Mathematical Logic
Prof. Arindama Singh
Department of Mathematics
Indian Institute of Technology, Madras

Lecture - 10
SAT and 3 SAT

So, we had seen cnf and dnf conversions and then started using them for verifying whether a proposition is satisfiable or valid. If the proposition is in cnf, then you can easily find out whether it is valid or not; if it is in dnf, you can similarly find out whether it is satisfiable or not. Then, to verify whether a dnf is valid or a cnf is satisfiable, we suggested that, we convert it to the other form by using distribution. But that is costly.

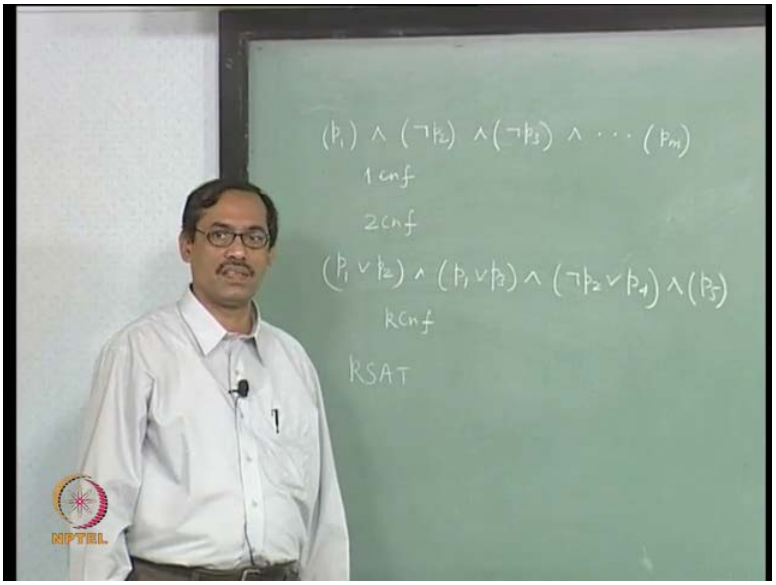
We try to see whether something can be done about that. It will take some time to do that, to do something in that direction, but there are some specific classes or sub-classes of these problems which can be solved easily. This easily means, here, whether you can solve it in polynomial time or you cannot solve it in polynomial time, the general satisfiability problem as it is.

We do not have any algorithm to solve it till now in polynomial time, but if you are told that it is satisfiable, you can verify that within some polynomial time or not. It is not exactly that; it tells that given an interpretation whether it satisfies the cnf. That can be determined in polynomial time. So, suppose you have a cnf, it has some clauses, these are the clauses and they are all there together, then you are given with an interpretation, it is told that this interpretation satisfies it. Now, you just want to verify this much, and then it should be easy. Why it is easy? Because, in that interpretation you will know what the values of the propositional variables are: 0 or 1. Just look at any clause there, in that clause you just take 1's or 0's. All that you have to see is whether it is giving 1, any of those clauses or not, then it is satisfiable, so that should be quick; is it clear?

It is just something like constructing one row of a truth table. You have a propositional formula, a proposition, you are given with an interpretation. Then, you are just constructing that row of the truth table; that is, it should take on linear time, well, polynomial time. But the general problem? From among all interpretations we are searching for an interpretation which will satisfy it; it looks to be exponential time, right? Because, number of rows in the truth table will be exponential in the number of variables.

They are not exactly in the length of the proposition given, or length of the cnf given, but they are related, so it can be done that way. What are the sub-classes where this can be done efficiently? One sub-class is: suppose you have the satisfiability problem, asked for one proposition which is in cnf, and in the cnf there is only one literal in every clause; so, basically it is a conjunctive clause. Now, it is a cnf, each disjunctive clause contains only one literal, one occurrence of a literal, the literal can be different in each clause.

(Refer Slide Time: 04:10)



For example, let it be given, see, it looks like p1, this itself is a disjunctive clause, there is only one literal in a clause. Similarly, there is another clause, it can be say not p1 or even not p2. Let us say, there is another clause again having one literal and so on. It looks this way, there is only one literal in a clause, not necessarily the same literal, still the thing should be easier to handle. Whether this is satisfiable or not?

The thing is, if p1 is there, not p1 should not occur, then it is satisfiable. That is the same thing should be true for all the literals coming up there and that should be easy. So, this is a sub-class. These formulas or these propositions are called to be in 1cnf; in each clause, there is only one or there are atmost one literal present. There is no disjunction in any of the clauses here; this is a 1cnf. Similarly, you can have 2cnf, we will say that each clause in this cnf will have atmost two literals.

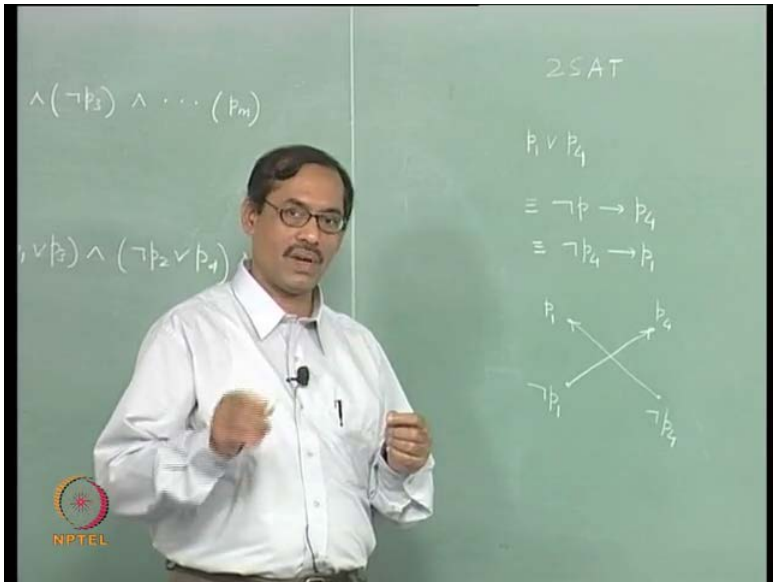
So, there is atmost one conjunction, there one disjunction there. For example, you take, say, p1 or p2 and p1 or p3. This is an example of a 2cnf. There can be only one literal

also. For example, I can have this also; this again is a 2 cnf. Similarly, you can define a kcnf, where in each clause there will have a atmost k minus 1 disjunctions, k literals; occurrence of literals is atmost k here.

In this case we say that the corresponding problem, satisfiability problem, is again coming in that number; we say whether, given a kcnf determine it is satisfiable or not. You see here, the general satisfiability problem is SAT, we will write that as kSAT. So, kSAT will be the problem: whether given any kcnf, it is satisfiable or not; that is the kSAT.

Now, you have an efficient algorithm for 1SAT, you can determine, given a 1cnf whether it is satisfiable or not. Similarly, we have an algorithm which is also efficient for 2SAT. If there are only two, maximum of two, literals present in each clause, then, you can solve it easily. There is a completely logical algorithm to that of 1SAT, but there is a graph algorithm which is easy to visualize, they are equivalent.

(Refer Slide Time: 07:41)

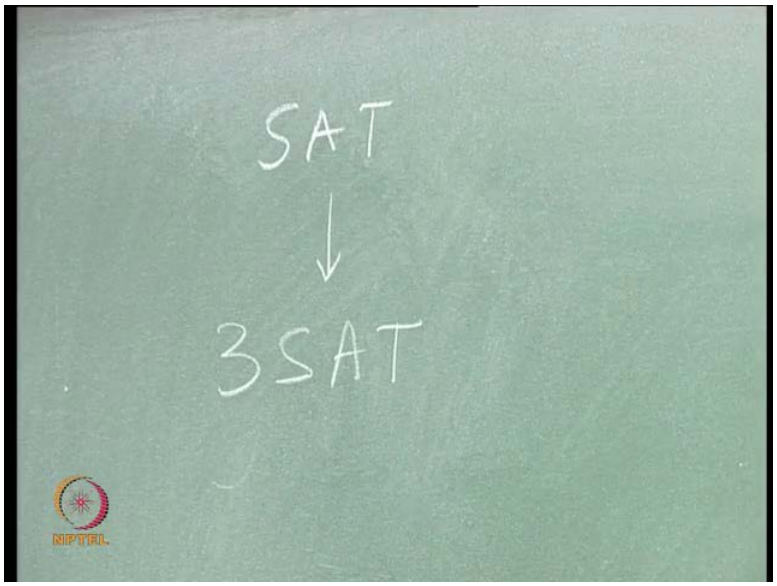


What it says is for the 2SAT, this is a simple observation, in 2SAT a clause will look something like this, where p1, p4 are literals, but not necessarily propositional variables, there can be negation symbols. This, you can write, equivalently as not p1 implies p4, because of double negation and implication rule: p implies q is not p or q, or you could have also written as not p, for implies p1. This is commutative, but implies is not; so we need both of them, you can write this one and you can write this one. Now, what you do,

given a 2cnf, we will construct a graph from this. Constructing the graph is simple; you look at a clause, for that clause, you will have either this or this, in fact both of them. For the clause p1 or p4, now what you do, take a node which is p1, let us label it is p1, take a node which is labeled as p4. Similarly, two other nodes which are labeled as not p1 and not p4. Once p1 or p4 is there, you have not p1 implies p4. From not p1 to p4 have gone a directed edge that represents your implies and similarly not p4 implies p1.

You have another directed edge, so what you do what the procedure is asking you to the given a 2cnf, first find out what are the literals coming there or what are the propositional variables. The propositional variables are p1 to pm, then you introduce 2m number of nodes, their labels will be p1 to pm, another set of this other m will be labeled as not p1 not p2, up to not pm. Then, if p1 or p4 is a clause, then you have these two edges in that graph. Now consider the graph, once the graph construction is over, this one will be unsatisfiable if and only if there is one path from some literal to its negation. Automatically there will be another from not of p to p. So, you have to go for a cycle. Then p to not p to p again. In this graph what we see is, suppose you get a cycle from some p to p itself, but coming via not p, and then the original 2cnf is unsatisfiable. It can be explained, can be proved. You just have a graph search problem, for the cycles.

(Refer Slide Time: 11:42)

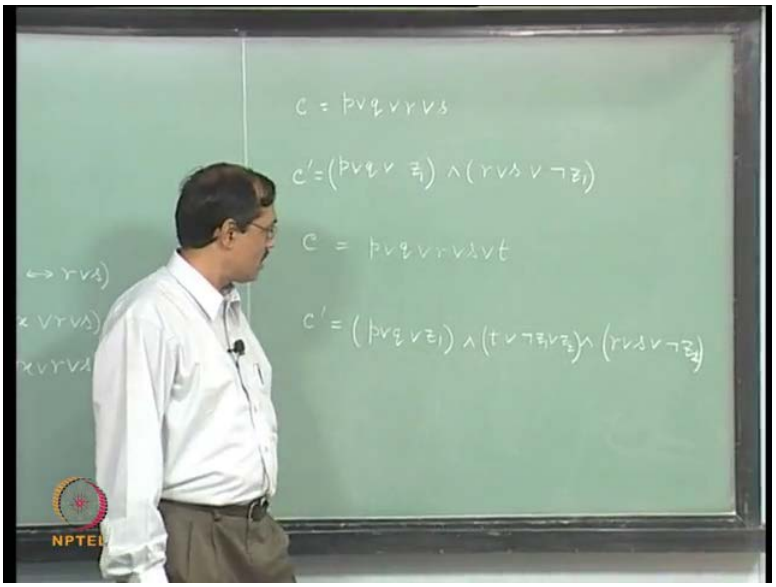


But coming to 3SAT there is problem; till now we have not discovered any algorithm which will solve a 3SAT instance in polynomial time; any arbitrary 3SAT instance. The

reason is, if you do that then SAT itself will be solved. If you can solve 3SAT in polynomial time, then SAT can be solved in polynomial time. There is a reduction. We need not go for 4SAT. If you want to solve this problem just considered a 3SAT, that is what it says. This is surprising because your sub-class itself is as powerful as the whole class; that is what it says. That amounts to searching for an efficient algorithm for 3SAT.

What it says is, 3SAT, there is a reduction from SAT to 3SAT; so an algorithm for solving 3SAT can be used to solve SAT itself, for that, that is satisfiable or not, only for the satisfiability determination. We are concerned with that one, they need not be equivalent.

(Refer Slide Time: 12:03)



Suppose, I have this p or q or r or s again. I look at a clause. Can I write this clause or corresponding to this clause, can I have another 3cnf? Let us see. Where I can show that this is satisfiable if and only if that target 3cnf is also satisfiable, this what I want, not exactly equivalence, only satisfiability is to be preserved, that is our aim. What I do is, I take p or q or another say z1, this I construct; or I write, next I write and r, r, s or not z1. So, to introduce this way, now question is, suppose I write it as c and write this as c prime we can say that c is satisfiable if and only if c prime is satisfiable.

We can do that, for otherwise, we try to prove it, let us try to prove. Let us take this as a conjecture and try to prove. If you are through, we will try for the next one, we will generalize. If not, then we have got a and stop it there, leave you there unhappy. Let us

see what happens. Suppose, this is satisfiable; that means at least one of them is 1 that is can be 0, allowed to be 0, all of them can also be 1, so at least one of them is 1, so is p is 1. So, this is 1, then this whole clause is 1; what about this? If it is 1, and through. Suppose it is 0. When can it be 0? When r is 0, when s is 0, when not z1 is 0, but not z1, I have freedom; it has not been fixed by the interpretation, it is a new proposition variable. So, question is can I extend the interpretation that satisfies to another giving new values for this new propositional variables, which will satisfy that, you can do here. I can take not z1 to be 1 by taking z1 as 0, if I take z1 as 0 it is still 1 and this also confirmed, so it is possible; it is satisfiable for the certain values certain interpretation of p, q, r, s and such that for the same interpretation of p, q, r, s, the initial proposition will be satisfiable.

There is some gap; there I have to prove c is satisfiable, if c prime is satisfiable. So, suppose I tried the first one. If c is satisfiable, then c prime is satisfiable. For that itself what I start with ia, a model of c. Now, what I do, I am to find one model of c prime, its connection is that model in the abstract case, but usually what should I do? I should try to see whether I can modify the given model to make a model for this. In fact the theorem does not ask you to take that model and then find the model for this, but that is a need for us, for the proof. Any model can do for this. Now what I do, I can give a procedure, but I can start with that model itself and assign z1 in such way along with that. It is an extension of that model so that it will be a model for c prime, that is all is used, there, model that you used in a previous one. Then, using the satisfiability condition of the previous one and then saying this must be satisfying if that. So, that is one part. You have to prove both the things: if c is satisfiable, then c prime is satisfiable that is our first. Second thing, if c prime is satisfiable, then c is satisfiable. For the first one itself if c is satisfiable, it does not matter. So, you start a model of that, now I have to construct a model for c prime. It might come from that model, it might not come, does not matter, but since it is there, let me use it.

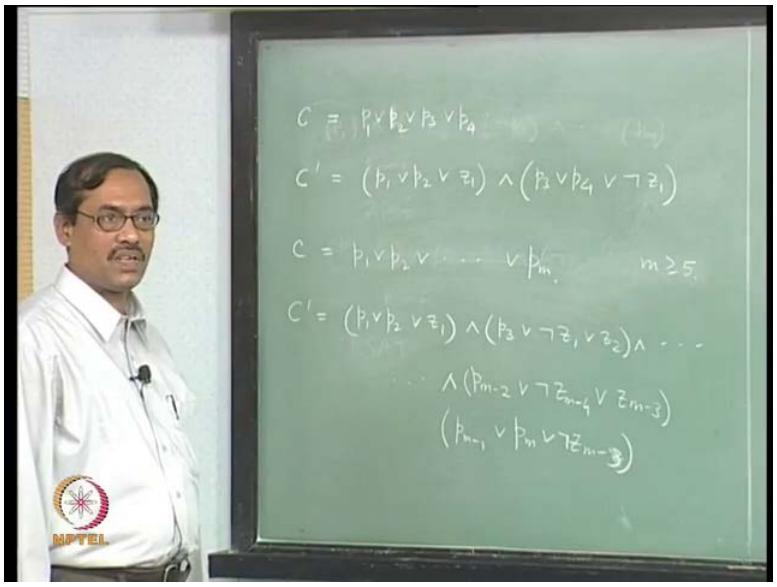
So, I see that I can extend that model to include a valuation for z1 for the new variable which will be a model for c prime, is that so? It looks viable. In that case. What about the converse? Suppose c prime is satisfiable. That means both of these are assigned to one, p is 1, q is 1, z1 is 1, r is 1, s is 1, not z1 is 1; in the same interpretation it is not possible. When you take a model of this, it says this clause is evaluated to 1 this clause is

evaluated 1, it does not mean all the variables are to be assigned to, here, it is our statement. The whole is assigned to 1, which says p is, at least one of these p, q, z1 is 1; so this says at least one of these is evaluated to 1, that is only given.

From this I choose p to be 1, that itself is c, let us see. Suppose c is p or q or r or s or t; for this part, we know our c prime will have p or q or z1 and r or s or not z. Now, you have to take care of this t. This is, up to this, I have to take care of this t. So, what I do, I have z1 here, I have not z1 there, so let me take, say not z1 or z2. I will make it z1, so this z1 and not z1 that is it. I do not want to leave that pattern, I want to keep it, so I have p or q or z1, then I take t or not z 1 or z2 and then r or s or not z2 as earlier.

In this order, as it is written and considering this way. First part. Let us see only second part, it should be through, it should be quick. First one. If this is satisfiable there is a model for this. How to construct this? How to give values to z1, z2 and so on? It is an extension. If p is 1, I keep it as it is. I have freedom to choose later, right? And I come to this: if t is 1, similarly I have freedom to choose z1, z2 later; if t is 0, then I have to choose one of this.

(Refer Slide Time: 21:22)



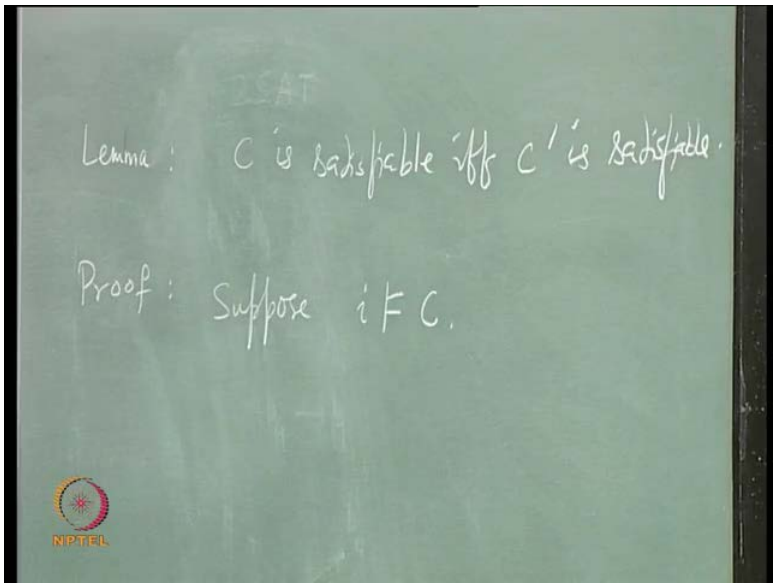
So, I take z1 to be 0, because it is not required, p is enough to satisfy it, so just let me take z1 0, then what happens here? z1 becomes 1 automatically, our construction will go like that. Automatically this one will become 1, but, z1 is 0 and z1 does not matter, so out of r or s there, can be false, there are considering the worst way, when it can be

breaking down. See, if r or s is 0, I have still freedom to choose not z2, so not z2, I choose to be 1, so z2 is 0, there is it, so this can be done, it looks like this.

I look in general. Suppose my c is equal to, let us take that first case, p or q or, I say, p1, p2 or p3 or p4. In that case I write c prime equal to p1 or p2 or z1 and we write or z1 or not z1. Next, we take p3 or p4 or not z1. Now suppose c equal to p1 or p2 or pm where m is greater than or equal to 5, then I leave that, inside pattern is coming, otherwise for four case, it is quick. Then, we write c prime equal to p or p2 or z1, and let us see, go next p3 or not z1 or z2.

It continues, this one pattern continuous till somewhere, after that we will come last to pm minus 1 or pm or zm minus 2, looking at this. This is the last term. If this is, may be, m minus 1. This should be two less, m minus 3, m minus 3, not of them. Yes, and that will be here, before it. Look at the case 5, minus 2, pm minus 2 or not of z before it, m minus 4 or zm minus 4, zm minus 3. Our proposed constants are. Now we should be able to prove that c is satisfiable if and only if c prime is satisfiable.

(Refer Slide Time: 24:08)

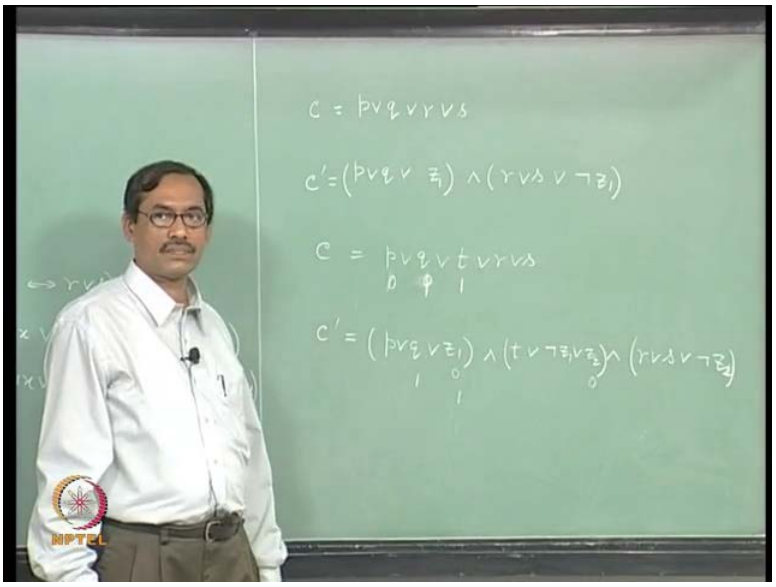


Let us try. We have this, c is satisfiable if and only if c prime is. Now what should we start? First part? Suppose c is satisfiable. Suppose, i is a model of c, so that means i is giving some values to p1, p2, pm, right? Some values, so that the whole thing is evaluated as 1. That means at least one of p1 to pm is 1; that much we know, others we do not know. Just look at our proof. What we did here for the case for 5, suppose q is 1, p

is 0, how are you going to do it? This is 1, this is 0, p is 1, that is the first case. We have taken, remember, p is 1.

This one we do not have to worry at all now, what about this? We choose this to be 0 so that not z1 can become 1 forgetting our t values, so in case of p to be 1 you have chosen z1 to be 0. Next, this becomes 1, now z2 can be chosen to be 0 so that the other one becomes 1, so z2 also we have chosen as 0. That itself does; we do not have to worry about what the other things are given q, r, t, s, you do not have to worry.

(Refer Slide Time: 26:19)

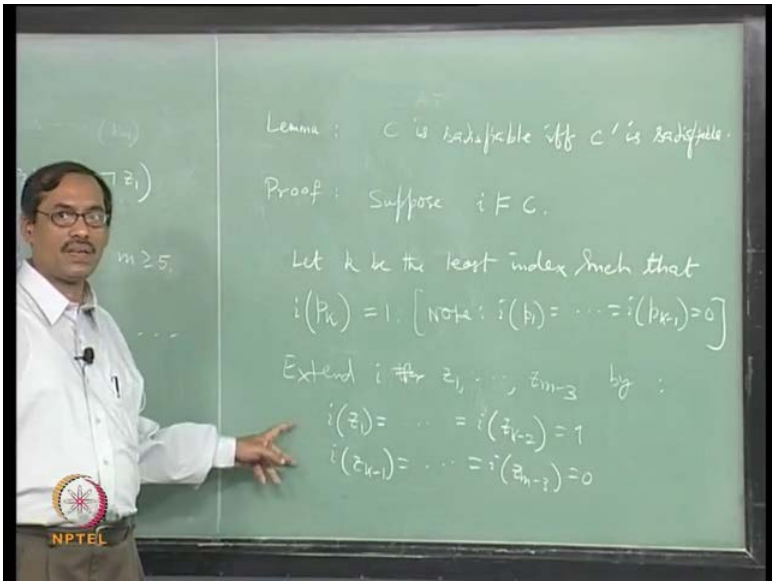


Looking at p alone, suppose p itself 0, say q is 1, now how are we going to start? From q itself? Forget p you do not have to worry, q is 1. Since this is 1, well, I take this to be 0 associated with it whatever that variable is, same thing, does same thing hold? Suppose instead of q is 1, I have p is 1, q is 0, now I have to satisfy this first. So, z1, I have to take one, there is an option now, so then t comes here q comes here, and z1 is freezed, then that onwards the other scheme works, z2 is 0 as it is.

There is no problem. Do you see the pattern? If p, q; if p is 0, but q is 1, then I have to take z1 as 0, q onwards, that is, if a variable p2 is 0, p1 is 0, then I choose z1 to be 0 and then take z2 to be 0, z3 to be 0. Now, all z's assigned, just take 0 whatever earlier model, that works. If t is 1, that is, p3 is 1, so that is, up to p2, let us say, they are all zeros. Then what I do, I have to take z1 as 1, all the others are 0's. Now, can you tell me the scheme. True, that many minus two number of 1's and then remaining 0's. Suppose, I call this as

k, before that all of them are 0's, but pk is 1, right? Then what I do, I take z k minus 2 as 1 till that. Before that, say, z1, z2 up to zk minus 2, that I take as 1, all the others are 0's that should work.

(Refer Slide Time: 28:42)



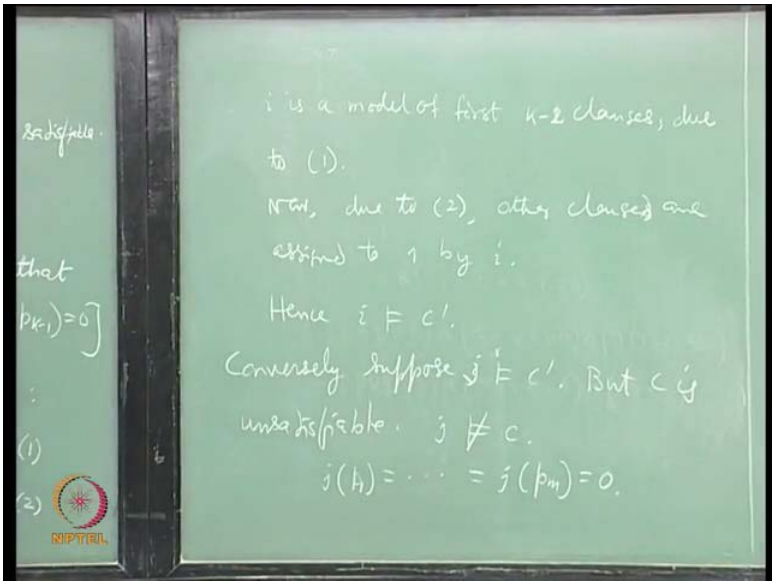
Let us see. I will write a, suppose I, use a model of c, in this model, let k be the least index such that pk is 1, all others before it are 0 such that pk in i is 1. So, once we say least index, it means before that all of them are 0, it is the least index such that i assigns it to 1. So, that means i assigns all the variables before it to 0, is that? If not clear, write it. Note: i of p1, i of P k minus 1, are 0; that is what it says. It is the least index and all the others we are not bothered. Now, what we do, extend i to, or for z1 to m minus 3, m minus 2, up to m minus 1, we have involved pm.

Then, z m minus 3, z m minus 4, so on up to z m minus 3, z1 to z m minus 3, what to do, suppose, which is the k from, then you want to put up to k minus 2 as 0 all the others are 1, is that, sorry, up to that 1, and 0 onwards, 0 after that. So, we say i of z1 equal to i of z k minus 2 as 1 and i of z k minus 1 to i of z m minus 3 is 0. This is our new interpretation i, though we are writing the same i for both the things, it is an extension, it is a new interpretation. Now, in its domain z1, variables z, variables which were not earlier; in that way it is different, but it is an extension, it keeps the same values to pk's as earlier.

Now, we should say that i is a model of c prime; it is only a construction, you say only up to some 5 we have to realize for it. So, our claim is now i is a model of c prime, this c

prime. Now let us find out why it is a model of c prime. For the first clause what happens, i of this up to z k minus 2, k minus two. So, first k minus two clauses are evaluated to 1, now i is a model of first k minus 1 clauses.

(Refer Slide Time: 32:23)



When you wrote c prime that way, you are not going to rewrite it, you have to write it that way only, so that first, second, they are meaningful. Why? The one you have there, you are giving 1, k minus 2, so you have to say k minus 2 clauses, due to first condition. Let us say first condition as one, second condition as two and so on. Next? What do we concern about? k minus first clause and onwards, so k minus first clause has what? pk, so pk is assigned to 1. Due to two other clauses are assigned to 1 by i, hence i is a model of c prime. That is the difficult part we have. From this, well, now the converse part. Conversely, suppose, i is an extension, so i is a model of c prime. Now, you must find a model for c so what to do here. There will always be one. There will always be 1, there will always be one clause as there, we have between the k the k minus 1 clause wherein all the z's will not contribute to that, being validated. It is because of the literal. Since we can always have a clause, therefore, the initial c is satisfiable.

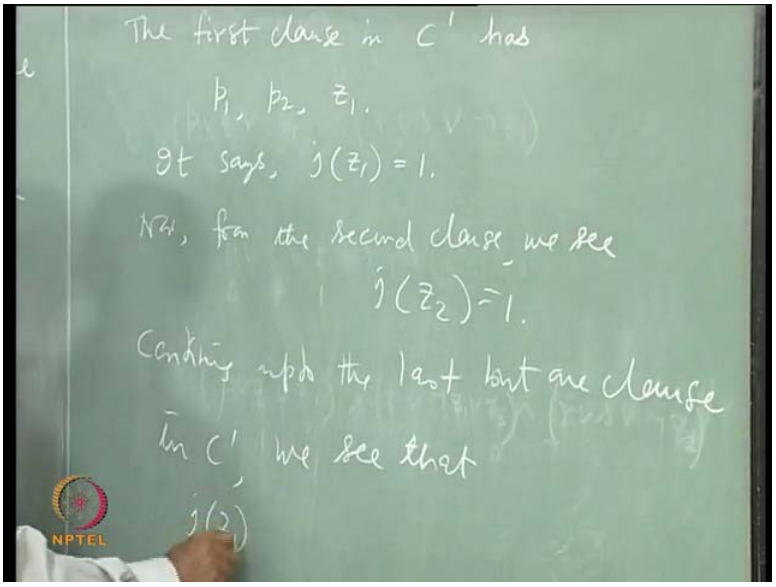
The idea is, if you look at the z's you have one z1 and then its negation; if you look at all, z m minus 3. Then, also its negation, so suppose you have a model for c prime it will assign z's to something 1 or 0 whatever it is. So, whenever one is assigned 1 the other is assigned 0, so in the same model, therefore in this case. Suppose, see what not z m minus

3 is assigned to? 0. Then, one of this will become 1; you have to proceed similarly, this is difficult to write, so you have to write it exactly.

That z1 and not z1 or z2 and there is you want to prove a contradiction, so in that case you assume that i is a model of c prime, i is not a model of c or there is no model of c rather; right? Not that the same i is mj. There is no model of c, this is what you want to say, is it? The literals from the previous, all the p, all the P, i is removed from the c dash, so we will have a new construction.

That is z1 and not z1 or z; now we can show that it is unsatisfiable; from that we can show that it is unsatisfiable only if one of those literals is one, that is. So, remove them; it means, assign them to 0, is it not? That is what you are telling; so better assign them to 0, that i be expressed, be it so, that is same thing as assuming c is unsatisfiable. Let us try. Suppose j is a model of c prime, but c is unsatisfiable. So, once c is unsatisfiable whatever model you choose, whatever interpretation you choose, it is not a model of c.

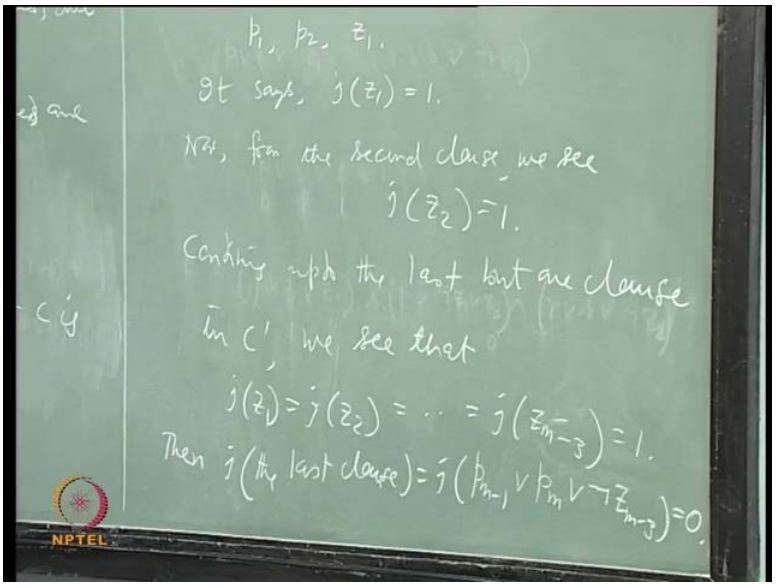
(Refer Slide Time: 39:00)



Therefore, j is not a model of c prime; this one j is not a model of c prime, model of c, right? It is not a model of c, so which means j assigns 0 to all those variables; so that means j of p1 equal to j of pm equal to 0. Now, look at our construction. All these are zeros, everywhere, up to this; all these are 0, but c prime is satisfiable, so starting from the first clause you have to take j of z1 as 1, is that right? Once j of z1 is 1, in the second clause p, three is not z1 is 0, Therefore z2 has to be 1; see our construction of least k, it is

really helping us now. The same pattern is coming here, so that means you see all of these, z1 to all of this, zm, i there has to be 1. If they are 1, then the last clause becomes zero therefore, j is not a model of c prime which is contradiction, so let us write it. Then, the first clause in c prime has p1, p2, z1; it says j of z1 is 1, now from the second clause we see j of z2 is 1, so continuing up to the last but one clause in c prime.

(Refer Slide Time: 40:00)

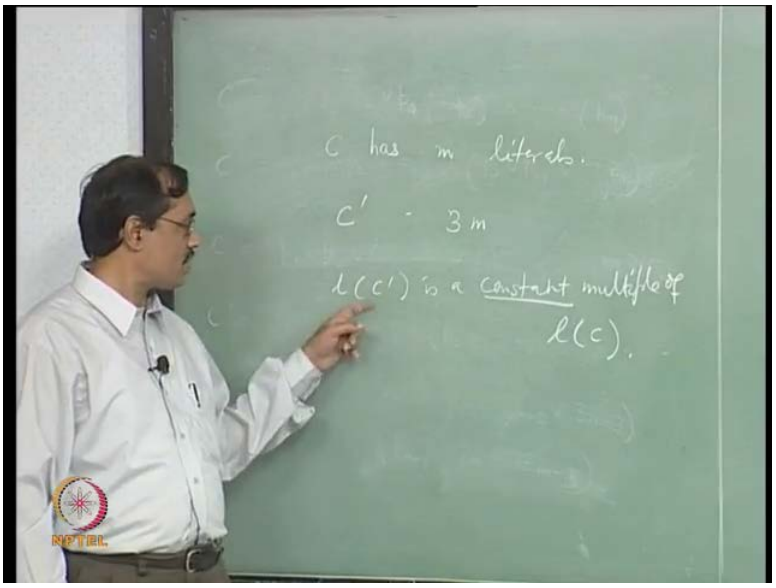


We see that j of z1 equal to j of z2 equal to j of z, what is that, m minus 2, m minus 3, z m minus 3 is 1, then j of the last clause, this is j of pm minus 1 or pm or not of z minus 3 is 0. So, this contradicts the assumption that j is a model of c prime. That means we have shown that c is satisfiable if and only if c prime is satisfiable, c prime is a 3cnf. So, that means a clause is satisfiable with m number of literals if and only if a 3cnf is satisfiable and we have given the construction for the 3cnf.

Now, if you start with any cnf; each clause you can write as a cnf, corresponding primes, then go on conjuncting them together. So, you end up with a 3cnf. Now the question is will this 3cnf have exponential length corresponding to the length of the original cnf?

Let us look at a clause; for each clause, how many clause, you have given in the 3cnf. Suppose c has, c has m literals. In c prime atmost three m occurrences of literals; each one is 3. So, atmost 3m; are there exactly 3m, really less, right? It will be less. So, let us write atmost 3m occurrences of literals, so that means length of this and length of this is just a constant multiple; length means total number of symbols occurring in it.

(Refer Slide Time: 42:03)



So, length of c prime is a constant multiple of length of c; and this constant is fixed for any given clause, it does not matter whatever it is, right? Why? Because you may start from any given cnf. In the cnf, first find out what is the maximum number of literals occurring in a clause, this constant will depend on that. For the given c you can find one constant here 3, 4 or whatever, so that constant remains for all; it does not change throughout ,when you write for clause in 3cnf. It takes the maximum possible, so since it is a constant, now if SAT can be solved for this efficiently, SAT can be solved for this efficiently. It is just a constant multiple, length is not making it exponential times the length of that; it just a constant multiple.

Therefore, SAT can be converted to 3SAT. If you solve any 3SAT, that will solve SAT. It is an NP problem, because SAT is NP-complete, 3SAT is also NP-complete. If we can solve 3SAT, our proof that 3SAT is in P, that will show NP equal to P. If you say 3SAT is not in NP, yeah, that will show that P is not equal to NP, here completeness is required. It is completeness that shows. Because SAT is NP-complete, 3SAT is also NP-complete. SAT is in P or not in P determines whether P is equal to NP or P is not equal to NP. Let us stop here today. We have done this much that SAT or 3SAT, they are equivalent in the sense that 3SAT, if it is proved to be in P or not, then P and NP, their fate also will be decided.