Lecture 19. The Halting Problem

The first unsolvable problem

- The Turing machine is a finite representation (i.e., it can be encoded as a finite string over a finite alphabet) of a language.
- Hence, there are only countably many TMs (i.e., countably many recursive languages and r.e. languages over any finite alphabet).
- But, there are uncountably many languages over any finite alphabet.
- Hence, some languages are not recursive (i.e., some decision problems are undecidable), and some languages are not even r.e.

The halting problem

The halting problem is one of the first problems proved to be undecidable:

> *Given an* arbitrary *Turing machine M and an* arbitrary *input string w, does M halt on w?*

The descision problem is equivalent to asking if a given string is in the following language:

$$H = \{\text{“}M\text{”}\ \text{“}w\text{”} : \text{Turing machine } M \text{ halts on input string } w\}$$

We will prove that the language H is not recursive.

Note that H is recursively enumerable since the universal Turing machine U semidecides it (U halts on input “M” “w” whenever M halts on w.)

Before proving that H is not recursive, we first prove a 'simpler' problem H_1 is not recursive and then apply a *reduction technique* (reducing H_1 to H) to prove H is not recursive.
Let

$$H_1 = \{\text{“}M\text{”} : \text{ Turing machine } M \text{ halts on input string “}M\text{”}\}$$

Then,

$$\overline{H_1} = \{\ x : x \text{ is not an encoding of any Turing machine}$$
$$\text{or}$$
$$x \text{ is an encoding of a Turing machine } M$$
$$\text{that does not halt on “}M\text{”}\ \}$$

Note that $\overline{H_1}$ is the "diagonal" language (see supplementary notes at the end).

We will show that $\overline{H_1}$ is not even recursively enumerable! Then,

- $\Rightarrow \overline{H_1}$ is not recursive.
- $\Rightarrow H_1$ is not recursive (closure under complementation).

Note that H_1 is recursively enumerable since the UTM semidecides H_1.

Theorem 1 *$\overline{H_1}$ is not recursively enumerable.*

Proof:

- Suppose $\overline{H_1}$ is recursively enumerable.
- By the definition of r.e. language, there exists a Turing machine M^* that semidecides $\overline{H_1}$.
- Is "M^*" $\in \overline{H_1}$?
  - If "M^*" $\in \overline{H_1}$,
    then by the definition of $\overline{H_1}$, M^* does not halt on "M^*".
    But, since M^* semidecides $\overline{H_1}$ (i.e., it halts on all strings in $\overline{H_1}$), "M^*" $\in \overline{H_1}$ implies M^* halts on "M^*".
    Contradiction.
  - If "M^*" $\notin \overline{H_1}$,
    then by the definition of $\overline{H_1}$, M^* halts on "M^*".
    But, since M^* semidecides $\overline{H_1}$, "M^*" $\notin \overline{H_1}$ implies M^* does not halt on "M^*".
    Contradiction.
- Hence $\overline{H_1}$ is not r.e.

After proving that H_1 is not recursive, we are now ready to prove H is not recursive by reduction.

Theorem 2 *The halting problem is undecidable, i.e.*

$H = \{\text{“}M\text{”}\ \text{“}w\text{”} : \ \textit{Turing machine } M \textit{ halts on input string } w\}$

is not recursive.

Proof idea: reduction from H_1 to H

- Suppose that H is recursive.
- By the definition of recursive language, there exists a Turing machine M_H that decides H.
- Using M_H as a subroutine, we would then be able to construct a new Turing machine that decides H_1!
- Contradicts the proven fact that H_1 is not recursive. Hence, H is not recursive.

Proof of the halting problem:

- Suppose that H is recursive.
- Then there exists a Turing machine M_H that decides H:

$$M_H(\text{“}M\text{”}\,\text{“}w\text{”}) = \begin{cases} \mathsf{y} & \text{if } M \text{ halts on } w \\ \mathsf{n} & \text{if “}M\text{” is not an encoding of any TM} \\ & \text{or } M \text{ does not halt on } w \end{cases}$$

- We would be able to construct the following TM M_{H_1} that decides H_1:

  M_{H_1}: On input string x, Run M_H on x“x”.
  If M_H halts at y (i.e., M halts on “M”), we let M_{H_1} halts at y.
  If M_H halts at n, we let M_{H_1} also halts at n.

- This contradicts to the non-existence of M_{H_1}.
  Hence, H is not recursive. $\square$

Verify the correctness of M_{H_1}

- If $x \in H_1$,
  i.e., $x =$ "M" where M is a TM that halts on "M".
  Then M_H on input x"x" will halt at y.
  So M_{H_1} will also halt at y.
- If $x \notin H_1$,
  - case (i): x is not an encoding of any Turing machine.
    Then, M_H will halt at n, and so does M_{H_1}.
  - case (ii): $x =$ "M" where M is a TM that does not halt on "M".
    Then, M_H on input x"x" will halt at n. So M_{H_1} will also halt at n.

Theorem 3 *Not every r.e. language is recursive*

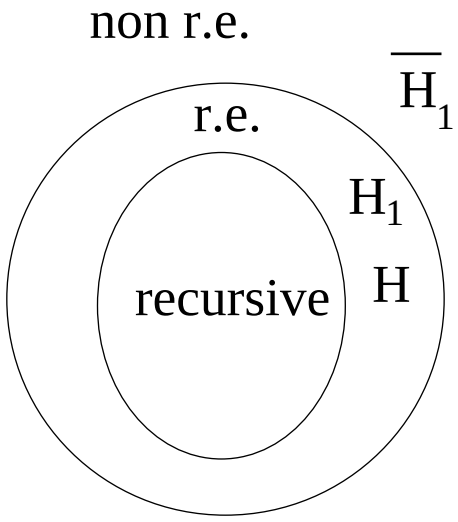
Example: H is r.e., but not recursive.

Theorem 4 *The class of r.e. languages is not closed under complementation.*

Example: H_1 is r.e., but $\overline{H_1}$ is not r.e.

Theorem 5 Not every language is r.e.

Example: $\overline{H_1}$ is not r.e.



Supplementary Notes

Where is the diagonalization in the proof of $\overline{H_1}$ being not r.e.?

Let Σ_{TM} be the alphabet used to encode Turing machines. Let $\{x_1, x_2, x_3, ...\}$ denote the set of all strings over Σ_{TM}. Note that some x_i are not legal encodings of Turing machines.

Let $c^{-1}(x)$ denotes the object decoded from x; it is a TM if x is a legal encoding of TM.

	x_1	x_2	x_3	x_4	$\cdots$
$c^{-1}(x_1)$					
$c^{-1}(x_2)$					
$c^{-1}(x_3)$					
$c^{-1}(x_4)$					
$\vdots$					

Write *halt* in the (i, j) entry if and only if x_i is the encoding of a Turing machine that accepts string x_j.
Let's say

	x_1	x_2	x_3	x_4	$\cdots$
$c^{-1}(x_1)$	$\times$	$\times$	$\times$	$\times$	
$c^{-1}(x_2) = M_1$	halt	not halt	halt	halt	
$c^{-1}(x_3) = M_2$	not halt	halt	halt	not halt	
$c^{-1}(x_4)$	$\times$	$\times$	$\times$	$\times$	
$\vdots$					

Now, construct a language (row) corresponding to the complement of the diagonal:

	x_1	x_2	x_3	x_4	$\cdots$
$c^{-1}(x_1)$	$\times$	$\times$	$\times$	$\times$	
$c^{-1}(x_2) = M_1$	halt	not halt	halt	halt	
$c^{-1}(x_3) = M_2$	not halt	halt	halt	not halt	
$c^{-1}(x_4)$	$\times$	$\times$	$\times$	$\times$	
$\vdots$					

Construct the diagonal language:

$$
\begin{aligned}
L_{diag} &= \{\ x_i\text{: the } (i,i) \text{ entry} \neq \text{halt}\} \\
&= \{\ x : x \text{ is not an encoding of Turing machine} \\
&\qquad\qquad\qquad\qquad \text{or} \\
&\qquad\quad x \text{ is the encoding of a Turing machine } M \\
&\qquad\quad \text{that does not halt on ``}M\text{''}\ \} \\
&= \overline{H_1}!
\end{aligned}
$$

By diagonalization principle, there is no TM (none of the rows in the table) that accept $\overline{H_1}$ (i.e., halts at x if x is not an encoding of TM or if x is an encoding of a TM M that does not halt on "M").

Therefore, $\overline{H_1}$ is not recursively enumerable.