# ICT Assignment-1

PRIYANSI SINGH
16UCS143

PAGE NO. :
DATE : / /

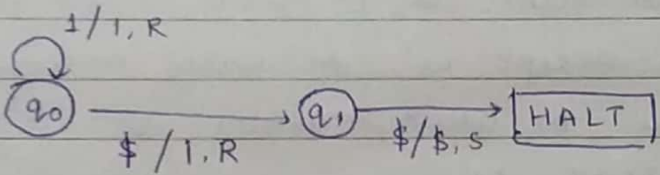
(1.) ~~T.M.~~ We can construct a T.M. M which computes function

$$s(n) = n+1 \quad , \quad \Sigma = \{1\}$$

in following way:

1) Keep on moving till we get 1st blank symbol '\$'.
2) Replace 1st '\$' with 1.
3) Halt the T.M.

State diagram:




S → stop

(2.) Given : $A_{TM} = \{ \langle M, w \rangle \mid$ T.M. M accepts $w \}$

We've to construct a TM which accepts $A_{TM}$.

We assume $A_{TM}$ is decidable. Let H be a decider for $A_{TM}$. On input w where w is string, H halts & accepts if M accepts w. Furthermore, H halts & rejects if M fails to accept w.

$$H(\langle M, w \rangle) = \begin{cases} \text{accept} & \text{if } M \text{ accepts } w \\ \text{reject} & \text{if } M \text{ rejects } w \end{cases}$$

(3) Given : L is Turing Recognizable & co-Turing Recognizable.
To prove : L is decidable

Proof : L is co-Turing Recognizable
⇒ $\bar{L}$ is Turing Recognizable

Scanned by CamScanner

PAGE NO.:
DATE: / /

Let $M_1$ be a decider of $L$ & $M_2$ be a decider for $\bar{L}$.

Let $M$ be a T.M. s.t. :

M (w) {

1. Run both $M_1$ & $M_2$ on input w in parallel
2. If $M_1$ accepts → accept
   If $M_2$ accepts → reject

}

Every string is either in $L$ or $\bar{L}$.

∴ $M_1$ or $M_2$ accept w. M halts whenever $M_1$ or $M_2$ accepts w, M always halts. so it's a decider

⇒ L is decidable language.

Hence Proved

(4) Given : $L = \{ \langle G \rangle \mid G$ is a connected & undirected graph $\}$

The following is a high-level description of a TM M that decides A.

M (<G>) { :

1. Select the 1st node of G & mark it
2. Repeat the following stage until no new nodes are marked.
3. For each node in G, mark if it's attached by an edge to a node that is already marked
4. Scan all the nodes of G to determine whether they all are marked
   If they are → accept
   else → reject

}

PAGE NO.:
DATE: / /

⑥ Given :

$EQ_{CFG} = \{ \langle G_1, G_2 \rangle \mid$ For the CFG $G_1$ & $G_2$, $L(G_1) = L(G_2) \}$

To prove :- $EQ_{CFG}$ is decidable.

Proof : We'll reduce $ALL_{CFG}$ to $EQ_{CFG}$ where

$ALL_{CFG} = \{ \langle G \rangle \mid G$ is a CFG & $L(G) = \Sigma^* \}$

We know that $ALL_{CFG}$ is undecidable.

CFG $G_0 = (V, \Sigma, R, S)$
$V = \{S\}$ & $S$ : string variable.
$l \in \Sigma$
$L(G_0) = \Sigma^*$

Let R be the decider for $EQ_{CFG}$ & S to decide $ALL_{CFG}$.

S (on input $\langle G \rangle$ where G is a CFG) {
1. Run R on input $\langle G, G_0 \rangle$ where $G_0$ is the CFG defined above with $L(G_0) = \Sigma^*$.
2. If R accepts → accept
R rejects → reject.
}

TM R determines if $L(G) = L(G_0)$ but $L(G_0) = \Sigma^*$,
it implies $L(G) = \Sigma^*$

∴ TM S decides $ALL_{CFG}$ but because it is undecidable,
this is contradiction
⇒ $EQ_{CFG}$ is decidable
Hence Proved.

PAGE NO.:
DATE: / /

(6) Given : $T = \{ <M> \mid TM\ M$ accepts $w^R$ whenever it accepts $w \}$

To prove : T is undecidable

Proof :

The basic idea is to reduce $A_{TM}$ to T, as we know that $A_{TM}$ is undecidable

Let A be decider of $A_{TM}$ s.t.

A (<M, w>) {

1. Check if <M, w> is valid encoding of a TM M & string w

If not → Reject

2. Construct following TM $M_2$ from M & w.

$M_2(w)$ :

1. If $x \in L(00^* 11^*)$ → accept
2. If $x \notin L(00^* 11^*)$ → Run M on w.

}

∴ T is undecidable

Hence Proved

(7) To prove : $A_{TM}$ can't be mapping reducible to $E_{TM}$.

Proof : let us prove this by contradiction.

Let $A_{TM}$ be mapping reducible to $E_{TM}$ via reduction f. It follows from the definition of mapping reducibility that $\overline{A_{TM}} \leq_m \overline{E_{TM}}$ via the same red^n func^n f.

But, $\overline{E_{TM}}$ is Turing Recognizable & $\overline{A_{TM}}$ is non-Turing Recognizable. This contradicts previous statement.

Hence, $A_{TM} \not\leq_m E_{TM}$ Hence Proved

Scanned by CamScanner

Priyansi Singh
16UCS143

PAGE NO.:
DATE: / /

(12) Given : $L_1$ and $L_2 \in NP$

To prove : a) $L_1 \cup L_2 \in NP$

b) $L_1 \cap L_2 \in NP$

<u>Proof</u> :

a) $L_1 \cap L_2 \in NP$

$L_1 \in NP$ & $L_2 \in NP$

$\Rightarrow \exists$ NDTM $M_1$ which decides $L_1$ in $O(n^k)$ time &

$\exists$ NDTM $M_2$ which decides $L_2$ in $O(n^l)$ time.

Let M' be a NDTM machine s.t. :

M'(w) :

1. Run $M_1$ on w. If $M_1$ rejected $\longrightarrow$ <u>Reject</u>
2. Else run $M_2$ on w. If $M_2$ rejected $\rightarrow$ <u>Reject</u>
3. Else <u>accept</u>

The longest branch in any computation tree on input w of length n is $O(n^{max(k,l)})$. So, M' is polynomial time non-deterministic decider of $L_1 \cap L_2$.

$\Rightarrow L_1 \cap L_2 \in NP$.

b) $L_1 \cup L_2 \in NP$

Taking same assumptions as in part a),

Let M' be a NDTM s.t. :

M'(w) :

1. Run $M_1$ on w. If $M_1$ accepted $\rightarrow$ <u>Accept</u>
2. Else Run $M_2$ on w. If $M_2$ accepted $\longrightarrow$ <u>Accept</u>
3. Else <u>Reject</u>

This will run in $O(n^{max(l,k)})$ time.

$\Rightarrow L_1 \cup L_2 \in NP$.

Hence <u>Proved</u>

(13) Given : $L \in$ NP-complete $\cap$ co-NP

To prove : co-NP = NP.

$L \in$ NP-Complete $\cap$ co-NP

$\Rightarrow$ $L \in$ NP-Complete and $L \in$ co-NP

$\Rightarrow$ (1) (2)

$L \in NP$ & $\forall L' \in NP, L' \leq_p L$

From (1) & (2),

NP $\subseteq$ CONP & CONP $\subseteq$ NP —— (3)

From (3), NP = co-NP

Hence Proved

(14) To Prove : $P \subseteq NP \cap$ co-NP

Proof : let $L \in P \Rightarrow L \in NP$ —— (1) $(\because P \subseteq NP)$

As we know P is closed under complement

$\Rightarrow \bar{L} \in P \Rightarrow \bar{L} \in NP$

$\therefore \bar{L} \in NP \Rightarrow L \in$ co-NP —— (2)

$\Rightarrow L \in NP$ and $L \in$ co-NP

$\Rightarrow$ $\underline{L \in NP \cap \text{co-NP}}$

Hence Proved.

PAGE NO. :
DATE : / /

(15) To prove : $P = NP \Rightarrow NP = co\text{-}NP$

Proof : let $P = NP$, then

1. For every $L \in NP$, we have $L \in P$, and since the languages in P are closed under complement

$\Rightarrow \bar{L} \in P$

$\Rightarrow L \in coNP$. —— (1)

2. for every $L \in coNP$, we have $\bar{L} \in P$

$\Rightarrow L \in P$

$\Rightarrow L \in NP$ —— (2)

From (1) & (2),

$NP = co\text{-}NP$

Hence Proved.

(16) Given :

HALT = { <M,w> | T.M. M halts on input w }

To prove : HALT is NP-hard

Proof : HALT can be NP-hard if :

$\forall B \in NP$, $B \leq_p$ HALT

- We will try to reduce SAT problem into HALT problem in polynomial time.
- For this, let us construct an algorithm A for which input is formula x.

suppose x has n variables

PAGE NO.:
DATE: / /

- Algo A tries out all possible $2^n$ truth assignment & verifies if x is satisfiable.

If x : satisfiable → A halts

else → A enters infinite loop.

Hence A halts on i/p iff x is satisfiable

⇒ HALT ∈ NP-Hard

∴ Halting problem is undecidable ⇒ HALT ∉ NP

Hence, HALT is NP-Hard but not NP.

(17) Given:

3COLOR = { G = (V,E) | A coloring of vertices of G by 3 colors then each adjacent vertices get different color.}

To prove: 3COLOR is NP-complete.

~~Proof~~:

3COLOR is NP-complete iff:

1) 3COLOR ∈ NP

2) ∀ A ∈ NP, $A \leq_p$ 3COLOR

We'll try to reduce 3-SAT problem into 3-COLOR problem in order to prove that 3-COLOR is NP-Complete.

1) Since we can verify coloring within polynomial time, hence,

3COLOR ∈ NP.

Scanned by CamScanner

Priyansi Singh 16UCS143

PAGE NO.:
DATE: / /

2) 3-SAT $\leq_p$ 3COLOR

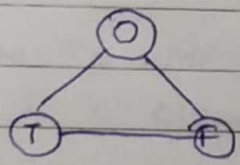
Idea : Use a collection of gadgets to solve the problem.

↳ Build a gadget to assign ~~o~~ 2 of the colors with the labels "true" and "false".

↳ Build a gadget to force each variable to be either true or false

↳ Build a series of gadgets to force those variables assignments to satisfy each clause.

Gadget 1: Assigning Meaning




These 3 nodes must have different colors.

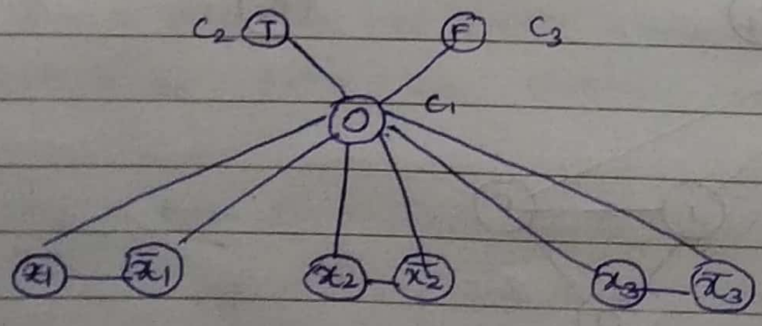
color assigned to T : 'True'
color assigned to F : 'False'
no ~~color ass~~ special meaning ~~to~~ associated with O.

Gadget 2: Forcing a choice

Let 3-SAT statement be:

~~$\phi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{y}_2 \vee z)$~~

$\phi = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee \bar{x}_3)$




all literals have 2 choices : $C_2$ or $C_3$.

if $x_1 : C_2$ then $\bar{x}_1 : C_3$

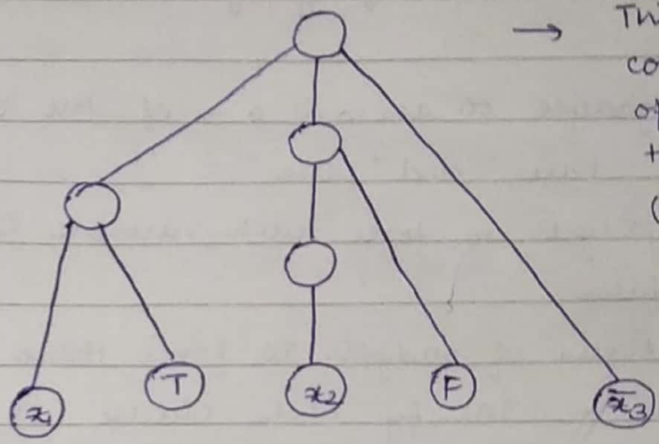
similarly with other literals.

Scanned by CamScanner

PAGE NO.:
DATE: / /

## Gadget 3 : clause satisfiability

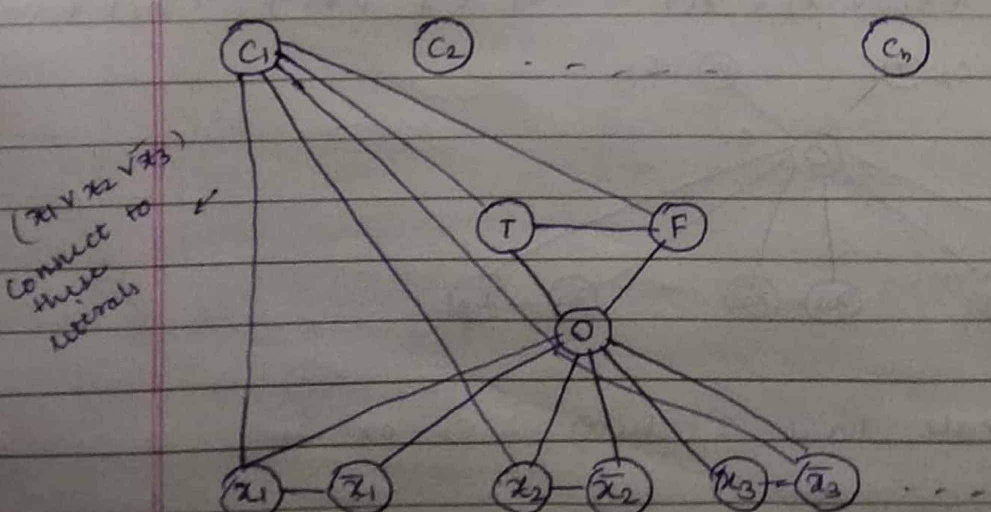
$(x_1 \vee x_2 \vee \bar{x}_3)$




→ This node is colorable iff one of the inputs is the same color as T
(One of the literals is true)

**Overall :**

1. Construct 1st gadget so we have a consistent definition of true & false
2. For each variable v :
   i) Construct nodes v & $\bar{v}$
   ii.) Add an edge between v & $\bar{v}$
   iii) Add an edge b/w v & O & b/w $\bar{v}$ & O.
3. For each clause c :
   Construct earlier gadget from c by adding in extra nodes & edges.




Thus, 3 COLOR is NP-Complete

PAGE NO.:
DATE: / /

(18) Given :
MAX-CUT = { <G, K> | G has cut of size k or more }

To prove : MAX-CUT is NP-complete

Proof 1.) MAX-CUT $\in$ NP because a non-deterministic algorithm can guess the cut & check that it has size at least k in polynomial time.

2.) We need to show :

$$3\,SAT \leq_p \text{MAX-CUT}$$

Let $\phi$ be an instance of 3-SAT with 'v' variables & 'c' clauses. We build a graph G with 6cv nodes

- Each variable $x_i$ corresponds to 6c nodes ; 3c labelled with $x_i$, & 3c labelled with $\bar{x}_i$.
- Connect each $x_i$ with each $\bar{x}_i$ node for a total of $9c^2$ edges.
- For each clause in $\phi$, select 3 nodes labeled by the literal in that clause & connect them by edges. Avoid selecting the same node more than once.
- Let K be $9c^2v + 2c$. Output <G, K>

→ Take a true assignment of $\phi$. It yields a cut of size K by placing all nodes corresponding to true variables on 1 side & all other nodes at the other side. Then, the cut contains all $9c^2v$ edges b/w literals & their negations & 2 edges for each clause, because it contains atleast 1 true & 1 false

PAGE NO.:
DATE: / /

literal, yielding a total of $9c^2 v + 2c = k$ edges

→ For the converse, take a cut of size k. The clause edges can contribute atmost 2c to the cut. G has only $9c^2 v$ other edges, so if G has a cut of size k, all of those other edges appear in it & each clause contributes its max$^n$.

Hence, all nodes labeled by same literal appear on same side of cut.

By selecting either side of cut & assigning its literals true, we get true assignment.

Hence, 3SAT $\leq_p$ MAX-CUT.

19. To prove: 2SAT $\in$ NL

Proof: 2-SAT can be solved by solving UNREACHABILITY on the graph.

Suppose a 2-SAT formula F is given. Construct following directed graph G:

1. vertices in G: variables in F & their negations
2. edge (a,b) exists iff one of the clause $(\bar{a} \vee b)$ exists in F.

Hence, formula is satisfiable iff there exists no path between a & b.

Hence, 2SAT $\in$ NL

since UNREACHABILITY $\in$ NL (Already known)

⇒ 2SAT $\in$ NL- complete

Scanned by CamScanner

Priyansi Singh 16UCS143

PAGE NO.:
DATE: / /

(20) Given $L = \{ \langle G \rangle \mid G$ is strongly connected digraph $\}$

To prove : $L$ is NL-complete

1.) First, we show that $L \in NL$

Consider the following TM which decides $\bar{L}$.

On input $\langle G \rangle$:

M ($\langle G \rangle$) {

1. Non-deterministically select 2 nodes a & b.
2. Run PATH (a, b)

If it rejects → accept

Otherwise → reject

}

Since storing nodes a & b take $O(\log n)$ space,

$\bar{L} \in NL$. This means $L \in$ CO-NL.

Since NL = CO-NL

$\Rightarrow L \in NL$

2.) $\forall L' \in NL$, $L' \leq_L L$

We do this by reducing PATH to L.

Consider following TM M:

M ($\langle G, s, t \rangle$) {

1. Copy all of G onto O/p tape
2. Additionally for each node i in G
3. Output an edge from i to s
4. Output an edge from t to i

}

If $\exists$ a path from s to t, the graph constructed is strongly connected.

else → not strongly connected.

PAGE NO.:
DATE: / /

Since we only have to keep track of $i^{th}$ node, it is logarithmic time reducible.

(11.) Since there is a root at $x = x_0$

$\Rightarrow$ $c_1 x_0^n + c_2 x_0^{n-1} + \dots + c_n x_0 + c_{n+1} = 0$

Rearranging the terms,

$$c_1 x_0^n = -(c_2 x_0^{n-1} + \dots + c_n x_0 + c_{n+1})$$

Taking absolute value on both sides, & applying triangular inequality, we get

$$|c_1 x_0^n| \leq |c_2 x_0^{n-1}| + \dots + |c_n x_0| + |c_{n+1}|$$

The inequality still holds if we substitute $c_{max}$ for all coeffecients

$$|c_1 x_0^n| \leq |c_{max}| (1 + |x_0| + \dots + |x_0|^{n-1})$$

Case I : $x_0 > 1$ :

$$|c_1 x_0^n| \leq |c_{max}| (1 + |x_0| + \dots + |x_0^{n-1}|)$$

Inequality still holds if we substitute $n|x_0^{n-1}|$ for $(1 + |x_0| + \dots + |x_0^{n-1}|)$

$\Rightarrow$ $|c_1 x_0^n| \leq |c_{max}| \, n \, |x_0^{n-1}|$

$\Rightarrow$ $|x_0| \leq n \dfrac{|c_{max}|}{|c_1|}$

$\Rightarrow$ $|x_0| < (n+1) \dfrac{|c_{max}|}{|c_1|}$