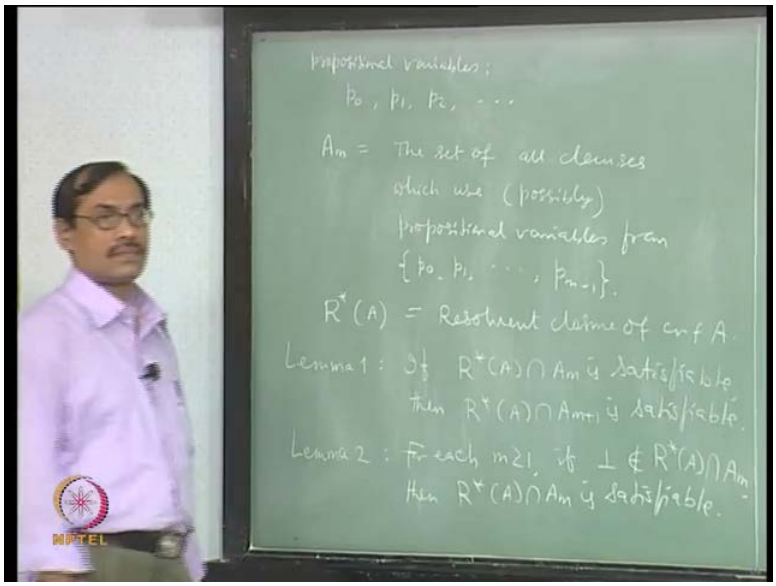
Mathematical Logic
Prof. Arindama Singh
Department of Mathematics
Indian Institute of Technology, Madras

Lecture - 14
Adequacy and Resolution Strategies

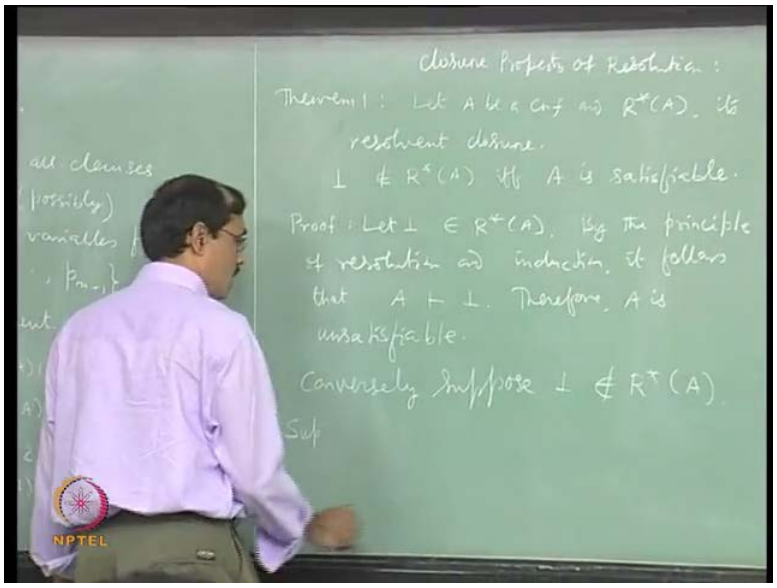
(Refer Slide Time: 00:10)



We are at the enumeration of propositional variables. Then we introduced one special set of cnf's. We call it Am, which is the set of cnf's or rather clauses. Am itself is a cnf. It is the set of all clauses, which use, so it is really possibly use, they may not use everyone of them, but they do not use anything beyond these things. It uses propositional variables from the first m variables. Then we introduced the resolution closure of a given cnf A, R star A. This is the resolvent closure or resolution closure of the cnf A. Let us write it resolvent closure of cnf A.

Then we came across two results. The first result was if R star A intersection Am is satisfiable, then so is R star A intersection Am plus 1. The next result was in better connection with what we are going to prove, but there we said that if bottom does not belong to R star A intersection Am, then it is satisfiable. For each m greater than or equal to 1, if bottom does not belong to R star A intersection Am, then this set R star A intersection Am is satisfiable. These two results we had proved. Now, we are going to the proper result what we wanted. So, let us write it as a theorem. We know bottom does not belong to R star A intersection Am.

(Refer Slide Time: 03:26)



Now, we should show that if bottom does not belong to R star A itself, then R star A will be satisfiable. But our main interest is in A, not in R star A. We will rather write A is satisfiable and also conversely, that is the next result. All that we need is A is a cnf; let A be a cnf and R star A its resolvent closure. We say that bottom does not belong to R star A if and only if A is satisfiable. The proof should be easier now. Tis result is called closure property of resolution. In some sense, it is closed that means once bottom is there, you know something. If bottom is not there, you also know something. That way, everything is done. That is why; it is called closure property of resolution. In some sense, this is also the soundness and completeness of resolution, but we will come to see it in a minute.

So, for proof, how do we start? Let us say bottom belongs to R star A. Can you say that A is unsatisfiable? This is what we should have if the statement is correct. If bottom belongs to this, then this would be unsatisfiable. That is one part. If bottom does not belong to R star A, then it is satisfiable. That is the other part. We are taking the contraposition, not proving it directly. Suppose bottom belongs to R star A. Then what happens? R star A is unsatisfiable. That is clear, but how to say A is unsatisfiable? Well, suppose from A in one step, you have got bottom. Can you say A is unsatisfiable? Why?

Student: Because A is…

Yes, because once bottom has been obtained in one step, A entails bottom. Suppose it has been obtained in two steps.

Student: Then also, it entails.

Then also, A entails bottom. Therefore, A is unsatisfiable. So, it is induction. Is it clear? It is induction and the principle of resolution which says that if A or some w has been obtained from A by resolution, then A entails w. So, bottom has been obtained from A. That is what it means when you say bottom belongs to R star A in some finite number of steps. We need induction there. We will not give the details here. We can just give a comment that by the principle of resolution and induction, it follows that A entails bottom. Is it clear? Therefore, A is also unsatisfiable. So, this is really the soundness of resolution. If something has been obtained, it is really entailed by those. That is the soundness.

Now, we are going to do the completeness. Conversely, suppose bottom does not belong to R star A. Now, you should tell me looking at the two results Lemma 1 and Lemma 2.

Student: A is satisfiable.

That is what we want to prove. What we have proved is, if R star A intersection Am or if bottom does not belong to that, then that is satisfiable. It should be clear now because A itself…

Student: R star A as Am union, sir, union, intersection of Am with R star A also does not contain bottom. Therefore, it is satisfiable.

So, R star A intersection of Am is satisfiable. Why A?

Student: Therefore…

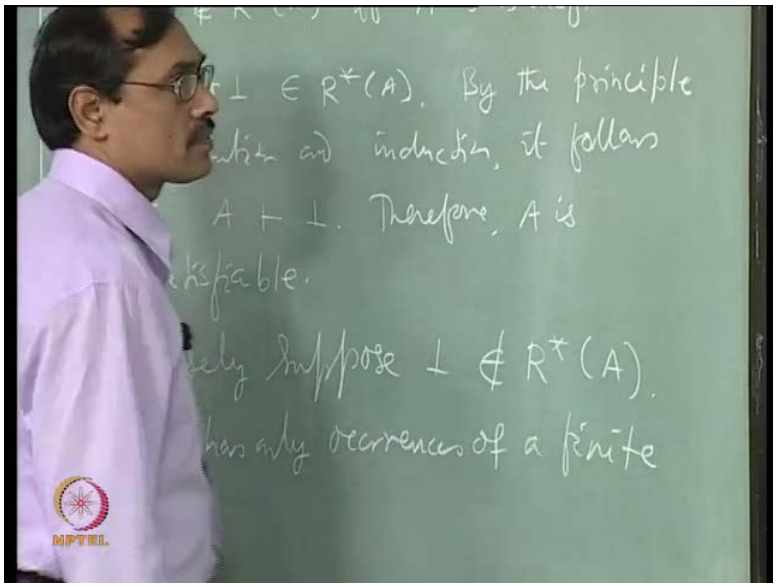
Why A?

Student: This is equal to the number of propositional variables in A.

Yes, that we have to take. See, A is the cnf. So, A has some propositional variables. There is a finite number of propositional variables A uses. Once A uses only finite number of propositional variables, then we can say all the propositional variables belong to some set p0 to pm or pm minus 1. Is that okay? Say up to p100 you are using, so take the set p0 to p100. If only p 100 is there, you take p0 to p 100. Is that clear?

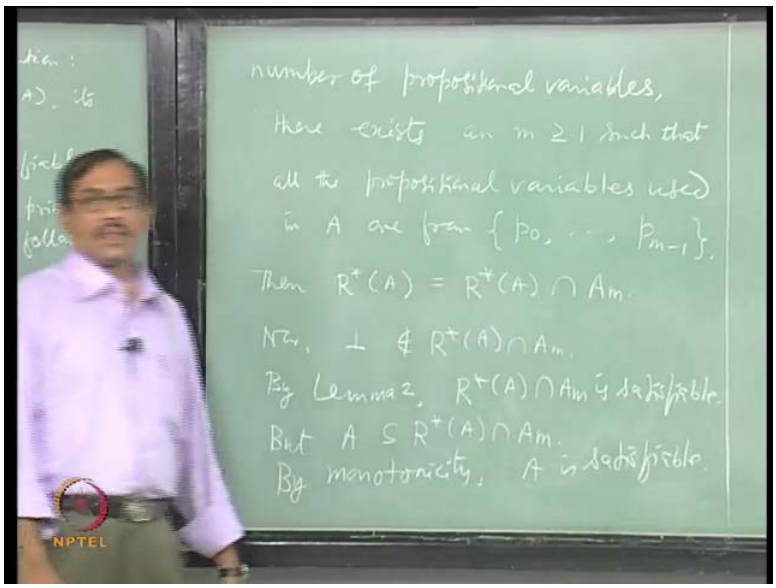
So, there from will start.

(Refer Slide Time: 09:48)



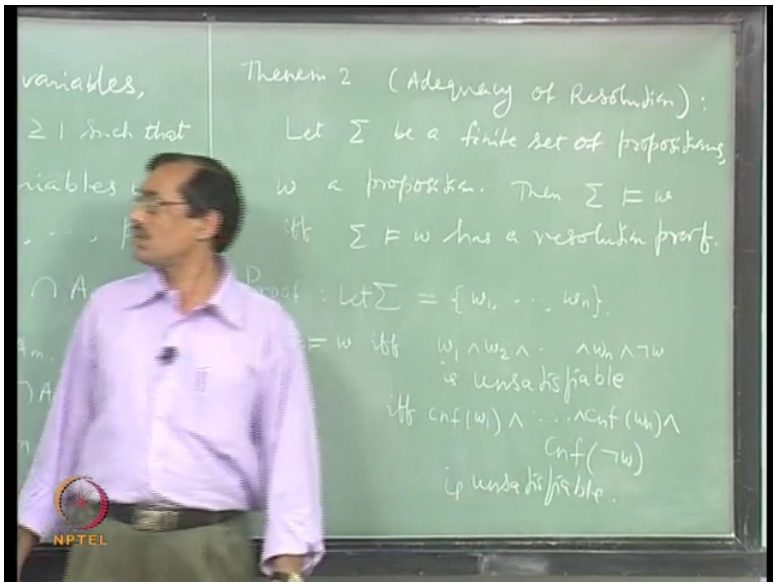
Suppose or rather we can give a reason instead of assuming. Since, A has only occurrences of a finite umber of propositional variables, there exists some m, which is greater than or equal to 1 such that all the propositional variables used or occurring in A are from p0 to pm minus 1. Then R star A becomes equal to R star A intersection Am because there is no other propositional variable up to Am, you have in all the clauses. All the clauses in R star A are in Am also. So, those two sets become equal. Once it happens, now bottom does not belong to R star A intersection Am.

(Refer Slide Time: 10:09)



By Lemma 2, R star A intersection Am is satisfiable. Is that right? But, A is a subset of this. Therefore, A is satisfiable. Directly you can see there is an interpretation i, which satisfies every clause in R star A intersection Am. A is a subset of that. So, this same i satisfies every clause in A. This is really monotonicity. So, by monotonicity, A is satisfiable. That ends the proof of closure property. So, once you have this, you can see adequacy of resolution, soundness and completeness, in fact this is, but we have to formulate in another way.

(Refer Slide Time: 13:07)

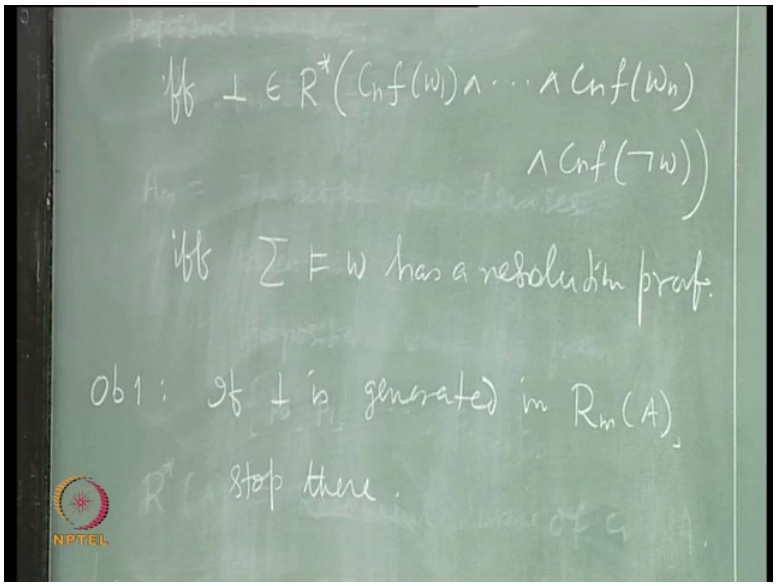


Let us write it. Let sigma be a finite set of propositions, w a proposition. Then what do we want is, sigma entails w if and only if sigma entails w has a resolution proof. Which says that entailment is completely captured by resolution. That is what you wanted. Now, proof should be easy. Yes. Can you see the proof? You have to use the definition of a resolution proof. So, resolution proof of sigma entails w starts with a cnf representation of sigma; along with not w. So, you have to bring from that.

Let us try. Sigma entails w. Now, we know that sigma is a finite set. If sigma is a finite set, you can write sigma equal to set w1 to wn for some n. Let us start with that. It is finite. Now, you may say sigma entails w if and only if w1 and w2 and wn. Only you can take implies or you may say and not, not w is unsatisfiable. We are using reductio ad absurdum directly here. Then this happens if and only if you take the cnf representation. You may write cnf of w1 and cnf of wn and cnf of not w, is unsatisfiable. Then what next? See, you can take the whole

thing as A itself. It is a cnf and cnf and cnf and cnf and so on. It is a set of clauses. That itself is here A now. Now, apply the resolution principle, the resolution, closure property of resolution, which says A be a cnf, when we say bottom does not belong to R star A if and only if A is satisfiable. This says if this is you have A, then this is unsatisfiable if and only if bottom belongs to R star of that.

(Refer Slide Time: 17:05)



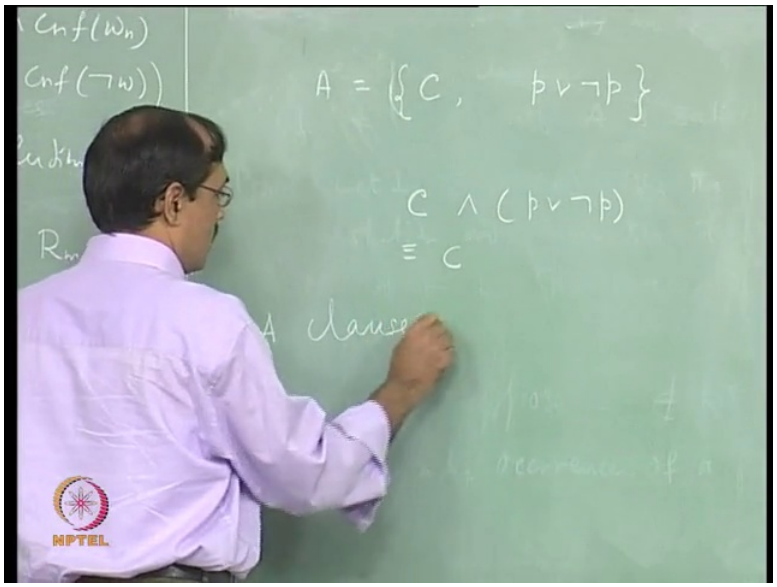
That says if and only if bottom belongs R star of A. This is, by Theorem 1. closure property of resolution, and this is exactly your definition for resolution proof. If bottom belongs to that, if bottom can be derived from this, then you say that that is a proof of sigma entails w. So, this says sigma entails w has a resolution proof. So, there is guarantee now. We are happy that our single rule, which is called the RPL, is enough. If the cnf is or set of cnfs is satisfiable, then we will never get bottom and we can show that we will never get. How to check that you will never get? This is because R star A becomes finite; somewhere it will terminate at a stage.

So, there is an algorithm to do it, but the algorithm is very inefficient. Now, you have to go on generating all the clauses and it is very crude also. That can be sharpened. Sharpened means somewhere some waste is being happening inside the algorithm, those wastes have to be thrown away. That is how, it will become efficient. What are the wastes, you have to find out, where our labor goes last. We are not getting anything out of our labor that we have to find out while we are generating this R star A.

One possible thing is suppose you have already got bottom. Suppose you have started with R1 A. In R1 itself, we have got bottom, but you know it is not R star A. If you go to R2 A, it is having some more clauses than R1. But, is there any need to go to R2 A? There is no need. We stop here because we wanted whether bottom belongs to R or not. Bottom belongs to it. We go for the bottom belongs to that. That is our first observation. Some simple observations like that will make it efficient, so let us see.First observation is this, that if bottom is generated in say Rm A, we stop here, some simple rule. You do not have to go for R star A. Now, this is easy.

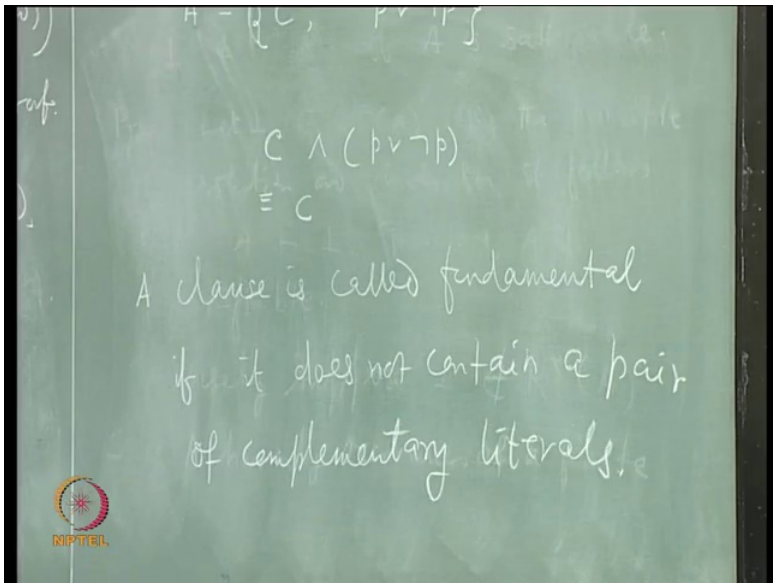
Second one is, suppose you have obtained one clause, which is tautological, which is valid that is, it is in the form p or not p or something else may be there. It is a clause. It is a disjunctive clause and it becomes valid when for some literal p, you have both p and not p occurring in it. If p is there, not p is there, some others may be, there, may not be there, that does not matter. Now, it is tautological. Once it is tautological, it means in the cnf, you add that clause with any other clause; you will get the other clause only.

(Refer Slide Time: 21:03)



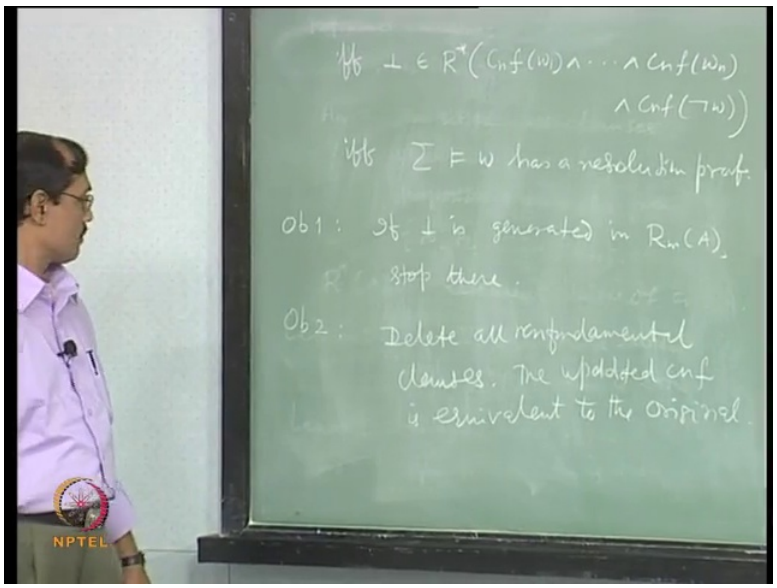
Do you see what I am telling? See in the cnf A, suppose you have some clause here, C and there is another clause. Let us write set for, there is another which is p or not p, some others may be there. Now, what happens? It means I have C and p or not p. We will take and with top. It gives only that. This will be equivalent to C itself. So, it is not only for p or not p or even something else. That does not matter. It will be altogether.

(Refer Slide Time: 22:22)



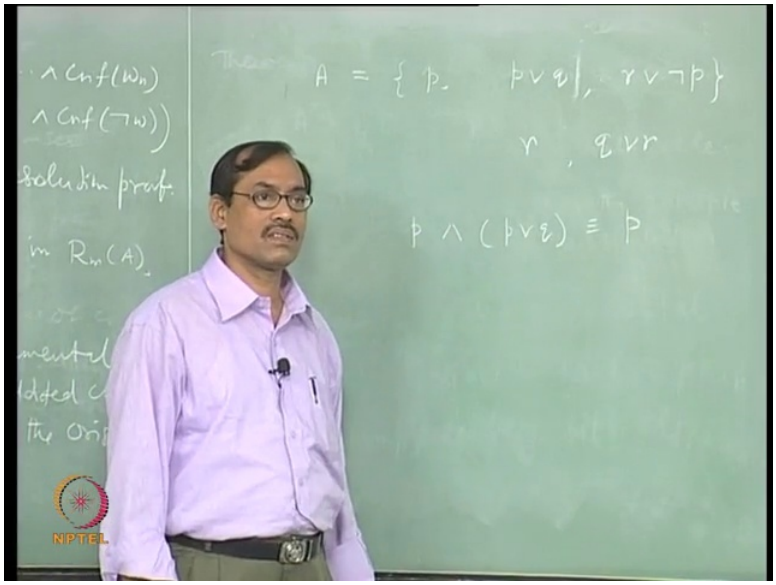
So, top or anything else is also top. It will be reduced to this. That means whenever you get such a clause here, there is a literal and its negation, we can say, to delete it. That is one of the wastes that we have to cut. That is our observation two. Such clauses are called tautological clauses, trivial clauses or non fundamental clauses. There are so many names. We call a clause to be fundamental if it does not have a pair of complementary literals. So, let us write it. A clause is called fundamental if it does not contain a pair of complementary literals.

(Refer Slide Time: 22:57)



Our observation two is, delete all non fundamental closures. Then the updated clause is equivalent to the original clause, not the clause, it is cnf. So, the updated cnf is equivalent to the original. That is why, we can delete it.

(Refer Slide Time: 23:56)

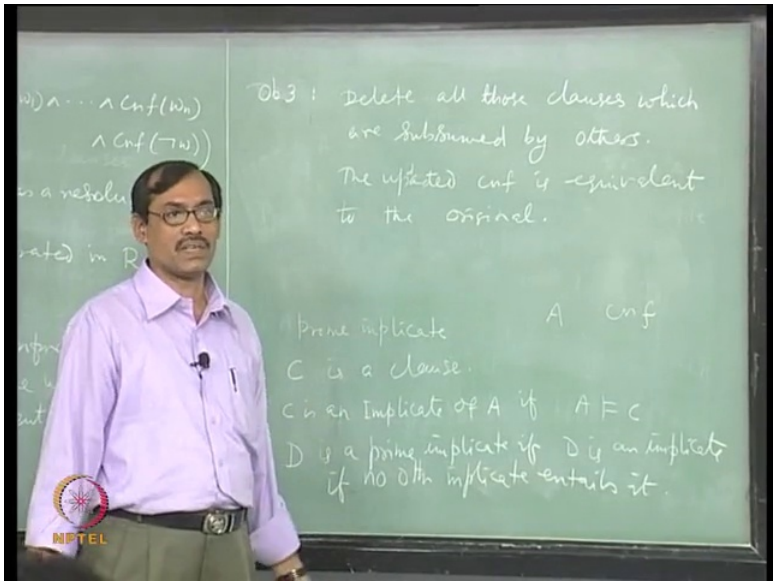


So, two types of wastes we have done away with. There is another kind of waste. Let us see what. Suppose in A, I have p, I also have p or q, I have something else. We take resolution. Suppose r or not p, we are going to take resolution. With p, if I take resolution, I get r with p or q, I take, I get q or r. So, r will be there, q or r will be there. See these types of things go on repeating. We will get one r again, one q or r, we will get something, some clause C and then you will get C or q. It will proceed, but you see from the beginning that the whole cnf is equivalent to p, r or not p. Why is it so? This is because it is united together. This is equivalent to p itself by the law of absorption. It is something like A intersection with a union B, so we will get A only. The same way, it is going. So, there is no need to keep this p or q and then get all these wasteful resolutions. Is it clear? So, that is our third observation.

You would say, but how to formally define it? So, our strategy is keep only a subset, the clause is enough, and delete all its supersets. That is what it says. All supersets will be deleted, p is a set now, singleton p. Then p or q is taken as p comma q set. If you take p, then you have to delete p or q; p, q set will be deleted. You can write that or you may say that a clause, you give a definition just like for observation two, a clause C subsumes a clause D if C is a subset of D, which means D is equal to C or some X. If you write in or form, it will

look like this. If you write in set form, it will look like this, C is subset of D. Then you say that C subsumes D. Here our strategy is, if a clause C subsumes clause D, then delete D. So, delete all those clauses, which are subsumed by others. That is what it says.

(Refer Slide Time: 25:34)



Delete all those clauses which are subsumed by others. Here also, equivalence is preserved because of that, p and p or q is equivalent to p. You say that the updated cnf is equivalent to the original. In fact, this strategy is very helpful in getting something else.

Those from electrical engineering can understand this better. They must have done Karnaugh maps and other types of things. Those kinds of algorithms cannot be generalized to more than four variables, Karnaugh maps for example. It is very difficult once you go for more than four variables, but something can be done even if there are more than four variables, where this subsumption helps. What is done is, that there you define a prime implicate. You define the prime implicate by taking, if C is a clause, you have a clause C and then you have cnf.

Let us take a cnf directly, say A is the cnf and C is a clause. You say that C is a prime implicate or let us say implicate first. You define in two stages. C is an implicate of A if A entails C. Then you say a clause D is a prime implicate, if D is an implicate, D is an implicate. What happens? You are telling that it is prime.

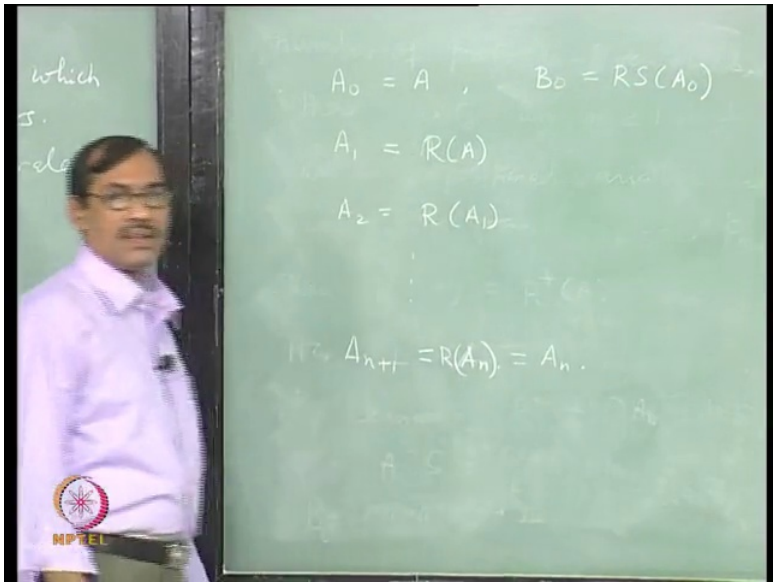
Student: If not subsumed by…

If not subsumed by any other implicate that is what you want to say. What turns out to be that D is an, D is a prime implicate if there is no other implicate in between them. If there is no another implicate, we can entail it. If no other implicate entails it that means between A and D, we will never get another thing. This entailment will assign. Then you say it is prime, so which turns out to be that coming to the subsumption because there are clauses. One clause entails another clause means one will be a subset of the other. That is why, subsumption comes in.

This set of prime implicates of a cnf can be computed by taking the residue of subsumption. Apply the subsumption tests, delete all those things, whatever remains that is the set of prime implicates because all those which can subsume, they have been kept. Those which are subsumed they are thrown away. So, all the non prime implicates have been thrown away. Whatever remains is the only prime implicates and that is equivalent to the original cnf.

That is what is followed usually for more than four variables. You use the set of prime implicates instead of going for the Karnaugh maps. This residue of subsumption idea can be used in the resolution itself because we do not want this unnecessary waste. We have to throw away all that clauses which are subsumed. Try to go back to the algorithm what we have done for computing R star A. There, you can modify and incorporate subsumption, this subsumption method.

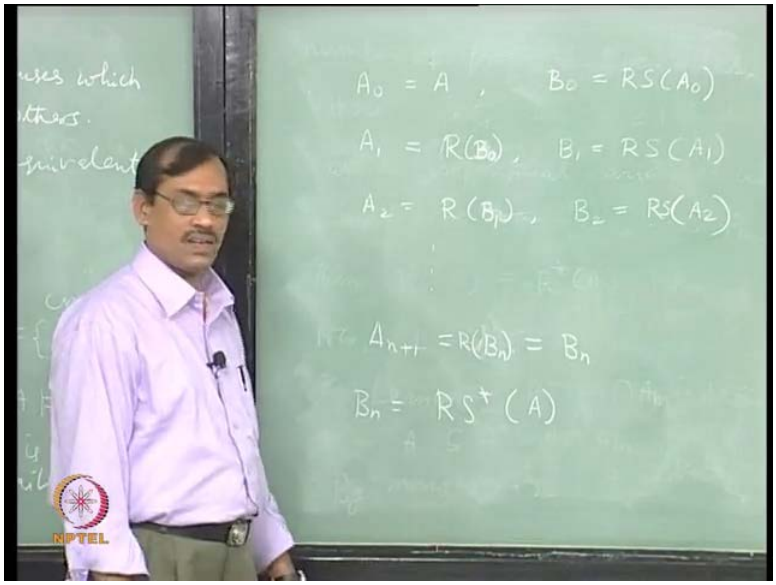
(Refer Slide Time: 32:16)



What we have done is start with A, call it A0. Then you take A1 as resolution R of A, then A2 as R of A1 and proceed. You stop when An plus 1 is equal to An itself, there you are stopping. This means An plus 1, which is equal R of An, which is found to be equal to An. Then you stop there. That is your R star A. Now, you want to employ a subsumption method. That means at every stage, you throw away the subsumed clauses, go on doing that. Here itself, you can start with the subsumption. Call it, say B0, which is equal to RS of A. So, RS we are writing for residue of subsumption, it means, throw away the subsumed clauses whatever remains is the residue. So, that is residue of subsumption.

Suppose we start A with the earlier example, let us take say p, p or q. Now, A0 is p, p or q. When I come to B0, I see that p or q is subsumed. So, I delete p or q. What remains is p alone that is RS is subsumed, residue of subsumption.

(Refer Slide Time: 33:59)



Then, while you come to A1 equal to R1 of A, again you write B1 equal to residue of subsumption of A1. But, then when you take R of A0 here, instead of A0, you start with B0 now. By taking A0, you achieve nothing. Again, some more unnecessary waste is there. So, do not go to A0, go back to B0 rather. You are losing nothing. Only residue of subsumption, we are getting at every step. Similarly, after getting residue of A1, instead of going to resolution on A1, you go to resolution on B1. It continues that way.

So, again you take B2 equal to residue of subsumption on A2. It continues that way. So, when you come to the last one An plus 1, you would write R of Bn. That should be equal to

An itself or you may say it is Bn. Bn is enough because from An, you have thrown out many things. You may not get them back. Let us write, that should be equal to Bn. When An plus 1 becomes equal to Bn, you stop there. That Bn will be called R, it is not exactly R star because many things have been gone. They have been deleted, unnecessary things. So, let us write that RS star of A. Subsumption is also used. That is what it says, RA star of A. so, incidentally these RA star of A will be the set of prime implicates. Everything is thrown away. What remains is that only. Here again, you can put that heuristic of looking at the bottom; your observation one, if anywhere bottom is generated, stop there. Instead of going for the RS star that is always monitored, so that the resolution becomes a bit efficient. Fine, but still everything is not over.

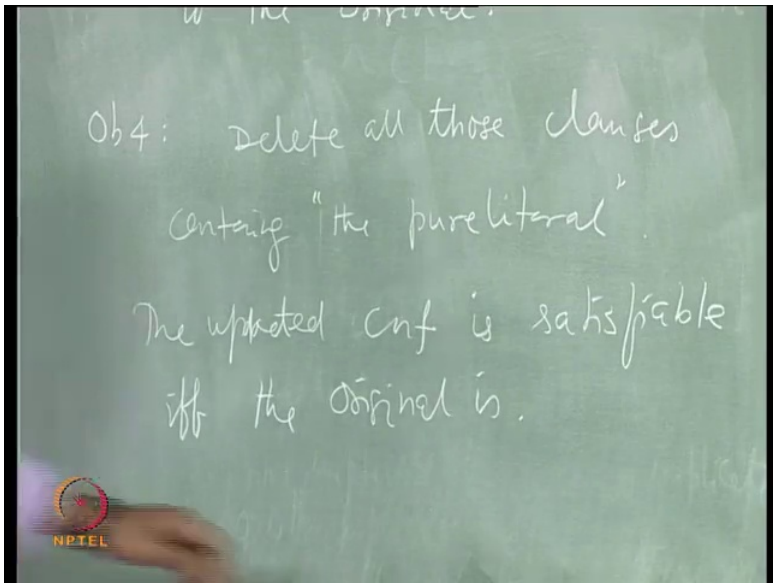
Student: Two at every step…

Yes, that will be the cost. You have to check only resolution, subsumption, always, every step. That is not in P because anyway, the problem is in NP. This will cost only some linearity there, some linear test, maximum and square check for this and n log n can be done, but every step. It does not matter. Now, let us see some more observations, which will help us. Suppose you take a cnf where in some of its clauses, a literal p occurs. Try to imagine this. In some of its clauses, the literal p occurs, but in none of the clauses, not p occurs. Such a thing is called a pure literal.

Now, when I want to see that whether this set is satisfiable or not, what I will do first? I will put that p equal to 1. I want to find a model for it. So, I, well simply take p equal to 1. It does not matter now and 1 or anything else will become 1. That is why; I am taking 1 instead of 0. If I take 0, then I have to delete that small p from everywhere, wherever it occurs, but if I take that p to be 1, then all those clauses wherever it occurs, they become true automatically. So, I can do away with all those clauses, that is still simpler.

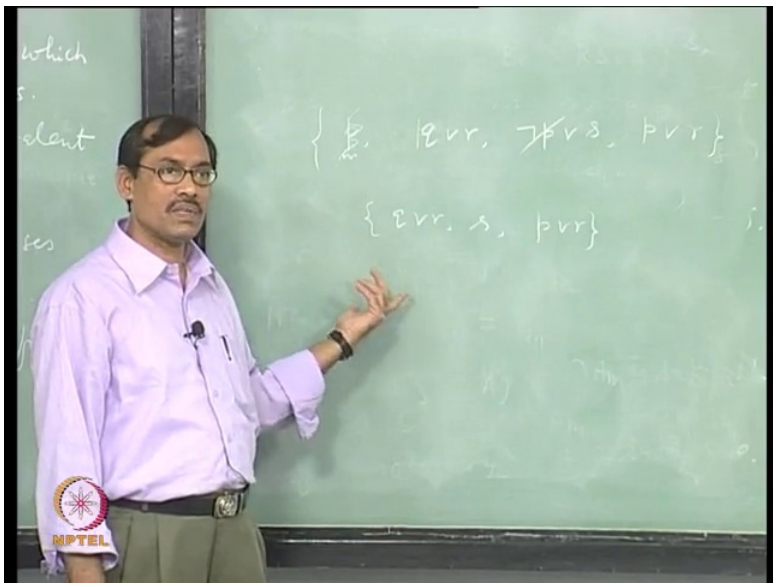
Now, if the remaining thing is also satisfiable, then the original will be satisfiable. Conversely, if the original is satisfiable, even if give 0 or 1, it does not matter now, this is satisfiable. So, you can always extend it to go back. Both the ways, it can be done. But, check that it is not equivalence preserving. If the updated one is A prime, original was A, A is not equivalent to A prime, but satisfiability is preserved. This is what we will be doing, as our next observation. They are called the pure literals.

(Refer Slide Time: 38:47)



Our observation four is, delete all those clauses containing the pure literal. This ‘the’ is ambiguous. This ‘the’ is ambiguous here. First, we have to find out what is a pure literal there. Then go for it. It is context dependent here. Then the updated clause, updated clause, updated cnf rather, is preserving satisfiability. It is satisfiable if and only if the original one is satisfiable. To see how does it operate, just see in the abstract at least.

(Refer Slide Time: 40:07



It looks something like this. So, I have not p or q, r or s, s or not p. This is my A. Now, I find that my not p is a pure literal. Nowhere, p is occurring; only not p is occurring. So, I identify

not p as a pure literal. Then what I do? From this A, I construct another set A prime where I delete all those clauses, which are having the occurrence of pure literal. That is all. This really simplifies a lot. Now, A is satisfiable if and only A prime is satisfiable. From this to this you know, you just give one and then you do it. From this also, extend the same way. Suppose it is satisfiable i of r, i of s is given. Now, you add to that i of not p equal to 1, i of p is 0. So, there is an extension, which is a model of it. This is sometimes called the pure literal heuristics. It is a heuristic we are applying. It is called pure literal heuristic.

The other one consists of the unit clauses. Unit clause is a clause having a single literal like p, not q and so on. A single clause, a clause is composed of that single literal, and then you call that clause as a unit clause. Suppose there is a unit clause. Then what happens?

Let us see an example. Here, p is a unit clause. It is not pure. It is not a pure literal because not p is occurring, but then there is a heuristic here also. What do you do? First, delete that unit clause. There is no need to keep it. Next, what you do? Wherever you see not p, delete all those not p's, not the clauses, delete not p, occurrences of not p. So, delete this. That is all we will be getting.

Student: Sir, p or p or r…

That is for subsumptions.

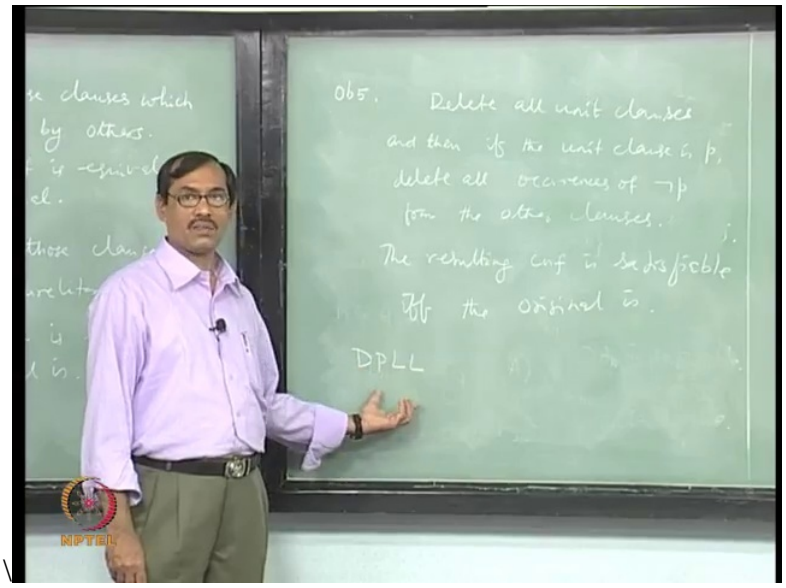
Student: Ok.

That is different. So, I am only explaining unit clause, not for this subsumption. If you apply subsumption still, it becomes more efficient. So, what happens here? Is it equivalent to this?

Student: Satisfiability.

It is not equivalent. Satisfiability is preserved. Basically, what you are doing is you have p, you have not p or s, you take resolution, you get s. So, that is what the deletion of not p say, but then p you are removing because after this, not p will never be coming there. So, p becomes a pure literal. You can delete p. Both the things are used simultaneously now. That is called the unit clause heuristics. So, what you do there is, first delete all unit clauses and then if the unit clause is p, delete all not p, all occurrences of not p from the other clauses. Now, it says that the resulting cnf is satisfiable if and only if the original is.

(Refer Slide Time: 43:49)



Student: Sir, the original entails the resulting cnf.

Original…

Student: Entailed the resulting cnf.

Yes, because you are deleting p also.

Student: Yeah, so the original will entail.

Entailed.

Student: Ok sir.

This p and something else, but this all we do not know, how to go there because p is there. So, it may not entail, because I can give another interpretation, p may be 0. It will not entail. So, this is another heuristic. In fact, there is an algorithm, which uses these two heuristics instead of going for resolution, only pure literal heuristics and the unit clauses heuristics, only those two. But, then these two are not complete. It will not succeed always. We need to do something more.

What they do is, you take arbitrarily another literal there, just choose one of the literals in the remaining one, when you are not able to use these two heuristics. Then give that value 1. In fact, you are doing something like a truth table, give that value 1. Then try to see. So, once

you give 1, you have a lot of simplifications, 1 or something will become 1 and so on. Then delete all those things. It is equivalent to deleting all those things. Then after that, what you do? Just use again those two heuristics, continue. If you find bottom, then its original is unsatisfiable. If you do not, then go back, go back, give that again 0 and start.

In the worst case, it can become exponential. That is fine. Everything in the worst case is exponential here. So, that is one of the other procedures. That is called DPLL procedure. In fact, this Davis-Putnam procedure was written first for the first order tautology, not for the propositional logic, which will do later. Later, these two people Longman and Loveland, they again included, and that became the DPLL algorithm for propositional logic. That does not use resolution, but if you use these two heuristics along with resolution, it is really efficient. Most of the cases are solved very easily.

Student: Most cases…

Worst cases are exponential and whether it can be done or not, we do not know. We have done it exponentially. That is all what it says. It does not say most case will be exponential. No. In these algorithms, there is another result, which says that whatever variant of resolution you are using, it does not matter, there is always a formula where it will be giving exponential result. This is a very strong result.

Student: Sir, essentially truth table cases.

Yes, essentially truth table case.

But, then we are interested in most of the cases, solving many cases, which are coming from practice. They can be solved.