG assume every string $w_i$)
  | $M_2$ | N    | N    | N    | N    |
  | $M_3$ | Y    | Y    | Y    | Y    |

- Entry matching $(M_i, w_j)$ is $Y$ iff $M_i$ accepts $w_j$

- If $A_{TM}$ were recursive, then so would be the diagonal $D$ and its complement.

  - $D = \{w_i : M_i \text{ accepts } w_i\}$.
  - $\overline{D} = \{w_i : M_i \text{ does not accept } w_i\}$.

- But $\overline{D}$ differs from every row, i.e. it differs from every r.e. language. $\Rightarrow \Leftarrow$. 
Unfolding the Diagonalization

• Suppose for contradiction that $A_{TM}$ were recursive.

• Then there is a TM $M^*$ that decides $D = \{\langle M \rangle : M \text{ does not accept } \langle M \rangle \}$:

• Run $M^*$ on its own description $\langle M^* \rangle$.

• Does it accept?
  $M^*$ accepts $\langle M^* \rangle$

  $\iff \langle M^* \rangle \in \overline{D}$

  $\iff M^*$ does not accept $\langle M^* \rangle$.

• Contradiction!
Alan Mathison Turing (1912-1954)

24 years old when he published *On computable numbers* . . .
Some More Undecidable Problems About TMs

• The Halting Problem: Given $M$ and $w$, does $M$ halt on input $w$?

Proof:

Suppose $\text{HALT}_{\text{TM}} = \{\langle M, w \rangle : M \text{ halts on } w\}$ were decided by some TM $H$.

Then we could use $H$ to decide $A_{\text{TM}}$ as follows.

On input $\langle M, w \rangle$,

• Modify $M$ so that whenever it is about to go into $q_{\text{reject}}$, it instead goes into an infinite loop. Call the resulting TM $M'$.

• Run $H(\langle M', w \rangle)$ and do the same.

Note that $M'$ halts on $w$ iff $M$ accepts $w$, so this is indeed a decider for $A_{\text{TM}}$. $\Rightarrow\Leftarrow$. 
Undecidable Problems, Continued

• For a certain fixed $M_0$:
  
  Given $w$, does $M_0$ halt on input $w$?

What about:

• For a fixed $M_0$ and a fixed $w_0$, does $M_0$ halt on input $w_0$?
Further Undecidable Problems

• Given $M$, does $M$ halt on the empty string?

   Proof by reduction:
“Co-X”

- For any property X that a set might have, a set $S$ is co-X iff $\overline{S}$ has property X.

- For example, a co-finite set of natural numbers is a set that is missing only a finite number of elements.

- A co-regular language is . . . ?

- A co-recursive language is . . . ?

- What about a co-CF language?

- We proved earlier today:
  - A language is recursive if and only if it is both r.e. and co-r.e.
Non-r.e. Languages

Theorem: The following languages are not r.e.:

- $\overline{A_{TM}} = \{ \langle M, w \rangle : M \text{ does not accept } w \}$
- $\overline{HALT_{TM}} = \{ \langle M, w \rangle : M \text{ does not halt on } w \}$
- $\overline{HALT_{TM}^\epsilon} = \{ \langle M \rangle : M \text{ does not halt on } \epsilon \}$

Proof:
Formalizing the Notion of Reduction

- \( L_1 \) “reduces” to \( L_2 \) if we can use a “black box” for \( L_2 \) to build an algorithm for \( L_1 \).

- A function \( f : \Sigma_1^* \rightarrow \Sigma_2^* \) is computable if there is a Turing machine that for every input \( w \in \Sigma_1^* \), \( M \) halts with just \( f(w) \) on its tape.

- A (mapping) reduction of \( L_1 \subseteq \Sigma_1^* \) to \( L_2 \subseteq \Sigma_2^* \) is a computable function

  \[ f : \Sigma_1^* \rightarrow \Sigma_2^* \]

  such that, for any \( w \in \Sigma^* \),

  \[ w \in L_1 \iff f(w) \in L_2 \]

We write \( L_1 \leq_m L_2 \).
Properties of Reducibility

Lemma: If \( L_1 \leq_m L_2 \), then

- if \( L_2 \) is decidable (resp., r.e.), then so is \( L_1 \);
- if \( L_1 \) is undecidable (resp., non-r.e.), then so is \( L_2 \).
Examples of Reductions from This Lecture

- For every Turing-recognizable $L$, $L \leq_m A_{TM}$.

- $A_{TM} \leq_m \text{HALT}_{TM}$.

- $\text{HALT}_{TM} \leq_m \text{HALT}_{TM}^\varepsilon$. 