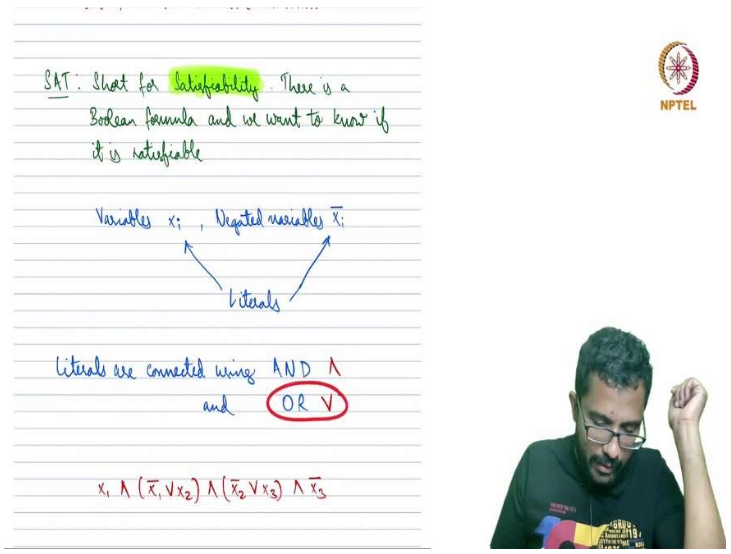
Theory of Computation
Professor Subrahmanyam Kalyanasundaram
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad
Non-Deterministic Polynomial Time - Part 2

(Refer Slide Time: 0:15)



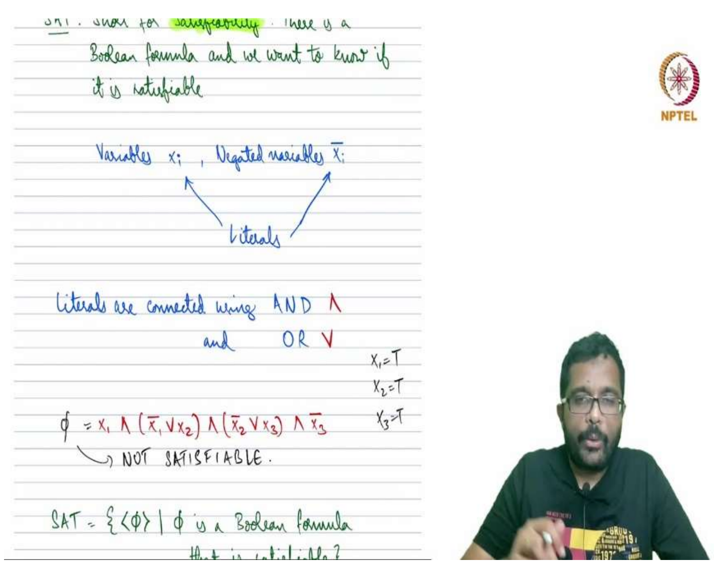
SAT: Short for Satisfiability. There is a Boolean formula and we want to know if it is satisfiable

Variables x_i, Negated variables $\overline{x_i}$

Literals

Literals are connected using AND $\wedge$ and OR $\vee$

$x_1 \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge \overline{x_3}$



Boolean formula and we want to know if it is satisfiable

Variables x_i, Negated variables $\overline{x_i}$

Literals

Literals are connected using AND $\wedge$ and OR $\vee$

$x_1 = T$
$x_2 = T$
$x_3 = T$

$\phi = x_1 \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge \overline{x_3}$
NOT SATISFIABLE.

SAT = $\{ \langle \phi \rangle \mid \phi$ is a Boolean formula

So, now we come to it another problem in NP called SAT, SAT is short for satisfiability. So, basically it is a Boolean formula and we are asking whether it is satisfiable. So, what is the Boolean formula contain? It contains variables $x1, x2, x3$ there are negated variables which is $\overline{x1},\ \overline{x2}$.

So, both of these variables and negative variables are sometimes together called literals, literals are both $x1, x2, x3$ and so on and $\overline{x1},\ \overline{x2}$, et cetera. So, both variables as well as the negated variables, and they are connected using 2 logical things logical connectors. So, one of them is AND which is depicted as this, depicted like this, like an inverted V or an A without a bar.

And the other one is OR so, OR is depicted like a V, it is exactly like V. So, one example is this, this Boolean formula given here, so, let us say this is φ. So, is there a way to assign values to $x1, x2, x3$, so, there are 3 variables, $x1, x2, x3$ such that this whole formula φ evaluates to true. So, I want to make this formula true, is there a way to assign $x1, x2, x3$ values so that the whole formula evaluates to true. Let us see.

So, it is an AND of 4 things.

$$\varphi = x1 \wedge (\overline{x1} \vee x2) \wedge (\overline{x2} \vee x3) \wedge \overline{x3}$$

So, there is an AND of $x1$, which means $x1$ should be true, maybe I will write it here. So, $x1$ should be true, because if $x1$ is false, the whole thing is false, because there is an AND of 4 things. So, $x1$ should be true, but then if $x1$ is true, then $\overline{x1}$ is false.

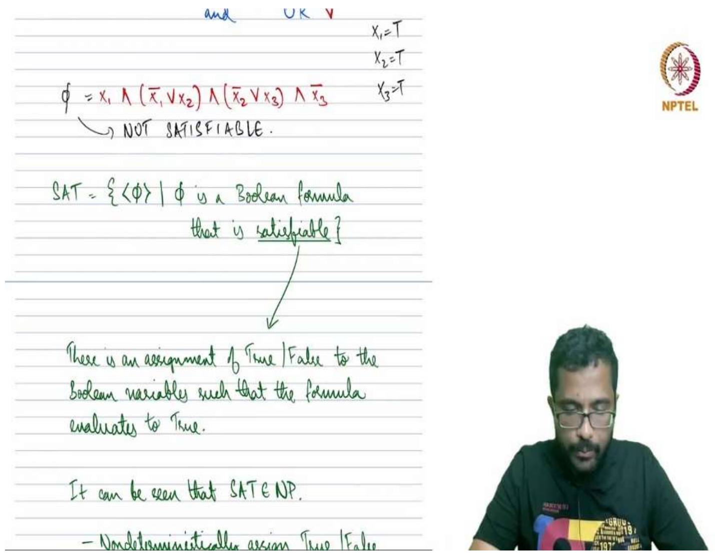
So, now, I want this part to be true, $x1$ is true means $\overline{x1}$ is false. So, which means $x2$ has to be necessarily true, otherwise, this part is going to be false. So, $x2$ also has to be true. Now, we know $x2$ is true, which means $\overline{x2}$ is false. So, now, I want this clause $(\overline{x2} \vee x3)$ to be true. Now, it necessarily means that $x2$ is true. So, the only way φ is satisfied is if $x1$ is true.

And now, I want to satisfy this clause, the second clause, so, the only way φ can be satisfied is the second clause is true, but we know x1 has to be true. So, x2 also has to be true. And now, since x2 has to be true, and the third clause has to be true, x3 also has to be true. So, from the first 3 clauses, what we gather is that x1, x2, x3 all has to be true for the φ to be true, but let us see.

So, the final clause is just $\overline{x3}$, but then we already saw that x3 has to be true. So, this has to be false, which means whatever we try, so, we got the x1, x2, x3 has to be true, but then x3 has to be false, but an x3 has to be assign either true or false, it cannot have both assignments, which means this is not satisfiable. Because we cannot get x1, x2, x3 values in a way that this evaluates to true.

Now, suppose this was not there, suppose the last clause was not there, $\overline{x3}$ was not there, then of course, this would have been, suppose then this was not there, then this clause φ would have been satisfiable with x1 true, x2 true, and x3 true. So, now, let me just bring it back. So, this clause is this formula is not satisfiable.

(Refer Slide Time: 4:58)



and OR $\vee$

$x_1 = T$
$x_2 = T$
$x_3 = T$

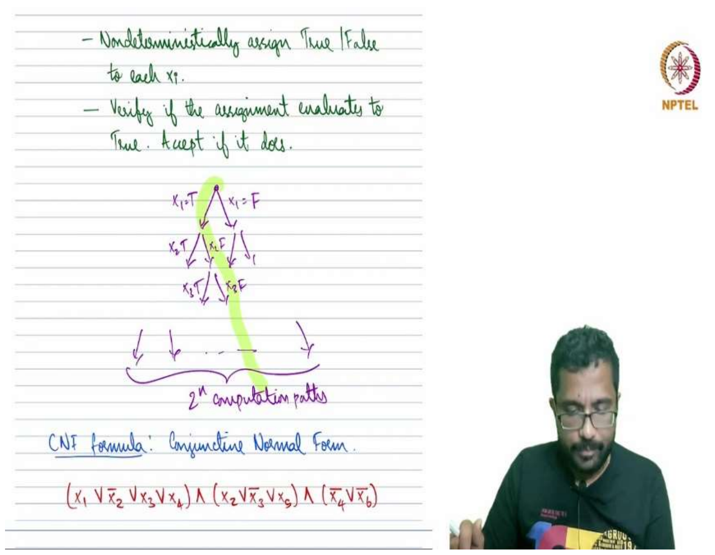
$\phi = x_1 \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_2 \vee x_3) \wedge \bar{x}_3$

NOT SATISFIABLE.

SAT = $\{\langle \phi \rangle \mid \phi$ is a Boolean formula that is satisfiable$\}$

There is an assignment of True/False to the Boolean variables such that the formula evaluates to True.

It can be seen that SAT $\in$ NP.

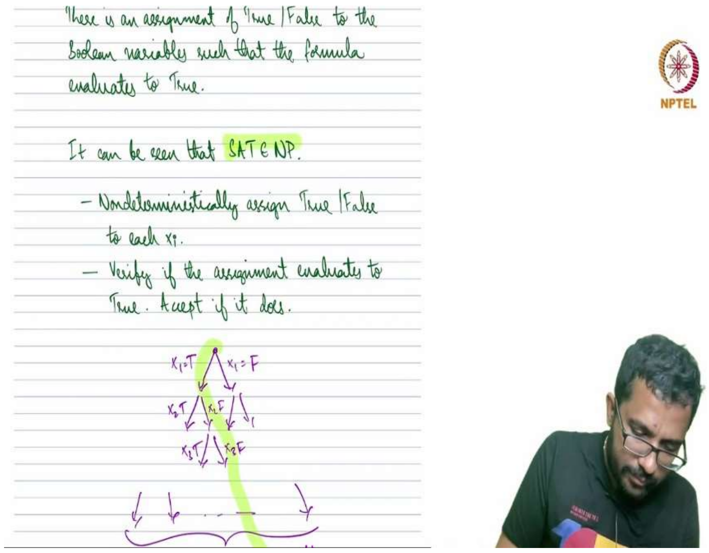


– Nondeterministically assign True/False to each x_i.

– Verify if the assignment evaluates to True. Accept if it does.

CNF formula: Conjunctive Normal Form.

$(x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee x_5) \wedge (\bar{x}_4 \vee \bar{x}_6)$


There is an assignment of True/False to the Boolean variables such that the formula evaluates to True.

It can be seen that SAT ∈ NP.

– Nondeterministically assign True/False to each x_i.

– Verify if the assignment evaluates to True. Accept if it does.

So, the question of satisfiability is, is a given Boolean formula satisfiable? Satisfiable means is there an assignment of true or false to the Boolean variables such that the formula evaluates to true? It can again be checked that this question is an NP. Why? Because we can do something similar to what we did earlier. We can non-deterministically, assign x1 to be true. And x1 to be false. So, for our each variable to be non-deterministically, pick, true or false.

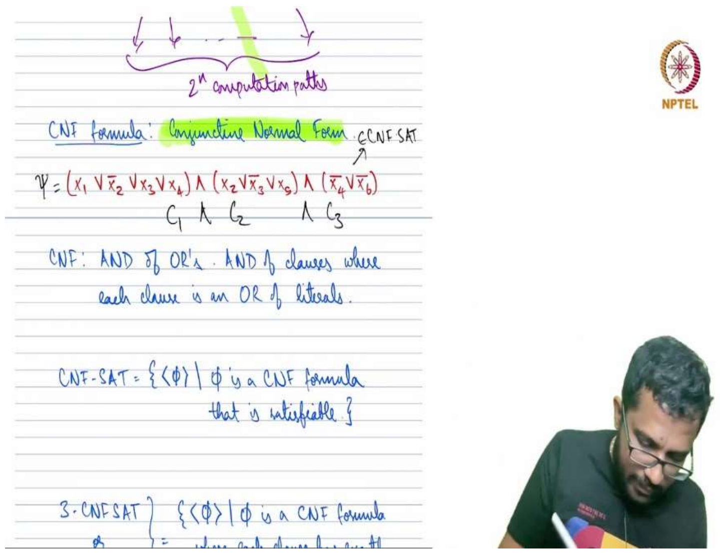
And now, within the 2 options, we pick x2 true and x2 false. And similarly, here, and so on, maybe x3 true, x3 false. And finally, we get something like, we get something like 2^n possible options, sorry, let me just say computation paths. And for each path, we check whether so but then each computation path, let us say, let us take some path. So, let us take some path.

Let us take some path, something like this, each computation path corresponds to a certain assignment. Now, we just have to verify that assignment path is it, does it satisfy the formula or not? If it satisfies, we accept if it does not satisfy, we reject. So, if there is a satisfying assignment, it is one of these 2^n possible paths. And then that path leads to acceptance.

If the formula is not satisfiable, meaning whatever you try, it is not going to accept that all paths will reject. So, this is obviously an NP algorithm. And this is an non-deterministic algorithm, it is a correct algorithm and the running time is polynomial because it takes n steps to non-deterministically assign yes, or no or true or false to each of these n variables, x1 to xn.

And then it takes again, linear time to verify, so this may, linear time, or maybe at most polynomial time to verify, depending on how the formula is structured, that whether this assignment satisfies the formula or not. So, I think linear time is enough. So, clearly this shows that SAT is in NP.

(Refer Slide Time: 8:07)



2^n computation paths

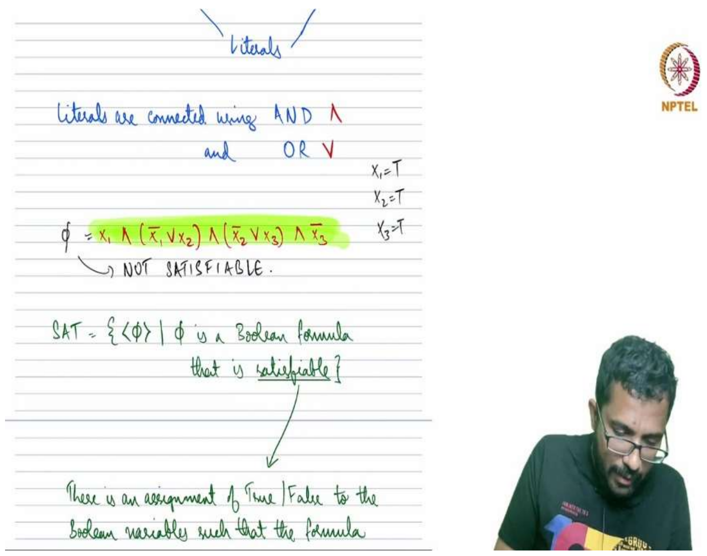
CNF formula: Conjunctive Normal Form $\in$ CNF SAT

$$\psi = (x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee x_5) \wedge (\bar{x}_4 \vee \bar{x}_6)$$
$$C_1 \wedge C_2 \wedge C_3$$

CNF: AND of OR's. AND of clauses where each clause is an OR of literals.

CNF-SAT = { ⟨ϕ⟩ | ϕ is a CNF formula that is satisfiable }

3-CNFSAT = { ⟨ϕ⟩ | ϕ is a CNF formula



Literals

Literals are connected using AND $\wedge$ and OR $\vee$

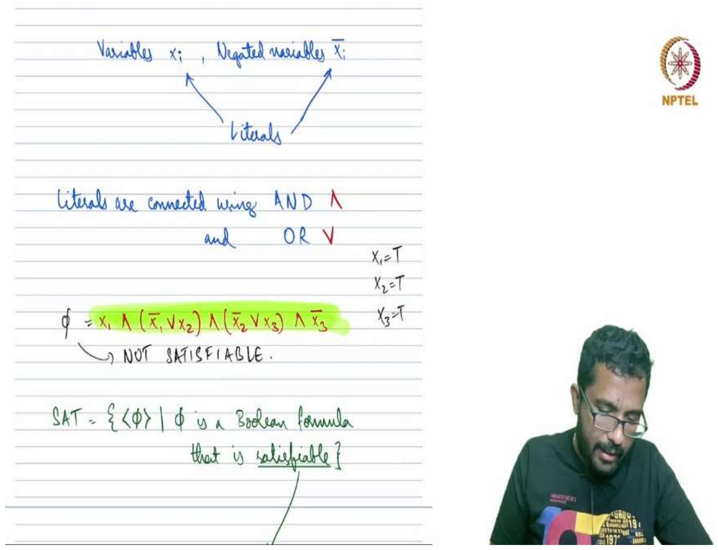
$x_1 = T$
$x_2 = T$
$x_3 = T$

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

NOT SATISFIABLE.

SAT = { ⟨ϕ⟩ | ϕ is a Boolean formula that is satisfiable }

There is an assignment of True/False to the Boolean variables such that the formula



Variables x_i , Negated variables $\bar{x}_i$

Literals

Literals are connected using AND $\wedge$
and OR $\vee$

$x_1 = T$
$x_2 = T$
$x_3 = T$

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

NOT SATISFIABLE.

SAT = $\{ \langle \phi \rangle \mid \phi$ is a Boolean formula that is satisfiable $\}$

Couple of further things that I want to just quickly briefly tell. So, the formula that we saw here, this formula that we saw here, it is in this form called conjunctive normal form, because I have an and of things. So, I have x1 AND x1 complement, OR x2 AND x2 complement, OR x3 AND x3 complement, so I have an AND of things. So I have an AND of clauses, where each clause is an OR of literals, this is called conjunctive normal form.

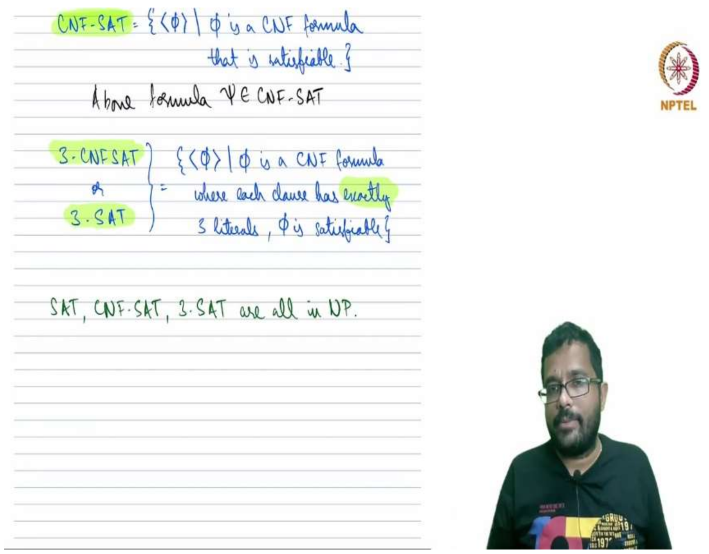
So, here also I have another formula, which is conjunctive normal form, where I have 3 clauses first clauses x1 OR x2 complement, OR x3 OR x 4. Second clause is x2 OR x3 complement, OR x5, third clause is x4 complement, or x6. So, it is like, the formula can be viewed as sorry, C1 AND C2 AND C3, it is an AND of 3 clauses, where each clause is an OR of literals.

So, for instance, C1 is simply x1 OR x2 complement OR x3 OR x4, C3 is simply x4 complement OR x6. So, it is an AND of clauses, where each clause has an OR of literals. That is called conjunctive normal form, also sometimes it is abbreviated as CNF, conjunctive normal form. So, I can define a language called CNF SAT. CNF SAT simply is asking if a given formula is it satisfiable? So, just like SAT, SAT is given a formula is satisfiable. CNF is also asking the same question, but the given formula is in CNF form.

So, this is in CNF form. So, there is a certain structure to the formula that is given then we have to determine whether it is satisfiable. So, this is a conjunctive normal form and you can check that this is actually satisfiable. This is in CNF SAT. So, you can set x1 to true which satisfies clause 1, x2 to true which satisfies clause 2, and let us say x4 to false which satisfies clause 3. So, that is enough, that is enough. So, this is satisfiable. So, let us say this is called ψ.

$$\psi = (x1 \lor \overline{x2} \lor x3 \lor x4) \land (x2 \lor \overline{x3} \lor x5) \land (\overline{x4} \lor \overline{x6})$$

(Refer Slide Time: 10:58)

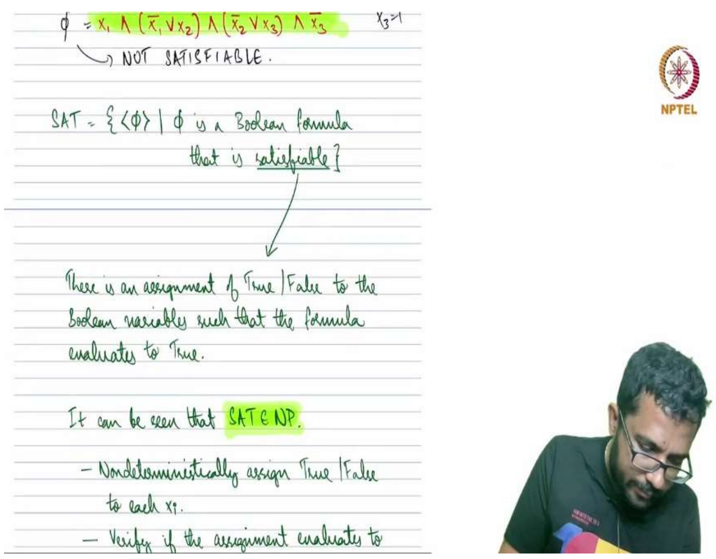


CNF-SAT = $\{\langle\phi\rangle \mid \phi$ is a CNF formula that is satisfiable $\}$

Above formula $\psi \in$ CNF-SAT

3-CNFSAT or 3-SAT $\Big\} = \{\langle\phi\rangle \mid \phi$ is a CNF formula where each clause has exactly 3 literals, ϕ is satisfiable $\}$

SAT, CNF-SAT, 3-SAT are all in NP.



$\phi = x_1 \land (\bar{x}_1 \lor x_2) \land (\bar{x}_2 \lor x_3) \land \bar{x}_3$ $\quad x_3 = 1$

$\hookrightarrow$ NOT SATISFIABLE.

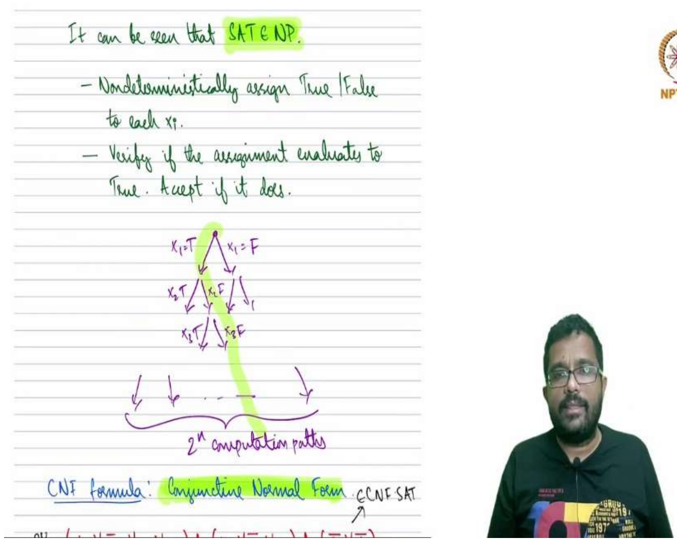
SAT = $\{\langle\phi\rangle \mid \phi$ is a Boolean formula that is satisfiable $\}$

$\downarrow$

There is an assignment of True/False to the Boolean variables such that the formula evaluates to True.

It can be seen that SAT $\in$ NP.

- Nondeterministically assign True/False to each x_i.
- Verify if the assignment evaluates to



It can be seen that SAT ∈ NP.

— Nondeterministically assign True/False to each x_i.

— Verify if the assignment evaluates to True. Accept if it does.

CNF formula: Conjunctive Normal Form ← CNF SAT

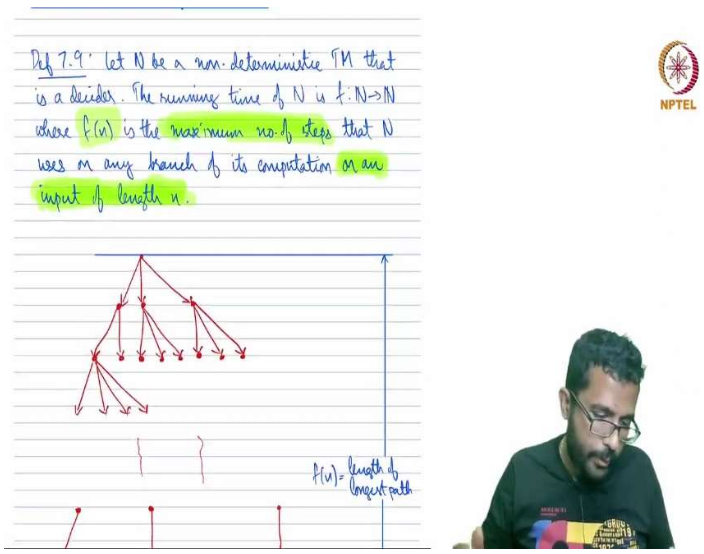
So, above ψ, above formulas ψ is in CNF SAT and further I want to make one more definition. So, the CNF SAT, earlier we defined SAT. Here, I want to define one more thing called 3 CNF SAT or 3 SAT. This is actually a special case of CNF SAT. So, CNF SAT means given a formula is satisfiable and the formula is in CNF form. 3 CNF SAT is saying that it is CNF form, but with additional condition that each clause has exactly 3 literals.

So, here in the formula ψ over here, if you see, clause 1 has 4 variables, clause 4 literals, clause 2 has 3 and cross 3 has 2. But in 3 CNF SAT each clause must have exactly 3 literals, exactly 3 literals that is and then we are asking whether it is satisfiable. So, this is even further restriction on the structure of the given formula.

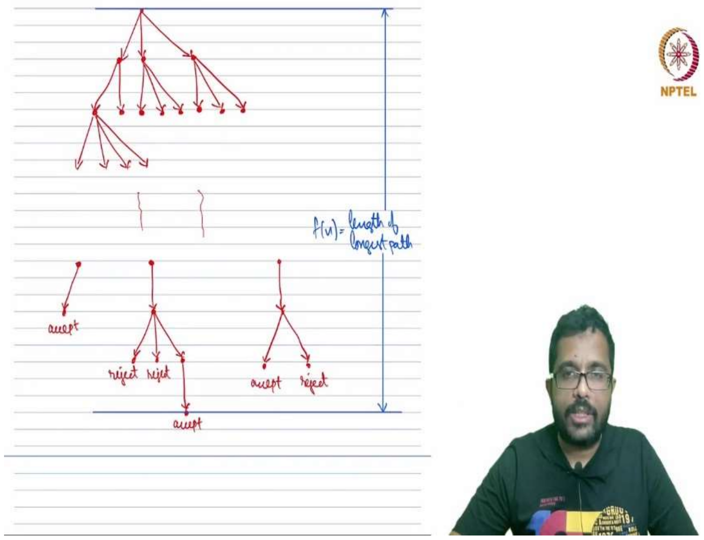
So, perhaps the hope is that maybe if the formula has certain specific restriction or a specific format, then maybe it is easier to figure out whether it is satisfiable or not. Anyway, the approach that I mentioned for SAT being in NP, I can again use the same approach for CNF SAT and 3 CNF SAT.

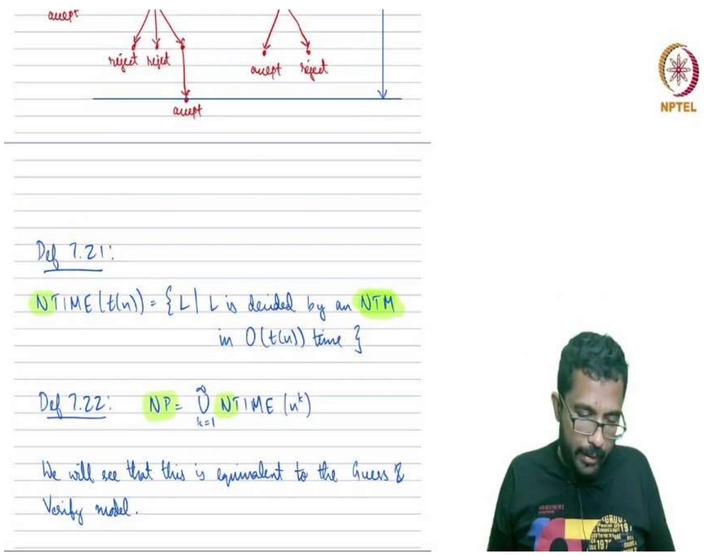
So, I can guess a certain assignment, what we are doing is we are guessing a certain assignment and then checking whether this assignment satisfies the formula. If the guessed assignment satisfies, we accept, if the guessed assignment does not satisfy the reject. If the formula is satisfiable one of the guesses will lead to accept, if the formula is not satisfiable all the guesses will reject.

(Refer Slide Time: 13:19)



Def 7.9: Let N be a non-deterministic TM that is a decider. The running time of N is $f: \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum no. of steps that N uses on any branch of its computation on an input of length n.



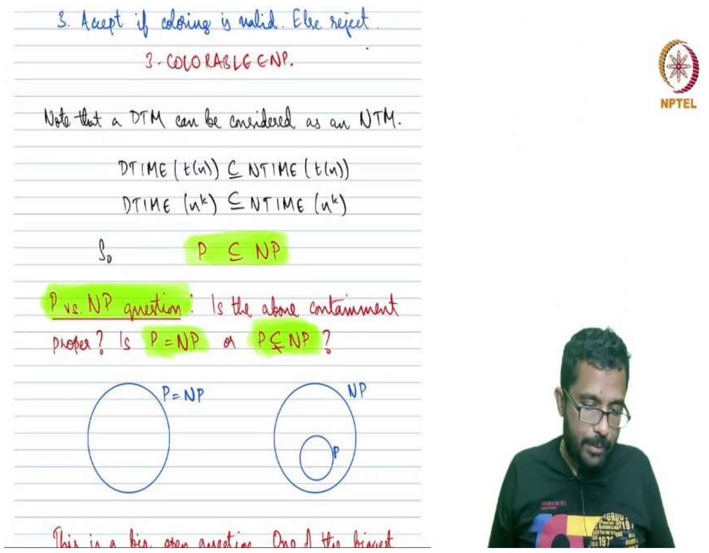


Def 7.21:

$\mathrm{NTIME}(t(n)) = \{ L \mid L$ is decided by an NTM in $O(t(n))$ time $\}$

Def 7.22: $\mathrm{NP} = \bigcup_{k=1}^{\infty} \mathrm{NTIME}(n^k)$

We will see that this is equivalent to the Guess & Verify model.


3. Accept if coloring is valid. Else reject.

3-COLORABLE $\in$ NP.

Note that a DTM can be considered as an NTM.

$$DTIME(t(n)) \subseteq NTIME(t(n))$$
$$DTIME(n^k) \subseteq NTIME(n^k)$$

So $P \subseteq NP$

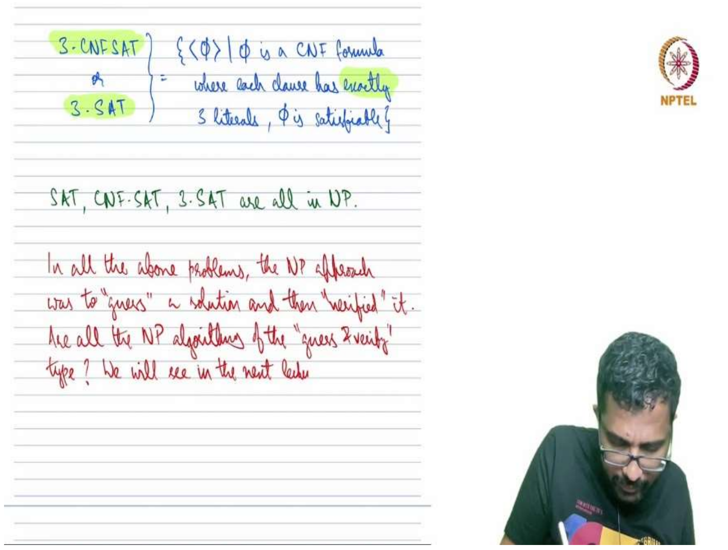
P vs. NP question: Is the above containment proper? Is $P = NP$ or $P \subsetneq NP$?

This is a big open question. One of the biggest

So, that concludes most of what I want to say. So, we said how to measure time complexity for an Turing machine, it is the length of the longest path and then we take over all the possible inputs of the same length. Then we define $NTIME(t(n))$, then we define NP then we also talked about the P versus NP question.

So, P is a class of languages that can be decided by a deterministic Turing Machine in polynomial time and NP is a class of languages that can be decided by a non-deterministic Turing machines in polynomial time. The question is, P is clearly a subset of NP is it a proper subset? And we discussed a bit about that. And then we saw some examples, we saw subset sum, we saw 3 colorable. And we saw SAT.

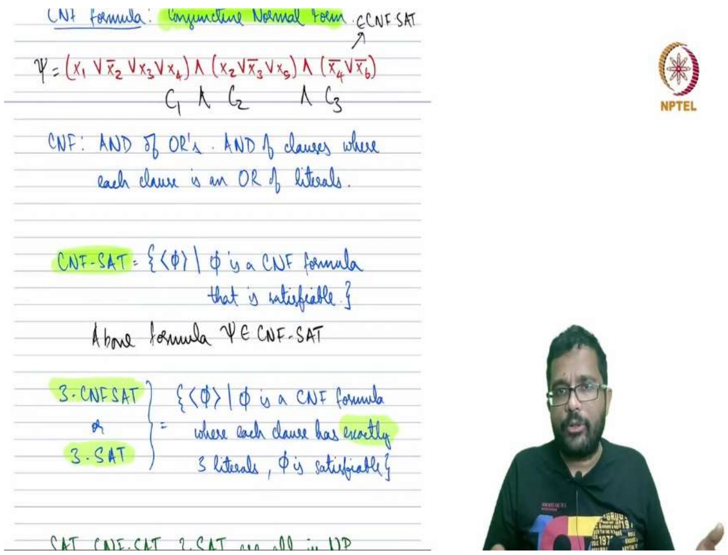
(Refer Slide Time: 14:10)



3-CNFSAT or 3-SAT $= \{ \langle \phi \rangle \mid \phi$ is a CNF formula where each clause has exactly 3 literals, ϕ is satisfiable $\}$

SAT, CNF-SAT, 3-SAT are all in NP.

In all the above problems, the NP approach was to "guess" a solution and then "verified" it. Are all the NP algorithms of the "guess & verify" type? We will see in the next lecture



CNF formula: Conjunctive Normal Form $\in$ CNF SAT

$$\Psi = (x_1 \vee \bar{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \bar{x}_3 \vee x_5) \wedge (\bar{x}_4 \vee \bar{x}_6)$$
$$C_1 \wedge C_2 \wedge C_3$$

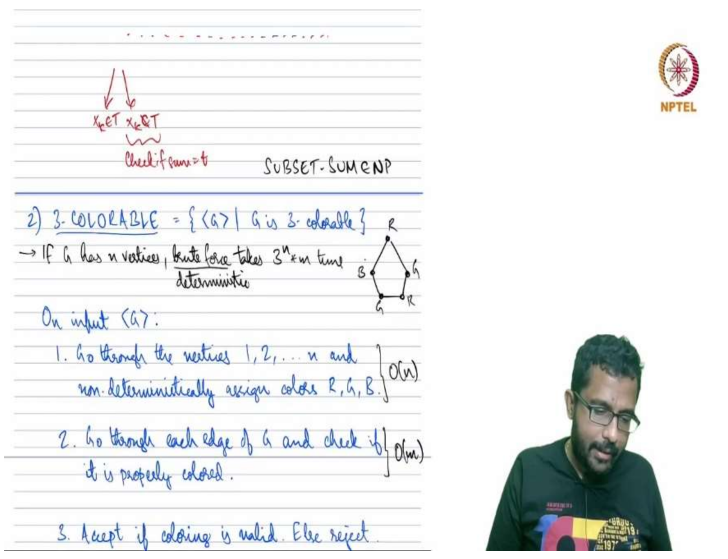
CNF: AND of OR's. AND of clauses where each clause is an OR of literals.

CNF-SAT $= \{ \langle \phi \rangle \mid \phi$ is a CNF formula that is satisfiable $\}$

Above formula $\Psi \in$ CNF-SAT

3-CNFSAT or 3-SAT $= \{ \langle \phi \rangle \mid \phi$ is a CNF formula where each clause has exactly 3 literals, ϕ is satisfiable $\}$

SAT, CNF-SAT, 3-SAT are all in NP



$x_i \in T$ $x_i \notin T$

Check if sum $= t$

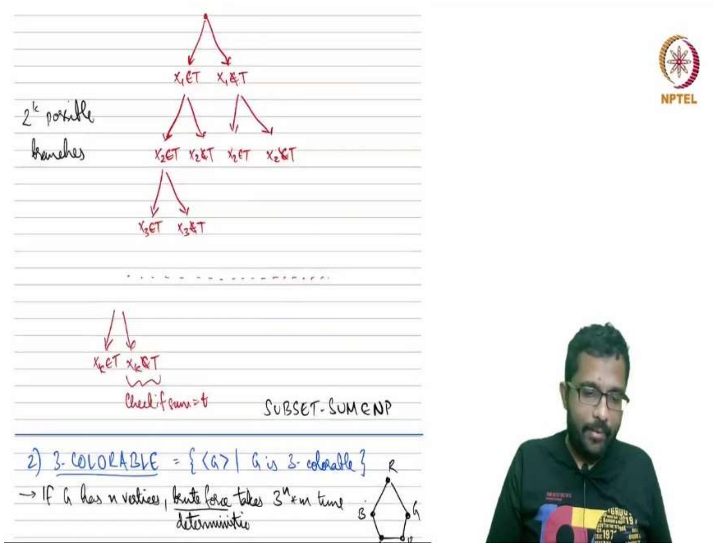
SUBSET-SUM $\in$ NP

2) 3-COLORABLE $= \{ \langle G \rangle \mid G$ is 3-colorable $\}$

$\rightarrow$ If G has n vertices, brute force takes $3^n * m$ time
deterministic

On input $\langle G \rangle$:

1. Go through the vertices $1, 2, \ldots n$ and non-deterministically assign colors R, G, B. $\}$ $O(n)$
2. Go through each edge of G and check if it is properly colored. $\}$ $O(m)$
3. Accept if coloring is valid. Else reject.

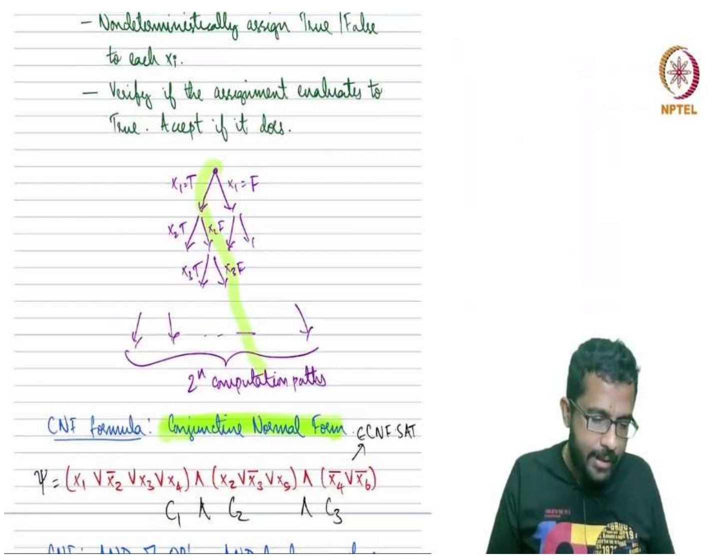


2^k possible branches

SUBSET-SUM $\in$ NP

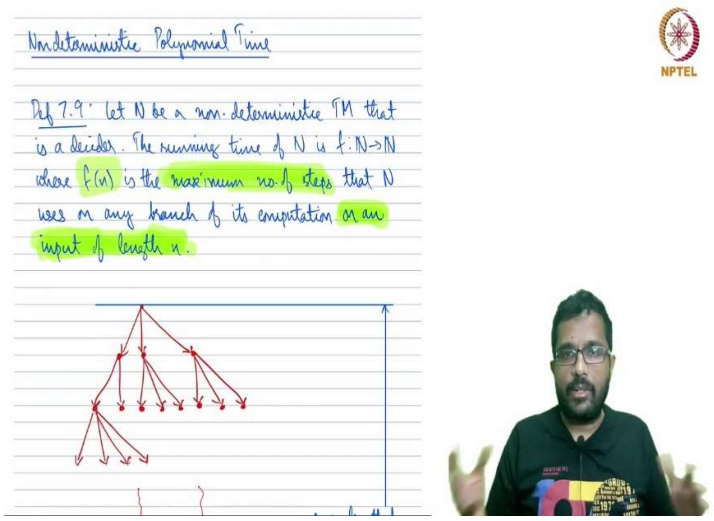
2) 3-COLORABLE = { ⟨G⟩ | G is 3-colorable }

→ If G has n vertices, brute force takes $3^n * m$ time
deterministic


- Nondeterministically assign True/False to each x_i.
- Verify if the assignment evaluates to True. Accept if it does.

CNF formula: Conjunctive Normal Form $\in$ CNF-SAT

$$\Psi = (x_1 \vee \overline{x}_2 \vee x_3 \vee x_4) \wedge (x_2 \vee \overline{x}_3 \vee x_5) \wedge (\overline{x}_4 \vee \overline{x}_6)$$
$$C_1 \wedge C_2 \wedge C_3$$


Nondeterministic Polynomial Time

Def 7.9: Let N be a non-deterministic TM that is a decider. The running time of N is $f: \mathbb{N} \to \mathbb{N}$ where $f(n)$ is the maximum no. of steps that N uses on any branch of its computation on an input of length n.

Each of these cases when we saw that it is in NP, the approach was the following. Let me just write that here. In all the above problems the approach, the NP approach was to guess the solution, we guess the solution and then verified it. So, that is an observation. Now, the question is now, are all the NP solutions like this, like if something is in NP does it always has to be this form

Or is it just that these 3 problems just happened to be guess and verify type? Are all the NP algorithms of the guess and the verify type? And the answer to that we will see in the next lecture. So, we saw what is non-deterministic Turing machines we saw NTIME, we saw NP and NP, we saw the P versus NP question and then we saw that 3 examples all of which had this guess and verify approach.

So, the question is, is it just that it is by accident that these 3 problems had the same guess and verify approach. So, I hope it is clear what I mean by guess and verify. In all these cases, in case of 3 colorability, we guessed the 3 coloring and checked whether this is a proper coloring. Here we guessed a subset and check whether the subset adds up to the sum.

Here we are guessing an assignment and check whether it is satisfying. So, the question is are all the NP algorithm of this type? The answer turns out that any NP algorithm can be converted into this type even if it is not of this type. But that we will see in the next lecture. And that concludes lecture number 46. This also concludes week 9, the lecture week 9.

So, we completed computability theory by seeing PCP post correspondence problem and an application which showed that it is undecidable to determine whether a given context free grammar is ambiguous followed by we began the complexity theory we defined the classes. We began time complexity we saw what is the $DTIME(t(n))$ and $NTIME(t(n))$, we saw P, we saw NP and then we saw examples for languages that are in P and NP.

And we will continue to learn about P, NP and other facts and aspects of time complexity theory in the next week. So, see you in week 10. Thank you.