

Recap:

- Turing Machine: $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{acc}, q_{rej}, B)$, where

$$Q = \{q_0, \dots, q_k\};$$

$$\Gamma = \{\gamma_0, \dots, \gamma_j\};$$

$$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\};$$

$$q_{acc}, q_{rej} \in Q;$$

$$B \in \Gamma.$$

Remark: Any Turing Machine M can be represented with a string $\langle M \rangle$.

- Computational Enumerability(c.e.) and Computability

Definition 1. Language L is computationally enumerable (c.e.) if there exists Turing Machine M such that $\forall x$:

$$x \in L \Rightarrow M(x) \text{ accepts};$$

$$x \notin L \Rightarrow M(x) \text{ rejects or doesn't halt.}$$

Definition 2. Language L is computable if there exists Turing Machine M such that $\forall x$:

$$x \in L \Rightarrow M(x) \text{ accepts};$$

$$x \notin L \Rightarrow M(x) \text{ rejects.}$$

Remark:

1. There're a countable number of c.e. languages.
2. There're an uncountable number of languages.

Proof. Assume there're a countable number of languages and $\{L_1, L_2, L_3, \dots\}$ enumerates all of them. Let the diagonal language of the enumeration be L_D , we have $L_D \notin \{L_1, L_2, L_3, \dots\}$, and thus $\{L_1, L_2, L_3, \dots\}$ does not enumerate all languages, which contradicts the assumption. \square

3. If a language L is computable, then it is also c.e. The reverse does not hold.

Universal Turing Machine:

Definition 3. Let $\langle M \rangle$ be the string that describes any Turing Machine M , Universal Turing Machine, denoted as $U(\langle M \rangle, x)$, is a Turing Machine such that $\forall M, x$:

$$M(x) \text{ accepts} \Leftrightarrow U(\langle M \rangle, x) \text{ accepts};$$

$$M(x) \text{ rejects} \Leftrightarrow U(\langle M \rangle, x) \text{ rejects};$$

$$M(x) \text{ doesn't halt} \Leftrightarrow U(\langle M \rangle, x) \text{ doesn't halt};$$

$L_u := L(U) = \{(\langle M \rangle, x) \mid M(x) \text{ accepts}\}$ is the language of Universal Turing Machine.

Remark: U can be constructed with a multi-tape Turing Machine.

Proposition 1. *There exist languages that are not c.e.*

Proof. Because there are an uncountable number of languages, but only countable number of c.e. languages, there must exist languages that are not c.e.

An example of non-c.e. language is

$$L_d = \{ \langle M \rangle \mid M(\langle M \rangle) \text{ does not accept} \}$$

We sketch the proof of non-c.e. of L_d as follows: suppose L_d is c.e., then there exists Turing Machine N such that $L_d = L(N)$. If $N(\langle N \rangle)$ accepts, then:

- From definition of L_d , $\langle N \rangle \notin L_d$;
- From definition of $L(N)$, $\langle N \rangle \in L(N)$.

which contradicts each other. Similarly if $N(\langle N \rangle)$ rejects, then:

- From definition of L_d , $\langle N \rangle \in L_d$;
- From definition of $L(N)$, $\langle N \rangle \notin L(N)$.

which also contradicts each other. Therefore L_d is not c.e.

Remark: Define $\bar{L} = \{x \mid x \notin L\}$, then \bar{L}_d is c.e. because $\bar{L}_d \in L(M)$. Furthermore, using Proposition 3 it is easily provable that \bar{L}_d is c.e. but not computable. \square

Proposition 2. *If L is computable, then \bar{L} is computable.*

Proof. Because L is computable, there exists Turing Machine M such that $\forall x$: if $x \in L$ then $M(x)$ accepts; if $x \notin L$ then $M(x)$ rejects. Define another Turing Machine M' that is the same as M except that it swaps q_{acc} and q_{rej} in M . Thus $\forall x$: if $x \notin \bar{L}$ then $M'(x)$ rejects; if $x \in \bar{L}$ then $M'(x)$ accepts. Therefore \bar{L} is also computable. \square

Proposition 3. *L and \bar{L} are both c.e. if and only if L is computable.*

Proof. If L is computable, then L is c.e. Because L is computable, \bar{L} is also computable and thus c.e., which proves sufficiency.

If L and \bar{L} are both c.e., there exist Turing Machines M and \bar{M} such that $\forall x$:

$$\begin{aligned} x \in L &\Rightarrow M(x) \text{ accepts, } \bar{M}(x) \text{ rejects or doesn't halt;} \\ x \notin L &\Rightarrow M(x) \text{ rejects or doesn't halt, } \bar{M}(x) \text{ accepts.} \end{aligned}$$

Now we create another Turing Machine M' such that $\forall x$: if $M(x)$ accepts then $M'(x)$ accepts; if $\bar{M}(x)$ accepts then $M'(x)$ rejects. M' can be constructed via dovetailing between M and \bar{M} . Rewrite above relationship in M' , we have:

$$\begin{aligned} x \in L &\Rightarrow M'(x) \text{ accepts;} \\ x \notin L &\Rightarrow M'(x) \text{ rejects.} \end{aligned}$$

Therefore L is computable. \square