Satisfiability (SAT) → Boolean satisfiability or SAT is the problem of determining if a boolean formula is satisfiable or unsatisfiable.

→ satisfiable : If the boolean variable can be assigned values such that the formula turn out to be TRUE then we say that formula is satisfiable. If it does not then it is ~~unsat~~ unsatisfiable.

CNF (Conjuctive Normal Form) :

CNF is a conjuction (AND) of clauses where every clause is disjunction (OR)

2-SAT ⇒ Given CNF with each clause have only two terms.

→ These can be solved in polynomial times.

Ex → $F = (x_1 \cup x_2) \wedge (x_2 \vee \bar{x_1}) \wedge (\bar{x_1} \vee \bar{x_2})$

3-SAT ⇒ Given CNF with each clauses have only three terms.

Ex $F = (x_1 \cup x_2 \cup x_3) \wedge (\bar{x_1} \cup x_2 \vee x_3) \wedge (x_1 \cup \bar{x_2} \cup x_3)$

Decision problem: A problem with a yes or no answer.

P → P is a complexity class that represents the set of all decision problems that can be solved in polynomial time.

→ That is given an instance of problem, the answer yes or NO can be decided in polynomial time.

Example Given a connected graph G, can its vertices be coloured using two colours so that every edge is monochramatic.

Shortest Path Problem

NP :→ NP is a complexity class that represents the set of all decision problems for which the instances where the answer is "yes" have proofs that can be verified in polynomial time.

This means that if someone gives us an instance of the problem and a certificate

(sometime called witness) to the answer being yes, we can check it in polynomial time.

Ex Integer factorization is in NP. This is the problem that given integers n and m, is there an integer f with $1 < f \leq m$ such that f divides n (f is a small factor of n)?

→ This is a decision problem because the answer is yes or no. If some hacker is an instance of the problem (so they had us integers n and m) and an integer f with $1 < f < m$ and claim that f is a factor of n (the certificate), we can check the answer in polynomial time by performing the division n/f.

NP-complete: NP complete is a complexity class which represents the set of all problems x in NP for which it is possible to reduce any other NP problem y to x in polynomial time.

→ Intuitively this means that we can solve Y quickly if we know how to solve X quickly. Precisely Y is reducible to X, if there is a polynomial time algorithm f to transform instances y of Y to instances $x = f(y)$

of x in polynomial time, with the property tha
answer to y is yes, if and only if the answ
to f(y) is yes.

→ 3-SAT problem is first known NP complete prob
→ It can be shown that every NP problem can be r
to 3-SAT. The proof of this is technical and req
use of technical definition of NP (based on non
deterministic Turing machine). This is known a
Cook's theorem.

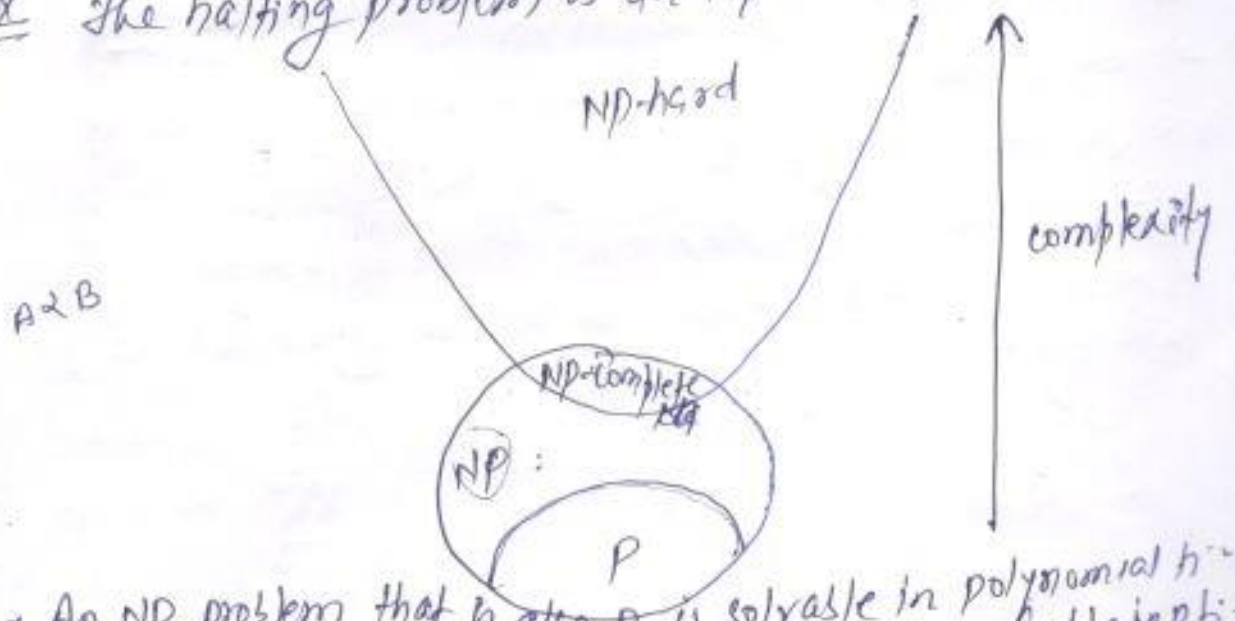
→ So, if a deterministic polynomial time algor
can be found to solve one of them, every NP p
can be solved in polynomial time.

NP-hard Intuitively, these are the problems
that are at least as hard as the NP comp
problem. NP-hard problems do not have to be in
and they do not have to be decision problem

⇒ The precise definition of NP-hard is th
a problem x is NP-hard, if there is an N
problem Y, such that Y is reducible to x in
polynomial time.

→ But since any NP-complete problem can be reduced to another NP-complete problem in polynomial time, all NP-complete problems can be reduced to NP-hard problem in polynomial time. Then if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP-problems in polynomial time.

ex The halting problem is an NP hard problem.



* An NP-problem that is also P is solvable in polynomial ti...
** An NP-hard problem that is also NP-complete is verifiable in pti...
*** NP-complete problems (all of which form a subset of NP-hard) might be

problem Type	Verifiable in P time	Solvable in P time
P	YES	YES
NP	YES	YES or NO*
NP-complete	YES	UNKNOWN
NP-hard	YES or NO**	UNknown***

Solvability decreases

⇒ Reducibility for any problem (NP-hard or any other) means the possibility to convert problem A into other problem B. If we know the complexity of problem B then the complexity of problem A is at least the same as the complexity of problem A.