Undecidable Problem about Turing Machine

In this section, we will discuss all the undecidable problems regarding turing machine. The reduction is used to prove whether given language is desirable or not. In this section, we will understand the concept of reduction first and then we will see an important theorem in this regard.

Reduction

Reduction is a technique in which if a problem P1 is reduced to a problem P2 then any solution of P2 solves P1. In general, if we have an algorithm to convert an instance of a problem P1 to an instance of a problem P2 that have the same answer then it is called as P1 reduced P2. Hence if P1 is not recursive then P2 is also not recursive. Similarly, if P1 is not recursively enumerable then P2 also is not recursively enumerable.

Theorem: if P1 is reduced to P2 then

1. If P1 is undecidable, then P2 is also undecidable.
2. If P1 is non-RE, then P2 is also non-RE.

1. Consider an instance w of P1. Then construct an algorithm such that the algorithm takes instance w as input and converts it into another instance x of P2. Then apply that algorithm to check whether x is in P2. If the algorithm answer 'yes' then that means x is in P2, similarly we can also say that w is in P1. Since we have obtained P2 after reduction of P1. Similarly if algorithm answer 'no' then x is not in P2, that also means w is not in P1. This proves that if P1 is undecidable, then P1 is also undecidable.

2. We assume that P1 is non-RE but P2 is RE. Now construct an algorithm to reduce P1 to P2, but by this algorithm, P2 will be recognized. That means there will be a Turing machine that says 'yes' if the input is P2 but may or may not halt for the input which is not in P2. As we know that one can convert an instance of w in P1 to an instance x in P2. Then apply a TM to check whether x is in P2. If x is accepted that also means w is accepted. This procedure describes a TM whose

language is P1 if w is in P1 then x is also in P2 and if w is not in P1 then x is also not in P2. This proves that if P1 is non-RE then P2 is also non-RE.

Empty and non empty languages:

There are two types of languages empty and non empty language. Let Le denotes an empty language, and Lne denotes non empty language. Let w be a binary string, and Mi be a TM. If L(Mj) = Φ then Mi does not accept input then w is in Le. Similarly, if L(Mj) is not the empty language, then w is in Lne.
Thus we can say that

Le = {M | L(M) = Φ}

Lne = {M | L(M) ≠ Φ}

Both Le and Lne are the complement of one another.