THEORY OF COMPUTATION

Subash Manandhar

Undecidability

- **Church Turing Thesis**
  - It states that every computation or algorithm can be carried out by a Turing machine.
  - This statement was first formulated by Alonzo Church.
  - The thesis might be replaced on saying that the notation of effective or mathematical method in logic and mathematics is captured by TM.
  - It is generally assumed that such methods must satisfy the following requirements:
    - The method consists of a finite set of simple and precise instruction that are described with a finite no. of symbols.
    - The method will always produce the result in a finite no. of steps.
    - The method can in principle be carried out by human being with only paper and pencil.
    - The execution of the method requires no intelligence of human being except that which is needed to understand and execute instruction.

Undecidability

- **Church Turing Thesis**
  - Invention of Turing machine has accumulated enough evidence to adopt this hypothesis.
  - It is believed that there is no function that can be defined by human, whose calculation can be described by any well defined mathematical algorithm that can not be computed by Turing machine.
  - Turing machine is believed to be ultimate calculating mechanism .
  - After adopting Church Turing Thesis, we can give precise meaning of term as
    - “An algorithm is a procedure that can be executed on Turing Machine.”

Undecidability

- **Universal Turing Machine**
  - If a Turing Machine is a sound model of computation it should be possible to demonstrate that it can act as a stored program machine, where the program is regarded as an input rather than hardwired.
  - We shall construct a Turing Machine M_U that takes as input a description of Turing Machine M and an input word X and simulates the computation of M on input X.
  - A machine such as M_U that can simulate the behavior of an arbitrary Turing Machine is called Universal Turing Machine.
  - Thus, we can describe universal Turing Machine T_U as a Turing Machine that on input <M , w> ; where M is a Turing Machine and w is a string, simulates computation of M on input w.
  - T_U accepts <M , w> iff M accepts w.
  - T_U rejects <M , w> iff M rejects w.

Undecidability

- **Encoding of Turing Machine**
  - Formulate a notational system where we can encode both an arbitrary Turing Machine T1 and an input string x over an arbitrary alphabet as string e(T1) and e(x) over some fixed alphabet.
  - This encoding must not destroy any information i.e. we must be able to reconstruct Turing Machine T1 and string x.
  - For encoding of Turing Machine, use alphabet {0 , 1}, although Turing Machine may have much larger alphabet.
  - Start by assigning positive integer to each state, each tape symbol and each of three directions in Turing Machine T1 we want to encode.
  - Here,

    we assume two fixed infinite sets $Q = \{q_1, q_2, \ldots\ldots\ldots\}$ and $S = \{a_1, a_2, \ldots\ldots\ldots\}$ so that for any Turing Machine $T = \{Q_1, \sum, \Gamma, \delta, q_0, B, F\}$; Q_1 is subset of Q and Γ is subset of S.

Undecidability

- **Encoding of Turing Machine**
  - We can represent a state or a symbol by a string of 0’s of appropriate length. Here 1’s are used as separators.
  - Once, we have established an integer to represent each state, symbol and direction, we can encoding the transition function δ.
  - Let, one transition rule is

    $\delta(q_i, a_j) = (q_k, a_l, D_m)$ for some integer I,j,k,l,m
  - Then, we can code this rule by string

    $S(q_i)\ 1\ S(a_j)\ 1\ S(q_k)\ 1\ S(a_l)\ 1\ S(D_m)$ say m_1 , where S is encoding function.
  - A code for entire Turing Machine T1 consists of all codes for transitions in some order, separated by pair of 1’s like $m_1\ 11\ m_2\ 11\ldots\ldots.m_n$
  - Now, code for Turing Machine and input string x will be formed by separating them by three consecutive 1’s. i.e. e(TM) 111 e(x)

Undecidability

- **Encoding of Turing Machine**
  - **Encoding functions**
  - First associate a string of 0’s to each states, each tape symbols and each of three directions.
  - Let, function S is defined as :
    - $S(B) = 0$
    - $S(a_i) = 0^{i+1}$ for each $a_i \in S$
    - $S(q_i) = 0^{i+2}$ for each $q_i \in Q$
    - $S(S) = 0$
    - $S(L) = 00$
    - $S(R) = 000$

Undecidability

- **Encoding of Turing Machine**
- **e.g.** consider an example where Turing Machine T is defined as T=({q_1, q_2, q_3}, {a , b}, {a ,b ,B}, δ ,q ,B ,F) where δ is defined as :

  $\delta(q_1, b) = (q_3, a, R)$

  $\delta(q_3, a) = (q_1, b, R)$

  $\delta(q_3, b) = (q_2, a, R)$

  $\delta(q_3, B) = (q_3, b, L)$

- Now, using encoding function S as defined above

  $S(q_1) = 000$

  $S(q_2) = 0000$

  $S(q_3) = 00000$

  $S(a_1) = 00$ say, $a_1 = a$ and $a_2 = b$

  $S(a_2) = 000$

  $S(B) = 0$

  $S(R) = 000$

  $S(L) = 00$

  $S(S) = 0$

Undecidability

- Now for $\delta(q_1, b) = (q_3, a, R)$ say m_1

  $e(m_1) = S(q_1)1S(b)1S(q_3)1S(a)1S(R)$

  $= 0001000100000100100$

- for $\delta(q_3, a) = (q_1, b, R)$ say m_2

  $e(m_2) = S(q_3)1S(a)1S(q_1)1S(b)1S(R)$

  $= 0000010010001000100$

- for $\delta(q_3, b) = (q_2, a, R)$ say m_3

  $e(m_3) = S(q_3)1S(b)1S(q_2)1S(a)1S(R)$

  $= 0000010001000100100$

- for $\delta(q_3, B) = (q_3, b, L)$ say m_4

  $e(m_4) = S(q_3)1S(B)1S(q_3)1S(b)1S(L)$

  $= 0000010100000100010$

Undecidability

- Now, code for Turing Machine T

  $e(T) = e(m_1)11e(m_2)11e(m_3)11e(m_4)$

  = 0001000100000100100011
  0000010010001000100011
  0000010001000100100011
  000001010000001000100
- For input string x where x = ab, code will be

  e(x) = S(a)1S(b) = 001000
- Now, code for Turing Machine T and input string x is

  e(T)111e(x) = 0001000100000100100011
  0000010010001000100011
  0000010001000100100011
  000001010000001000100111001000

Undecidability

- **The Halting Problem**
  - There are limits to the power of Turing Machine.
  - A Turing Machine continues until it reaches accept state or reject state where it will halt.
  - If it never reaches one , then it continues computing forever.
  - There exists problems that Turing Machine cannot solve.
  - The best known problem i.e. unsolvable by a Turing Machine is the halting problem.
  - Halting Problem is:
    - "Given an arbitrary Turing Machine T as input and equally arbitrary tape t, decide whether T halts on t "
    - "to determine for any arbitrary given Turing machine T_M and input w, whether T_M will eventually halts on input w."

Undecidability

- **Undecidable Problem about Turing Machine**
  - The problems for which no algorithms exists are called undecidable or unsolvable. e.g. halting problem
  - Following problems about Turing Machine are undecidable:
    - Given a Turing Machine M and input string w, does M halts on input w?
    - Given a Turing Machine M, does M halt on the empty tape?
    - Given a Turing Machine M, is there any string at all on which M halts?
    - Given a Turing Machine M, does M halts on every input string?
    - Given two Turing Machines M1 and M2, do they halt on same input string?

Undecidability

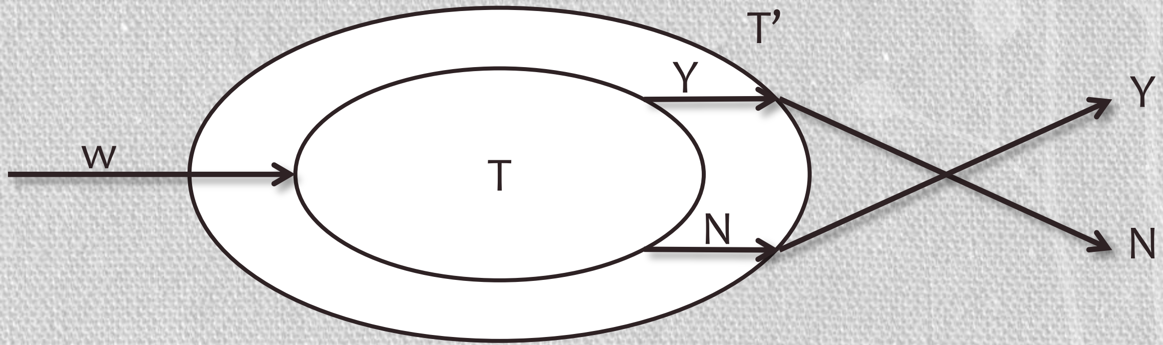
- **Language of Turing Machine**
  - If $M = (Q , \sum , \Gamma , \delta , q_0, B, F)$ is a TM, then language of TM L(M) is the set of strings w in $\sum^*$ such that $q_o w \vdash^* \alpha p \beta$ where $p \in F$ and α and β are any tape strings.
  - **Recursively Enumerable Language**
    - The set of Recursively Enumerable languages are precisely those language that can be accepted by TM.
    - Strings that are not in the language may be rejected or may cause TM to go into infinite loop.
  - **Recursive Language**
    - A language is recursive if there exists a TM that accept every string of the language and rejects strings that are not in language.
    - Recursive language always halts TM.

Undecidability

- **Turing Recognizable Language**
  - A language L is Turing Recognizable if there exists a TM such that for all strings w;
    - If w ϵ L , eventually TM enter q_{accept}
    - if w $\notin$ L , either TM enters q_{reject} or TM never terminates.
- **Turing Decidable Language**
  - A language L is Turing Decidable if there exists a TM such that for all strings w;
    - If w ϵ L , TM enter q_{accept}
    - if w $\notin$ L , TM enters q_{reject}
- *Hence, decidable language always terminates while recognizable language can run forever without deciding.*

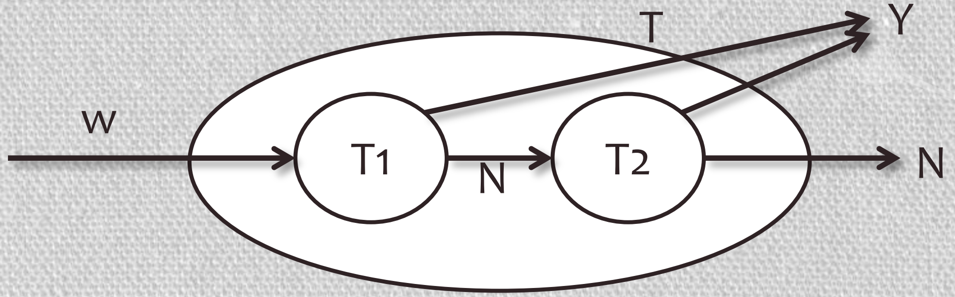
Undecidability

- **Properties of Recursive Language**
- **The complement of recursive language is recursive:**
  - Let, L be a recursive language.
  - T be a Turing Machine that halts on all inputs and accepts L.
  - Let us construct a T' from T so that if T enters a final state on input w, then T' halts without accepting.
  - If T halts without accepting, T' enters final state.
  - So, L(T') is the language accepted by T' is the complement of L.
  - Hence, the complement of recursive language is recursive.



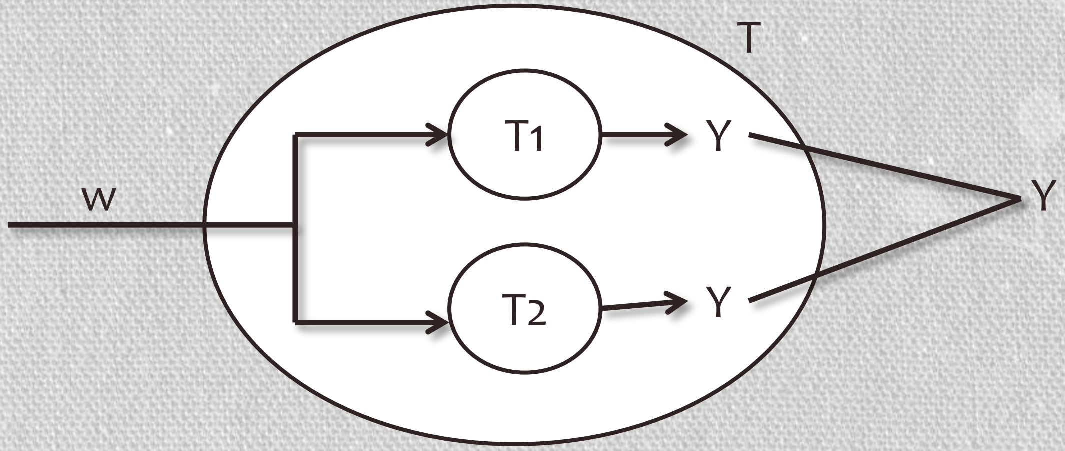
Undecidability

- **Properties of Recursive Language**
- **The union of two recursive languages is also recursive:**
  - Let, L1 and L2 be two recursive languages accepted by Turing machines T1 and T2.
  - Construct a Turing Machine T that first simulates T1
    - If T1 accepts than T accepts
    - It T1 rejects then T simulates T2 and accepts if and only if T2 accepts.
  - Here, T is guaranteed to halt because both T1 and T2 are algorithms i.e. T accepts L1 U L2
  - Hence, L1 U L2 is also recursive since there exists Turing Machine T for it.



Undecidability

- **Properties of Recursive Language**
- **The union of two recursively enumerable languages is also recursively enumerable:**
  - Let, L1 and L2 be two recursively enumerable languages accepted by Turing machines T1 and T2.
  - T be a Turing Machine that can simulate T1 and T2 simultaneously on separate tapes.
  - If either accepts then T accepts as follows



Undecidability

- **Properties of Recursive Language**
- **If a language L and its complement L’ are both recursively enumerable then L (and L’) is recursive:**
  - Let, L and L’ be two recursively enumerable languages accepted by Turing machines T1 and T2.
  - Construct a Turing Machine T that simulates T1 and T2 simultaneously.
  - T accepts w if T1 accepts and rejects w if T2 will accept.
  - Thus T will say either yes or no but not both.
  - Since, T is algorithm that accepts L, L is recursive (hence L’ also)