Weekly Challenge 07: Recognizability and Decidability

CS 212 Nature of Computation
Habib University

Fall 2025

1. Countable machines

Turing-recognizable languages are often also called "recursively enumerable", meaning there exists a special variant of Turing machines called an enumerator that "enumerates" it. Enumerator are turing machines with an output "printer" that can output a string anytime. When an enumerator starts working regardless of the input it sequentially outputs every string in the language that it enumerates through the output printer. So if $L = 0^n1^n$ then the enumerator for L will one by one output 0^i1^i for each $i \in \mathbb{N}$. The advantage of enumerator comes with the following theorem:

Theorem 1 (Sipser theorem 3.21)**.** *A language L is a Turing-recognizable language iff there exists some enumerator that enumerates it.*

With the concept of enumerators the fact that any Turing recognizable language can be index by the set of natural numbers becomes evident. As every Turing recognizable language is countable, we can index them with the natural numbers so any Turing recognizable language L can be written as $L = \{w_0, w_1, w_2, w_3, \dots\}$ where each w_i is some string in L.

By building on these two ideas we can get a more interesting and useful Turing machine with the following theorem.

Theorem 2. *Let L be an infinite Turing-recognizable language, then there exists a decider M that on input $i \in \mathbb{N}$ halts with the i^{th} string in L on its tape such that $\forall i, j \in \mathbb{N}$ if $i \neq j$ if $M(i)$ halts with w_i on its tape and $M(j)$ halts with w_j on its tape then $w_i \neq w_j$.*

Proof. Let L be an infinite Turing-recognizable language, then we have and enumerator E that enumerates L. We construct the Turing machine M as follows:

$M =$ "On input $i \in \mathbb{N}$:

1. Run E on the blank tape until E outputs i distinct strings.
2. Write the i^{th} distinct string that E output on tape, and remove everything else from the tape then *accept*."

Here M is a decider that works as desired. □

Let L be a Turing-recognizable language consisting of descriptions of Turing machines,

$$L = \{\langle M_0\rangle, \langle M_1\rangle, \langle M_2\rangle, \dots\}$$

where every $M_i \in L$ is a decider. Prove that there exists some decidable language L', such that L' is not decided by any decider $M \in L$.

(Hint: for this problem you need not worry about the working of an enumerator, you can just use the machine described in Theorem 2.)

Solution: Considering L. Suppose we enumerate all machines in L as

$$L = \{\langle M_0 \rangle, \langle M_1 \rangle, \langle M_2 \rangle, \dots\}.$$

Also we enumerate all strings as

$$\Sigma^* = \{w_0, w_1, w_2, \dots\}.$$

Now, we define a language D as follows:

For a string w_i,

- If M_i **accepts** w_i, then w_i **does not belong** to the language D.
- If M_i **rejects** w_i, then w_i **belongs** to the language D.

Since each M_i is a decider, this procedure always halts, therefore, D is **decidable**.

Assume, for contradiction, that there exists some machine M_k from L decides D.
Then, by definition of D,

$$w_k \in D \iff M_k \text{ rejects } w_k.$$

However, if M_k correctly decides D, then it must accept exactly the strings in D, that means,

$$w_k \in D \iff M_k \text{ accepts } w_k.$$

The two statements above directly contradict each other, saying both ($w_k \in D$) and ($w_k \notin D$) simultaneously which can not occur.

Therefore, no decider in L can decide the language D.

References:

MIT OpenCourseWare. "Lecture8: Undecidability."18-404J Theory of Computation, Fall 2020. Massachusetts Institute of Technology, 2020. PDF available at: `https://ocw.mit.edu/courses/18-404j-theory-of-computation-fall-2020/3ab8452b4b29eb28a1416e4a1575323c_MIT18_404f20_lec8.pdf`