COSE215: Theory of Computation

Lecture 16 — Undecidability

Hakjoo Oh

2016 Spring

Recursively Enumerable Languages

Definition

A language L is *recursively enumerable* (RE) if there exists a Turing machine that accepts it.

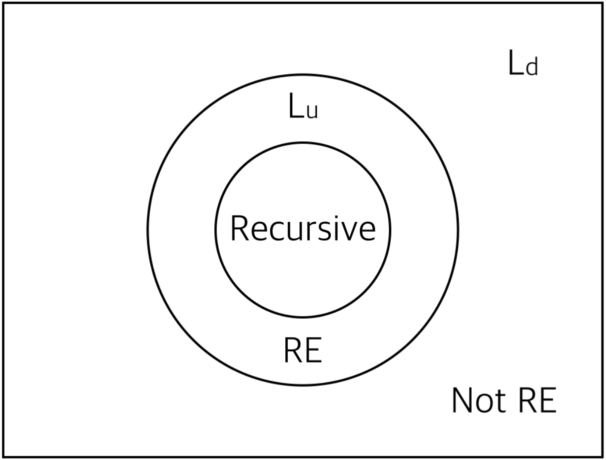
$$L \text{ is RE} \;\Leftrightarrow\; \exists M \in TM.\, \forall w \in L.\; q_0 w \vdash^* x_1 q_f x_2$$

Recursive Languages (Decidable Languages)

Definition

A language L is *recursive* if there exists a Turing machine that accepts it and always terminates.

Overview



L_d: A language that is not recursively enumerable

$$L_d = \{w_i \mid w_i \notin L(M_i)\}$$

Preliminary steps:

1. Enumerating binary strings
2. Representing Turing machines in binary strings

Enumerating Binary Strings

- A binary string can be represented by a unique integer i:

  The integer for binary string w is the integer value of $1w$.

- w_i: the ith binary string

$$
\begin{array}{rcl}
w_1 & = & \epsilon \\
w_2 & = & 0 \\
w_3 & = & 1 \\
w_4 & = & 00 \\
w_5 & = & 01 \\
w_6 & = & 10 \\
w_7 & = & 11 \\
w_8 & = & 000 \\
w_9 & = & 001 \\
 & \vdots &
\end{array}
$$

Representing Turing Machines as Binary Strings

$$M = (Q, \{0, 1\}, \Gamma, \delta, q_1, B, F)$$

1. $Q = \{q_1, q_2, \ldots, q_r\}$
2. $\Gamma = \{X_1, X_2, X_3, \ldots, X_s\}$
3. Directions : $\{D_1, D_2\}$

Encoding for the transition function $\delta(q_i, X_j) = (q_k, X_l, D_m)$:

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Encoding for the Turing machine M:

$$C_1 11 C_2 11 \cdots C_{n-1} 11 C_n$$

(C_i: encoding for the ith transition rule of M).

Example

$$M = (\{q_1, q_2, q_3\}, \{0, 1\}, \{0, 1, B\}, \delta, q_1, B, \{q_2\})$$

$$\begin{array}{rcll}
\delta(q_1, 1) & = & (q_3, 0, R), & 0100100010100 \\
\delta(q_3, 0) & = & (q_1, 1, R), & 0001010100100 \\
\delta(q_3, 1) & = & (q_2, 0, R), & 00010010010100 \\
\delta(q_3, B) & = & (q_3, 1, L), & 0001000100010010
\end{array}$$

Encoding in binary string:

$$0100100010100110001010100100110001001001010011000100010001001 0$$

Turing machines can be ordered

M_i: The ith Turing machine

Definition

We define M_i to be the Turing machine whose binary representation is w_i.

Turing machines can be ordered

M_i: The ith Turing machine

Definition

We define M_i to be the Turing machine whose binary representation is w_i.

When M_i is not a valid Turing machine, define M_i to be a Turing machine with one state and no transitions, e.g., M_1.

The Diagonalization Language

Definition

$L_d = \{w_i \mid w_i \notin L(M_i)\}$

Theorem

L_d *is not a recursively enumerable language.*

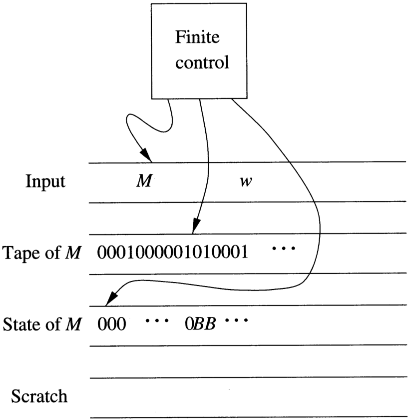
$$
\begin{array}{cc|ccccc}
 & & & & j & & \\
 & & \mathbf{1} & \mathbf{2} & \mathbf{3} & \mathbf{4} & \cdots \\
\hline
 & \mathbf{1} & 0 & 1 & 1 & 0 & \cdots \\
i & \mathbf{2} & 1 & 1 & 0 & 0 & \cdots \\
 & \mathbf{3} & 0 & 0 & 1 & 1 & \cdots \\
 & \mathbf{4} & 0 & 1 & 0 & 1 & \cdots \\
 & \vdots & \vdots & \vdots & \vdots & \vdots &
\end{array}
$$

L_u: A language that is RE but not recursive

$$L_u = \{(M, w) \mid w \in L(M)\}$$

L_u is recursively enumerable

The Turing machine that accepts $L_u = \{(M, w) \mid w \in L(M)\}$:

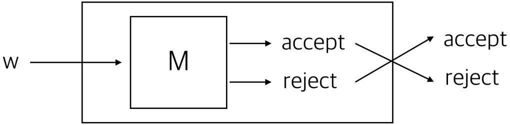


"Universal Turing Machine"

Properties of complements (1)

Lemma

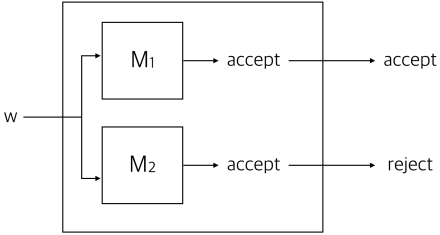
If L is a recursive language, then so is $\bar{L}$.



Properties of Complements (2)

Lemma

If both a language L and its complement are RE, then L is recursive.



L_u is not recursive

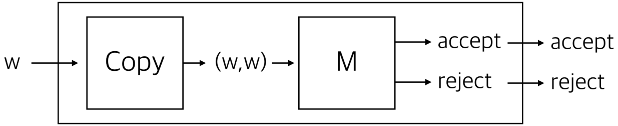
Theorem

L_u is RE but not recursive.

- Suppose L_u were recursive.
- Then by the property of complements, $\bar{L_u}$ is also recursive.
- However, if we have a TM M to accept $\bar{L_u}$, then we can construct a TM to accept L_d (explained next).
- We already know that L_d is not RE, contradiction.

Construction of TM to accept L_d from TM to accept $\bar{L_u}$

Suppose $L(M) = \bar{L_u}$. We construct M' s.t. $L(M') = L_d$ as follows:



Summary

We have

1. defined the class of recursively enumerable languages,
2. defined the class of recursive languages,
3. defined a non-recursively enumerable language L_d and prove it, and
4. defined a non-recursive language L_u and prove it.