Recursive Enumerable and Recursive language.

*→ Turing decidable languages

A Turing machine M decides L if M computes the characteristic function

$$\chi_L(w) = \begin{cases} 0 & \text{if } w \notin L \\ 1 & \text{if } w \in L \end{cases}$$

* A language L is Turing decidable if there exist a TM M that accept L and that halt on every $w \in \Sigma^*$ producing output 1 if $w \in L$ and output 0 otherwise.

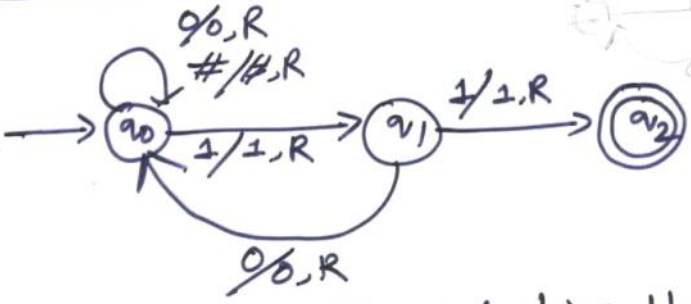
* Recursive Enumerable languages

A language L is recursive enumerable if $\exists$ a turing machine that accept L and either reject or loop forever for every word in the complement of $L(L)$

Recursive languages :

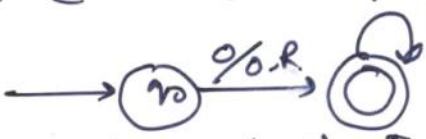
A language is recursive if ∃ a turing m/c that decides L

Example ① : TM for $(0+1)^*11(0+1)$



- All words with substring 11 are accepted
- All words without 11 and ending with 1 are rejected
- All words without 11 and ending in 0 or ∈ are in loop forever.

Example ② :



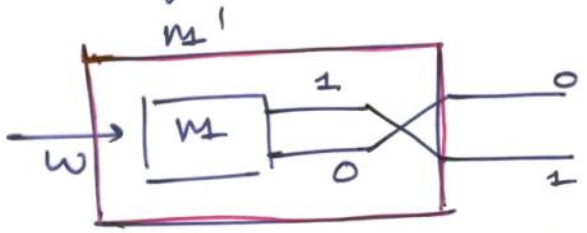
Accepts all words that starts with 0
rejects all words that do not

Note: Every recursive language is also recursive enumerable ~~because~~ but converse is not true.

Properties of r.e and regular languages

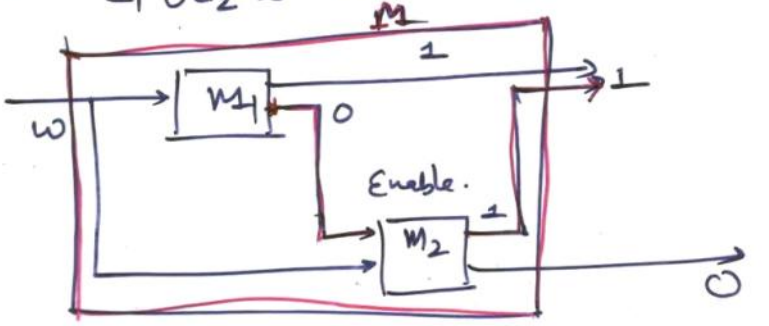
thm: If L is recursive, then its complement $\overline{L}$ is also recursive.

Proof: if L is recursive $\exists$ a TM decide L



Now m/c M_1' decides $\overline{L}$

thm: if L_1 and L_2 are Regrsive then $L_1 \cup L_2$ is recursive.



Thm: If L_1 and L_2 are two recursive enumerable languages then $L_1 \cup L_2$ is also recursive enumerable.

proof :- L_1 is accepted by TM M_1

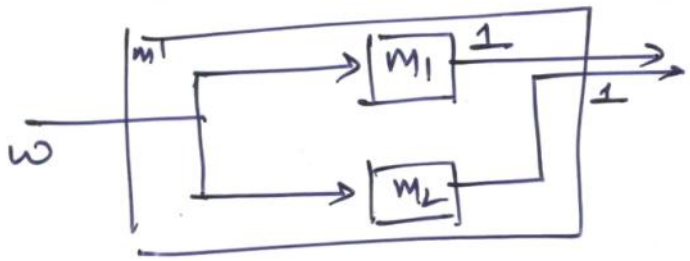
L_2 is accepted by TM M_2

It may be possible both M_1 and M_2 do not halt.

Construct a TM M such that M simultaneously simulates M_1 and M_2 on two tapes.

If any one of L_1 and L_2 is accepted this is also accepted by M.

$$L(M) = L_1 \cup L_2$$



Thm: if L_1 and L_2 are recursive enumerable then $L_1 \cap L_2$ is also recursive enumerable.

for $L_1 \longrightarrow M_1$

$L_2 \longrightarrow M_2$

Now build a new TM M that first runs the input string w on M_1 and then if string is accepted it runs the same input w on M_2

Thm: if L is recursive enumerable whose complement $\overline{L}$ is also recursive enumerable, then L is recursive.

$L \longrightarrow M_1$.

$M_2 \longrightarrow \overline{L}$

M will always halt with o/p 0 or 1 but Not both.

Undecidability

A problem is undecidable if there is NO algorithm that takes as input an instance of the problem and determine whether the answer to that instance is YES or NO

Note:
→ if ∃ any such algorithm then problem is decidable.

TM ⟶ algorithm (church thesis)
problem is decidable ⟶ recursive language

→ { Halting problem }
for a given input string w and a Turing machine M, can we tell whether or not M halts on w

thm: The halting problem is undecidable.
i.e there is no TM that can accept any string w and any coded TM, M and always decide correctly whether M halts on w.

Halting problem

Ex.1

```
while (i != 1) {
    i = i - 2;
}
```

→ if segment started with i equal to some odd number (integer) then loop will terminate otherwise Not

Ex.2

```
while (i != 1) {
if (i % 2 == 0)
    i = i/2
else
    i = 3*i + 1;
}
```

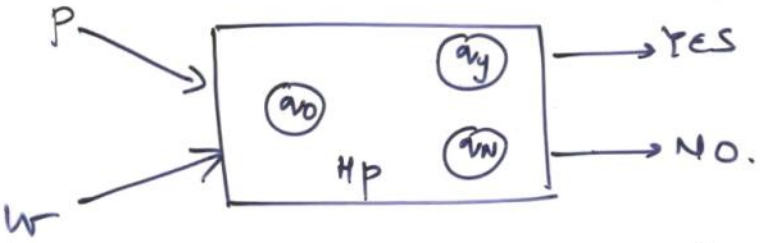
Try with i = 11

O/P 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Exercise:- Try it with i = 27, 21, 31, 96, 97, 871, 6171, 7703

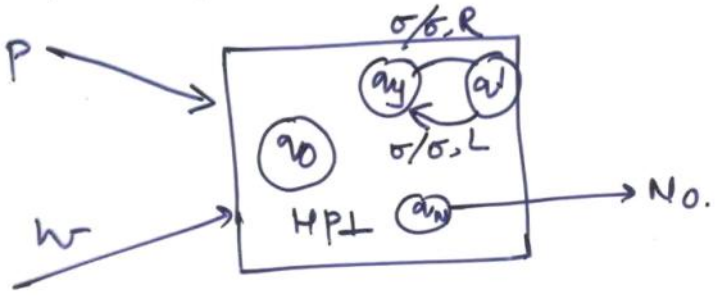
→ It has been found this program always terminates, even though it may take a long time to do so.

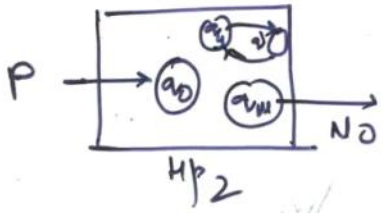
→ No. one is able to prove it terminates for all positive integers

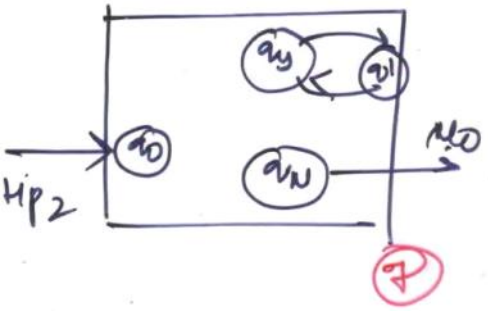


→ P is program which prints 'xyz' as o/p and halts.

→ Hp is a program that takes P and w and tell whether P with input w. print "xyz"

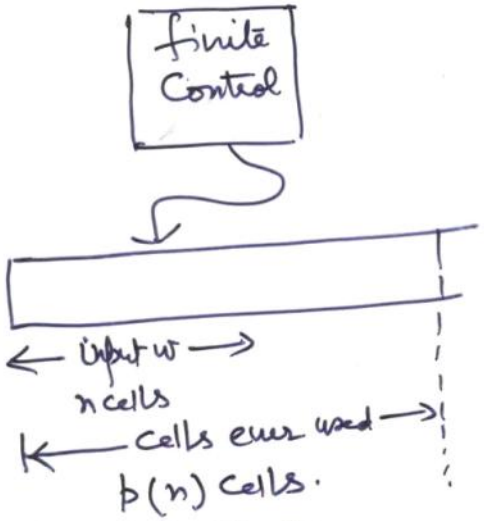






Space Complexity

→ A Turing M/C runs in space $f(n)$ if it scans a maximum of $f(n)$ tape cells for any input of length n



→ SAT is $O(n)$ space

input ϕ with $x_1 \cdots x_k$.

1. create array T, mapping variable indices. $1 \cdots k$ to true or false.
2. for every possible value of T:
   (i) Copy ϕ to work tape
   (ii) Replace the variables of ϕ with the Constants given by T
   (iii) Calculate formula
   (iv) if result == true accept ϕ
3. Reject ϕ.

T : $O(n)$ space
scratch copy of ϕ = $O(n)$ space

8

Example: DBL – Word :

1. If w is of odd length, reject
2. Compute $m = |w|/2$
3. for i from 1 to m do the following:
   (i) Record the current tape symbol in l
   (ii) Move right by m cells
   (iii) if the current tape cell $\neq$ l reject
   (iv) Move left by (m−1) cells
4. Accept.

How much work space does this require?

m : $O(\log n)$ space

l : $O(1)$ space

Total : $O(\log n)$ space

Polynomial space

$SPACE(f(n)) = \{L : L$ is decided by an $O(f(n))$ space deterministic Turing M/c $\}$

$NSPACE(f(n)) = \{L : L$ is decided by an $O(f(n))$ space non-deterministic TM $\}$

$PSPACE = \bigcup_k SPACE(n^k)$

$NPSPACE = \bigcup_k NSPACE(n^k)$.

Note:

(1) A Turing Machine M that uses space $f(n)$ runs in time $O(f(n))2^{O(f(n))}$

(2) A Turing M/C M that runs in time $f(n)$ uses $O(f(n))$

$P \subseteq PSPACE$

$NP \subseteq NPSPACE$

THM: if M is a polynomial-space bound TM (deterministic or Non-deterministic) and $p(n)$ is it's polynomial space bound then there is a constant c such that if M accepts its input w of length n, it does so within $c^{1+p(n)}$ moves.

proof outline:

t = No. of tape symbols

s = No. of states.

No. of different ID's when only $p(n)$ tape cells are used is at most $s\,p(n)\,t^{p(n)}$.

$\Rightarrow c = s + t^{1+p(n)} = t^{(1+p(n))} + (1+p(n))s\,t^{p(n)} + \cdots$

Savitch theorem:

for any f where $f(n) \geq n \; \forall n$

$$NSPACE(f(n)) \subseteq SPACE(f^2(n))$$

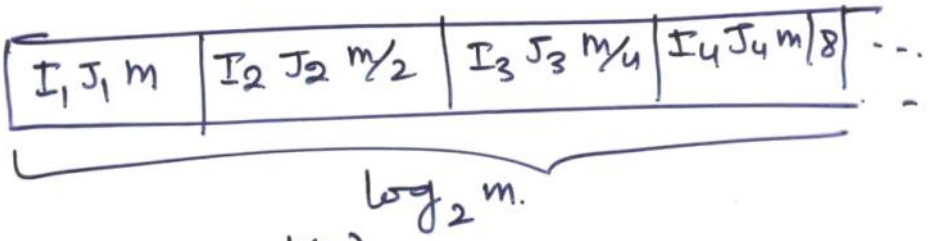
proof outline:

Consider the deterministic test for whether a NTM N can move from ID I to ID j in at most m moves.

→ A DTM D systematically tries all middle ID's K to check whether I can become K in $m/2$ moves and then K can become J in $m/2$ moves

```
Boolean function reach(I, J, M).
Begin
    if (m==1) then
      test if I==j or I can become J
        after one move
      return true if so, false if not.
    end;
    Else
      for each possible ID K DO
      if (reach(I, K, m/2) AND reach
                              (K, J, m/2)) then
        return true;
      return false
    End;
end;
```

Tape of a DTM simulating a NTM by recursive calls to reach.



$m \leq c^{p(n)}$

$O(p(n))$.

→ Each stack frame themselves take $O(p(n))$ space.

→ PSPACE Completeness

Defn: A problem B is $\underline{p}$-SPACE Complete if

(i) $B \in$ PSPACE and.

(ii) for every $A \in$ PSPACE, $A \leq_p B$.