

**Logic for CS**  
**Prof. Dr. S. Arun Kumar**  
**Department of Computer Science**  
**Indian Institute of Technology, Delhi**

**Lecture - 06**  
**Logic of Computer Science**

So, let us start today so I will I briefly describe Tautology Checker. Which is available for you to do this for you to play around with maybe it is written in ml but, besides I mean this is the only piece of actual program code that I will be discussing the rest you will be writing. So, it is a good idea to see how our whatever we have done so far is used in this tautology checking fact.

So, first day of course let us look at Tautology Checking. Let us go straight to this I am so say essentially, what we are interested finally in question of whether a certain argument is valid. That is the most important thing you have given a set of admissions then, there is some conclusion you come to and you want to know whether that conclusion logically follows from this from the set of assumptions. So of course, what typical informally striated argument something like this I mean it is really of full when you stated in natural language when there are all kinds of things I just read it.

(Refer Slide Time: 01:56)

**Arguments**

A typical informally stated argument might go as follows:

If prices rise, then the poor and the salaried class will be unhappy.

If taxes are increased then the businessmen will be unhappy.

If the poor and the salaried class or the businessmen are unhappy, the Government will not be re-elected.

Inflation will rise if Government expenditure exceeds its revenue.

Government expenditure will exceed its revenue unless taxes are increased or the Government resorts to deficit financing or takes a loan from the IMF to cover the deficit.

If the Government resorts to deficit financing then inflation will rise.

If inflation rises, the prices will also rise.

The Government will get reelected.

Therefore the Government will take a loan from the IMF.

NPTEL

Because, in case it is too small to read If prices rise, then the poor and the salaried class will be unhappy. If taxes are increased then the businessman will be unhappy. If the poor and the salaried class or the businessman will be unhappy the Government will not be reelected. Inflation will rise if Government expenditure exceeds its revenue. Government expenditure will exceeds its revenue unless taxes are increased or the Government resorts to deficit financing or takes a loan from the IMF to the cover the deficit. If the Government resorts to deficit financing then inflation will rise. If inflation rises, the price will also rise. The Government will get reelected. Therefore the Government will take a loan from the IMF. I mean now if this re argument that actually came in some news paper article the question or whether the conclusion is valid is from whatever assumptions that the author has made is extremely difficult to find them in.

So, to prove the validity or the invalidity of this argument and the conclusion is actually very hard. Of course, what there are something this written in natural language which means that firstly one has to sort of clean up ambiguity. So, one and so forth and may be one have reward it retranslated to some more sensible logical forms so that the ambiguities of language are eliminated that is one thing. So this is an common problem with all natural language kind of arguments fact.

(Refer Slide Time: 03:40)

**Validity & Falsification**

The validity of such arguments involves showing that the conclusion is a **logical consequence** of the hypotheses that precede it. The following alternatives exist:

1. Using a truth table.
2. Using theorem 4.2
3. Using one of the parts of theorem 4.3.

If the argument is not valid, then a falsifying assignment needs to be also given.

NPTEL

So, the other thing is in order to know whether it is valid. Basically what we are asking is whether this last statement therefore the government will take a loan from the IMF follows from all these other states all of them are decelerated in nature of course. And say essentially what we are asking is whether the last statement is a logical consequence of the previous statements.

(Refer Slide Time: 04:14)

**Logical Consequence: 1**

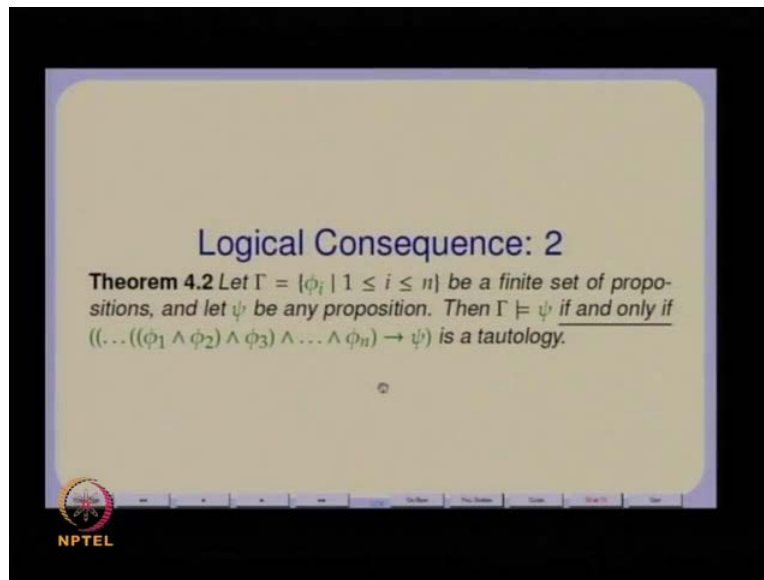
**Definition 4.1** A proposition  $\phi \in \mathcal{P}_0$  is called a **logical consequence** of a set  $\Gamma \subseteq \mathcal{P}_0$  of formulas (denoted  $\Gamma \models \phi$ ) if any truth assignment that satisfies all formulas of  $\Gamma$  also satisfies  $\phi$ .

- When  $\Gamma = \emptyset$  then logical consequence reduces to **logical validity**.
- $\models \phi$  denotes that  $\phi$  is logically valid.
- $\Gamma \not\models \phi$  denotes that  $\phi$  is not a logical consequence of  $\Gamma$ .
- $\not\models \phi$  denotes that  $\phi$  is logically invalid.

NPTEL

And for Logical Consequence we have a huge numbers of different possibilities. One of course is that one is that by definition you are basically looking at a world in which all the hypotheses are true. And then, you are asking whether in that world it is always guaranteed that the conclusion would also be true that is what the definition says.

(Refer Slide Time: 04:42)



The other possibilities is of course, to use one of these other theorem that we have got. One is to take all the hypotheses and take their conjunction and see whether conditional to the conclusion whether you what you get is a tautology. So, that is where for checking validly of arguments. You can use the version of a tautology at least proposition arguments it is not necessarily true for other kinds of argument. So, for propositional arguments you just asking whether this the conjunction of the hypotheses logically implies the conclusion also this being the tautology is equivalent to same like this logically implies the conclusion.

(Refer Slide Time: 05:26)

**Other Theorems**

**Theorem 4.3**

1. Let  $\Gamma = \{\phi_i \mid 1 \leq i \leq n\}$  be a finite set of propositions, and let  $\psi$  be any proposition. Then

(a)  $\Gamma \vdash \psi$  if and only if

$$\vdash \phi_1 \rightarrow (\phi_2 \rightarrow \dots (\phi_n \rightarrow \psi) \dots)$$

(b)  $\Gamma \vdash \psi$  if and only if  $((\dots ((\phi_1 \wedge \phi_2) \wedge \phi_3) \wedge \dots \wedge \phi_n) \wedge \neg \psi)$  is a contradiction.

2. A formula  $\phi$  is a tautology iff  $\neg \phi$  is a contradiction (unsatisfiable).

■

NPTEL

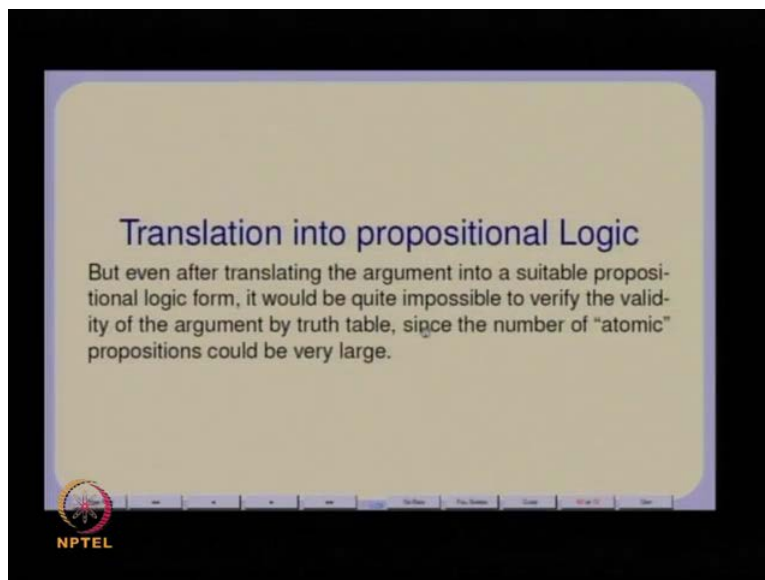
There is of course, other means for example this theorems tell us other way. One thing is to actually look at this whole thing conditionally just conditionally and check whether this is a valid formula. This is also question of checking whether this is tautology. And other possibilities is to actually take the negation of the conclusion along with the conjunction of the hypotheses and check whether you have a contradiction. I mean so these are essentially three different ways in which most checkers and a provers it works right. So, what we will do is we will take a previous one namely this and we will use this us in order to as a principle way of designing a tool.

So, this is using a essential theorem. So you have one possibilities will of course is to use truth table right one. I mean just construct a massive truth table ladder check for validity I mean check all the some. But, what happens is an but the other problem is not it is just enough to say that this argument is if it is valid of course it is enough to say whether it is valid. But, if it is not valid it is not enough to say that it is not valid. See if it is not valid that means you have to provide a scenario in which all the hypotheses are true and the conclusion is false. So, which means what we need to do is in order I have complete checker would actually give a falsifying assignment in case the argument is invalid. So, that is an important aspect an in fact this is an aspect that is normally not address that theorem provers proof here. But, there is another field of computer science which is getting to be very popular now and that is the notion of module checking. So, in that area of model checking when you have some logical statement if it is not true then, you have

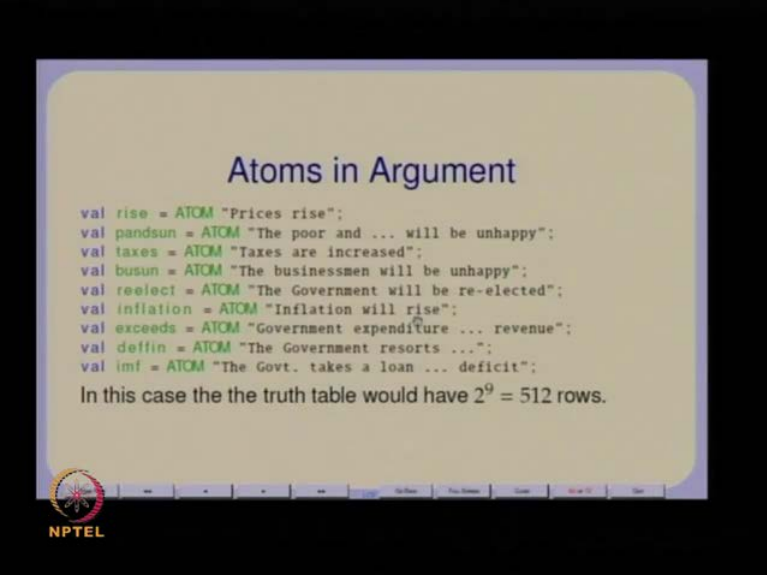
to provide a counter exam. In the case of propositional logic what we are asking is of falsifying assignments so it is the equivalent. So, what we are doing in our proctology checking is to provide the equivalent of a model checking mechanism.

So, if it is true it will definitely verify it. And but if it is false then it has to provide a falsifying it has to provide a counter example or a falsifying assignment or some other proof that it is indeed false and that proof is something that it should be able to check. So, in this particular case we are looking for a essential or falsifying assignment which will make the argument in which will show that the really the argument is invalid right. So, one thing of course you said that you have to as I said clean up the argument and translate it into an argument involving only the propositional connectors. Which, also means you have to identify what are known as the atoms in the arguments. What are the atomic proposition which can be taken for granted which can be which do not have to be split any more. So, you have to split sentences into terms of the propositional connectives like and or if than or not and you have to actually identify atomic statements whose truth value could be anything so and since you are looking for validity.

(Refer Slide Time: 09:34)




(Refer Slide Time: 09:36)



**Atoms in Argument**

```
val rise = ATOM "Prices rise";
val pandsun = ATOM "The poor and ... will be unhappy";
val taxes = ATOM "Taxes are increased";
val busun = ATOM "The businessmen will be unhappy";
val reelect = ATOM "The Government will be re-elected";
val inflation = ATOM "Inflation will rise";
val exceeds = ATOM "Government expenditure ... revenue";
val deffin = ATOM "The Government resorts ...";
val imf = ATOM "The Govt. takes a loan ... deficit";
```

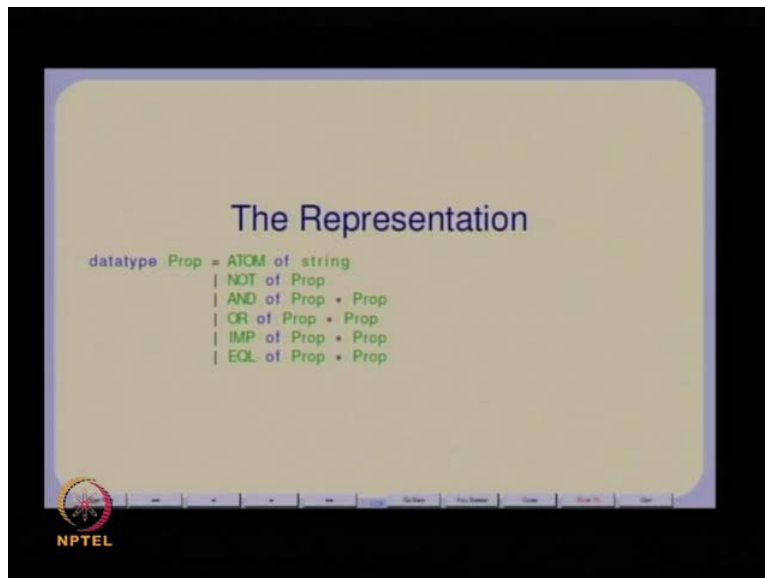
In this case the the truth table would have  $2^9 = 512$  rows.

  
NPTEL

So, one thing I of course that part of that cleaning up is to essentially identify the atoms in the argument. So, here is a simple identification. So, we have got so many atoms which are which are so I will use essentially strings taken from the argument as atomic statement. Which, cannot be split further into sub statement in fact these are the atoms in the arguments identified in some way and we already see that there are nine atoms. So, your truth table is going to be quite ignominious it is going to have 512 rows. And of course, dip I do not know since there are compound propositions. There going to be that many columns his so is this gone be a huge truth table consisting of about few thousands cells. Now, the point is not actually necessary to construct the entire truth table. That is that one thing. So therefore, of course non other aurgatharance that we are going to design is anyway going to be less than exponential. That is it is going to be a bottom line but, that is not what we are saying. But, in practice what we are saying is so a firstly it is not necessary to consider construct the entire truth table.

So, it is only necessary to consider possible falsifiers and if there are no falsifier essentially the argument is valid. So it is possible to due tautology checking in some more discipline fashion and that is what we have to look at fact.

(Refer Slide Time: 11:32)

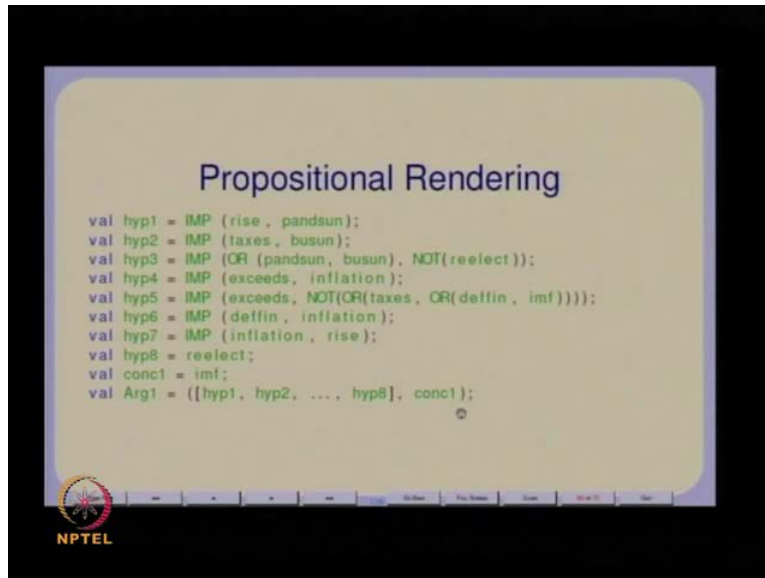


And as for this basic Representation is concern. We represent proposition as abstract and tax trees using standard data type or recursive data type definition. Which, is essentially a reflection of the grammar which was used for the design of abstract and tax trees. So, as there is so atom of string and after that rest are all connectives. Which, with obvious meanings that we can attribute to them NOT as vacation AND is and an OR is or impish conditional and EQL really foundering as we came to a bar of something fact. So, this is our we can this is as a standard representation this way what I am going to what I do not need to worry about initial aspects of scanning and parsing whatever the argument. We just have representation in terms of this data type and that is directly gives you abstract and tax trees correct.

So, essentially the proposition will be form from these atoms.



(Refer Slide Time: 13:01)



And so if Propositional Rendering without going into the other kind aspirates of the translation essentially gives you sequence of hypotheses like this. So, there are eight hypotheses and these are the compound propositions. So, if you look at sub proposition sub formally that these eight a hypotheses have the conclusion seems to be it is just an atomic one. But, if you look at all the sub formula that you have any truth table construction will require columns for each of the sub formularies. And so that is how you build up truth values of the full formula. So if, I look at all those sub formula then essentially what you have is something like we had nine atoms may be something like twelve or thirteen different kinds of sub formula.

Because, this one this hypotheses five for example has four sub formula. At least because there are four operated there this thing has hypotheses three has three sub formulas at least. And so any truth table construction would actually have your 512 rows for this. And multiply by 13 or 14 columns with a before and then after that you will require some or problems. Because, what you going to do is you will have to take their conjunction of all these hypotheses and the conclusion. So, that might be 15 or 16 columns. And then essentially find the truth value of the argument that can be quite huge. So, it be a few thousand cells we do not want to do that we want to do more so to speak logical detective way of doing tautology checking all right.

(Refer Slide Time: 15:09)

The Strategy

We need to either show that

$$Arg1 = IMP(bigANDH, conc1)$$

is a tautology or can be falsified. Using theorem 4.2 for validity

```
val bigAND = leftReduce (AND);
fun Valid ((H, P):Argument) =
  if null (H) then tautology (P)
  else tautology (IMP (bigAND (H), P))
```

and for falsification we use

```
fun falsifyArg ((H, P): Argument) =
  if null (H) then falsify (cnf(P))
  else falsify (cnf (IMP (bigAND (H), P)))
```

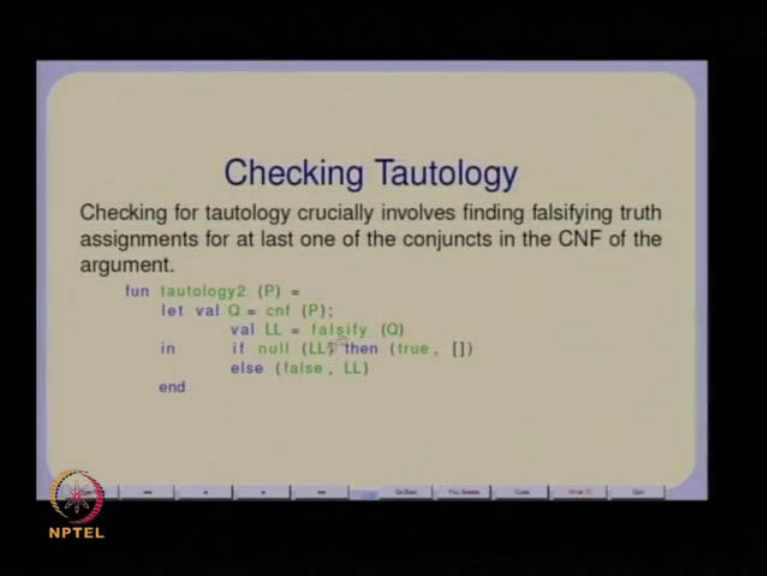
NPTEL

So, essentially we need to show that this one was not come out right this so an argument as far as I am concerned a list of hypotheses is a list of hypotheses and a conclusion here. So, it is an ordered pair of list of hypotheses and a conclusion where both the list is a list of proposition and the conclusion is also a proposition. So, essentially what we are going to do is we going to do an argument is a big and the all the hypotheses where edge is a the set of a list of hypotheses and a essentially we are saying that you take the conjunction of the hypotheses and show that the implication show that this conjunction implies the conclusion.

So, I can so there is a so I can define function called which does it big AND what we are going to do is. So, you take this so essentially there are of course trowel cases where you might just have a conclusion. I mean that is a trowels case of just claiming that some proposition is at tautology. So, this is what so I would and an argument is valid provided you take this implication and prove that it is a tautology. Of course, the result of checking whether its tautology is just going to be a true or false result and that is not sufficient. If, it is false then we also requires previous falsification so here is the function to falsify an argument. Which, essentially checks whether the conjunctal normal form of this entire proposition big and of H implies P whether that conjunct to normal form can be falsify. So, this is one possible strategy and not necessarily the best strategy this is one possible strategy here and this is the strategy that I have adopted.

If, there is a problem here the big and gives me conjunctive normal forms very easily. But, the fact that the big AND has to implies something essentially means when, I replace the implicate the arrow or then I will get a negation of the big AND. Which, will be a big OR and then if I have to take the conjunctal normal form of that essentially that formula will explore it. Because, I will have to distribute the or I had to push it down in order to get the conjunctly normal form. But, any way we going to do that. So, actual Checking of Tautology therefore reduces to I can get rid of that tautology function. And just look at falsification because, any way I have to do falsification if there is going to be any proof.

(Refer Slide Time: 18:32)



The slide is titled "Checking Tautology" and contains the following text and code:

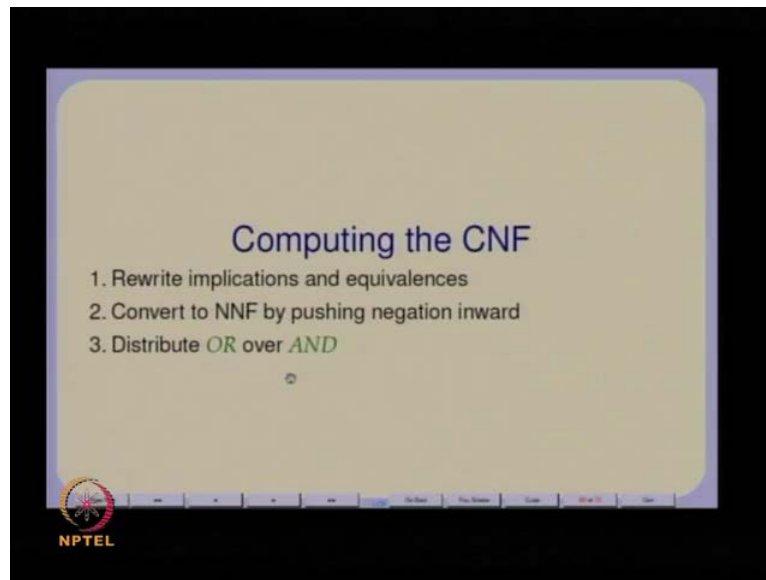
Checking for tautology crucially involves finding falsifying truth assignments for at least one of the conjuncts in the CNF of the argument.

```
fun tautology2 (P) =  
  let val Q = cnf (P);  
      val LL = falsify (Q)  
  in  
    if null (LL) then (true, [])  
    else (false, LL)  
  end
```

The slide also features the NPTEL logo in the bottom left corner and a navigation bar at the bottom.

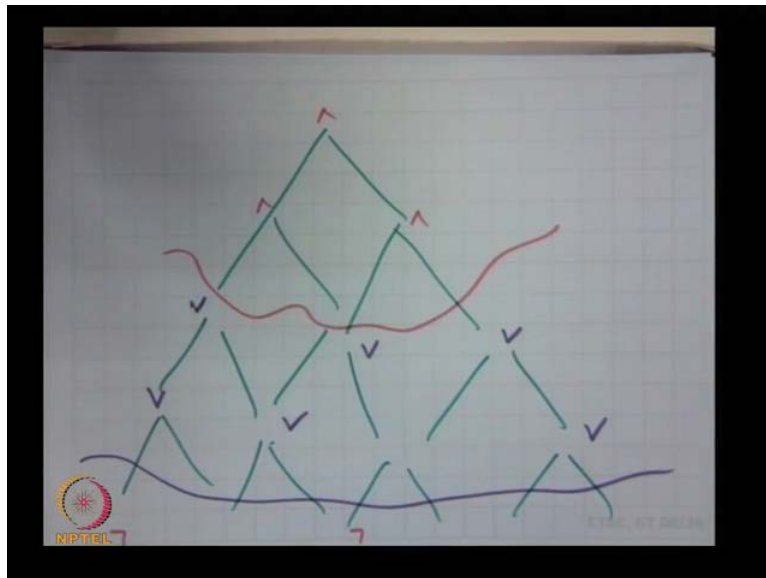
So, what do I do. I take entire argument represented as a proposition and I take a conjunctal normal form and check whether I can falsify it. And if, I can falsified then it is that itself gives me of falsifying assignment. An assignment which will give me a truth assignment for the individual atoms which ensures from which I know that argument is invalid and that can be checked. For example manually if you like if, you have enough time. So, if the result of falsification is an empty list that means there is no possible falsification therefore its tautology. So, this is the way which proctology checks all words. So, falsification is an important aspect is not just checking validity of architecture and I use the falsification in order to determine various some tautologies in fact.

(Refer Slide Time: 19:48)



So, one thing of course Computing this CNF taking any propositional logic taking any element of that data type of opposition basically we have to defined rewrite rules. One is of course pushing down negation and when you push down negation then yours or's an ands get inverted. But, before that what you need to do is to Rewrite all the implication an equivalences rewrite implication in equivalences in terms of and or's and not's and after that you just push down or all the negation so that you get at the bottom of the tree only literals. And there is no negation after worse about so, there is a front gear of negation which gets created and above it since there are no other operated they can only be ands and or's. And of course, what you can do is once you distributes all though or's over the ands you get a stratified formula something like what I said. We would get a essentially satisfied formula like this.

(Refer Slide Time: 21:04)



And what the big AND does it actually flatance this street to a list of literance. So, this big AND essentially just flatance this tree into a list of literance. So now, I am essentially dealing with lists of literance. So, the list of literance means basically that you have a list of conjuncts and each conjuncts is a list of is a disjunction of literance. And so you have a list of literance and you have to look at those literance in that way. So, this is so when you do all this so you will get a essentially this list. So, how do you do falsification? So, essentially what we are so the whole idea now of the tautology checker has moved from actually determining whether it is a formulized tautology to just checking whether it can be falsified. So, falsification is important.

(Refer Slide Time: 22:28)

**Falsifying CNF**

1. Suffices to find a falsification of at least one conjunct
2. A conjunct in the CNF can be false iff all the disjuncts in it are false.
3. A disjunct is false iff it does not contain a "complementary pair".

Assume the CNF is  $Q \equiv \bigwedge_{i=1}^m D_i$  where each  $D_i \equiv \bigvee_{j=1}^{n_i} L_{ij}$  where the literals of  $D_i = P_i \cup N_i$  where  $P_i$  is the set of positive literals (atoms) and  $N_i$  consists of the atoms appearing as negative literals.

Then  $D_i$  is false iff  $P_i \cap N_i = \emptyset$ .

NPTEL

And basically if it cannot be falsified then it is tautology. So, one thing is true so you have a CNF and in order to falsify a CNF it suffices to find just one conjunct in this CNF which is false. So, it seems it is a conjunction of essentially disjunctions its posses it is necessary to find just one of those to be false. Now, when is a disjunct false that is where the complication comes at disjunct it false if an only if it does not have what is non as complementary pair. So, if it contains a complementary pair that means, if it contain both the positive for some atom P it contains both P and not P. Then, the disjunction of P and not P is any way true so that disjunct never be false. However, if there disjunct consist of just completely disjoint literence with no common atoms for all the atoms in all the literals are unique.

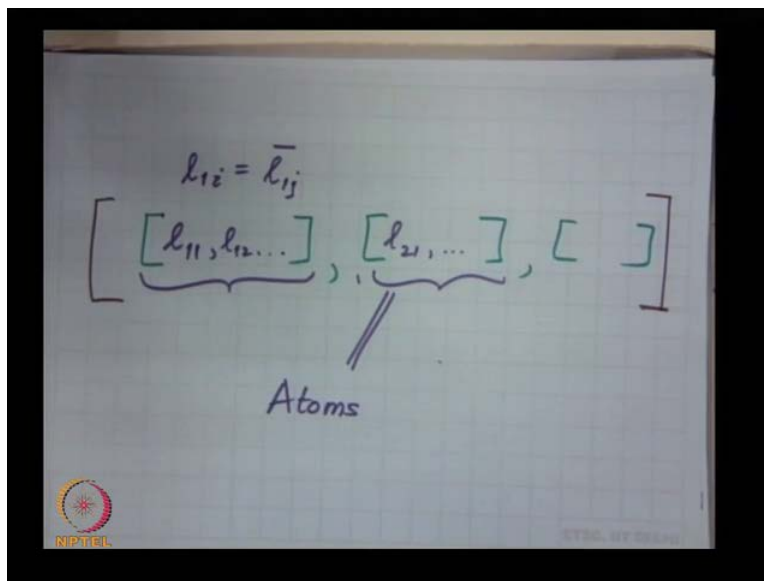
Then, it is possible to falsified by setting all those literals false which means the positive atoms you said them to false and the negative atoms you said them to true. So, then you can this of course the only way disjunction can be falsified is all the literance in the disjunction are false. And the way to do that is to ensure that so if so now, the problem therefore just reduce its so then it can be falsified. But, we do not need to actually do that it is just enough to check whether there is a complementary pair or not. If, there is not complementary pair then it can be falsified and therefore there is no problem. So, this is what so you take this literals in any disjunct. So, let say take a disjunct  $D_i$  and just patrician it into the positive and the negative literance. So, in fact once you have patrician it you can even remove the negation sign from the literance so you patrician it

into two set of atoms. One set of atoms is  $P_i$  and the other set of atoms is  $N_i$  and all you check is whether they have is any common element in it. So, what I am saying is it so the so this is what actually tautology checkers does. So now, the falsification of the CNF reduces essentially to just checking whether there are common elements into list.

Student: sir essentially this you are taking for common elements so how are you falsifying all the atoms into that.

Then, basically the way to falsify all the atoms is to as I said assign to all the positive atoms the value false and all the negative the atoms which are occur in negative form assign them true. So, you get a falsifying assignment which will make at least one conjunct false. And that is enough it makes a disjunct false but, then you have a conjunction of disjunction so one of the disjunction has to be made false that is right. So, essentially what we are saying is. So if, I have a list of list. If, I have some list of list of this form.

(Refer Slide Time: 26:18)



So, this is essentially list of this kind which contain literals. And this brown one is essentially an and of all these lists each of these green ones is an or, of all the iteration inside you might have various literals  $l_{11}$   $l_{12}$  and so on and so forth  $l_{21}$  and so on and so forth. And these literals I am partitioning each of these to check so in order to check whether these whole thing can be falsified I need to just go through this list and check whether any one of them can be falsified.

So, if I find let us say this one if I find this cannot be falsified the only way it cannot be falsified is because. There are two literals  $l_i$  and  $l_j$  which are actually complimented repairs. That is the only way this cannot be falsify cannot be falsified mean as it is always true.

So, that is what happens so I go through these and if, I find these the atoms of this are all distinct then I know that it can be falsified. And how can I falsified I can falsified provided all the lit each literal is assign the value 0 is assigned truth value 0. And in order to assign each literal the truth value is 0 I take the positive literals there are just atoms and I assigned them the truth values is 0. I take the negative literals I take the atoms and assigned them one. And therefore, I have got a part assignment of subset of the atom in the entire proposition which will guaranty the falsification of one of the disjunct.

And basically for all other atoms I do not care what assignment is done. Whatever other assignment may be done or truth value this disjunct is still guaranteed to be false. And therefore that is a falsifying assignment. So, that is a exactly what happens in this case.

Student: is this concept is here the only problem I cannot understand that when you specifying a letters do not be her tarry assume that it is true.

No why should be literal.

Student: than the government will be reelected thisdo not you (Refer Time: 29:18).

No what we are all that we are saying is that that happens to be an atom right. But, what I am saying is look at this complicated thing an which has a lot of or's in it. Now, even it is this complicated there is a way of you take this statement. I can zoom this way comes down to this it has so many or's in it. That even if I assumes that this entire proposition is true it is not there is I can it is quite possible. That is some this proposition can be made true only by sorting this by assigning certain atoms of false value.

Supposing, there is a negation there. I mean so it is there is no guaranty that you have to assigned all the atoms value true no right some of the in fact there are various assignments in which this can be true some of the atoms could be true. In fact this contains two or's basically any truth assignments in which at least one of the atoms is assigned true is would make this true right. I meant so that is not really the important thing therefore what were saying about validities is just



that in the set of all truth assignments in which all these hypotheses are true is the conclusion also true that is all we are asking. How all these hypotheses can be true can be because of certain atoms being assigned as false value of false truth value.

So, let us let me just so we are going to use this tautology two which essentially reliance on falsification. And an this is the strategy that the tautology checker actually implies. So, if you read the source code of the checker actually you will find this and also there is a partitioning of the list of literals into two sets of atoms. So, the  $P_i$  and the  $N_i$  are actually atoms and basically there is a one hires out of function we checks for disjointness right. So, this is how this tautology check over.

Student: (Refer Time: 32:26)

I actually we do not polynomial inbound you have to look at the measures. So, once I have converted things into CNF in terms of this CNF it is polynomial type. But, the main problem is converting things into CNF that can double sizes. So, if I had a especially look at this there implication of a big AND. So huge number of you can have a huge number of hypotheses and you take this implication which means first you rewrite this implication by a negating the big AND. And negation of the big AND makes it a big OR. Then, you want a CNF which means you have to distribute or and then there you are exploding the number of terms the length of the formulas. That is where the problem is so the conversion into CNF can actually explored the formula sets. But, otherwise what we are saying is write them the just given so if you look at the complexity of this method just in terms of given a CNF. And the length of a CNF then, its polynomial at the length of a CNF. But, if you look at the size of the original argument when it could be exponential in the size of the original argument.

So, it is important actually take to know what is your measure this I will looking at the size of the CNF as  $N$  and determining the complexity in terms of that  $N$ . Or I looking at the size of the original argument as  $N$  and then determining this. An also there is a section of sometimes it thinks are different I mean between the number of atom there might be a you might have different complexities based on the number of atoms. And the number of connectives right I mean you might have to separate the two also because your explosion and the number of atoms is fixed initially. The explosion happens because of the connectives. So, its exponential any way

if I look at it as in terms of the size of the original argument. The only thing I can do is as try to restrict it for example, the actual tautology check of code if you look at this it. It does not rate stop with finding one falsifying assignment it actually finds all falsifying assignments. And the way the falsifying assignment are actually found is by just listing each is just by listing all the positive atoms an assuming that all other atoms are negative after I mean look at this strategy.

So, what you are saying here is I split this list into a collection of positive atoms and the collection of negative atom and check whether there is disjoint. And supposing, there are not disjoint supposing, there are disjoint. Then, I actually do an assignment I can actually do an assignment but, then there might be an atoms in these things which, are not mentioned in this list. So, what do I specifying the list just this atoms which have to be set true assuming that all other atoms that occur in the proposition are set to false. So, actually I do not even find all I do find many other I saw I go through all the list in order to find various falsifying assignments but not necessarily all of them. So, that is why it is important to check the emptiness of this list where is that it is important to check the emptiness of this is the list of list. So, what it actually gives me is it gives me a bullion value false and a list of lists where each list within this list is essentially specify which of the atoms have been assigned true that is it. So, it does not check all the falsifying assignments necessarily but it does check all the conjuncts. And gives you what at and essentially this ll tells you each list inside this ll tells you which of the atoms have to be assigned true and leads all the other assumes that all the other atoms are assigned false or lives them unspecified.

But, it is necessary to actually set them all false because otherwise you may not make the make this disjunct false right. So, there are there those certain things happening inside the tautology checker but, I do not take necessary find all possible for falsification. I just find some falsification but I go through all the conjunction. So, it basically I am finding some falsification for each conjunct is possible wherever possible that is the list of list of truth assignment that I give as this ll. And otherwise and if that list is empty basically it means that I will not be able to I found a complementary pair that list is empty basically it means that found a complementary pair. So, that conjunct so that list of so each conjunct there was at least one complement repair which, means each of these regarded as this junctions is true and therefore this conjunction is true and therefore this conjunction is tautology.

Student: (Refer Time: 39:07)

So, if you think of CNF your and or above the associate in commutative so its unique in the same way that numbers have a unique prime factorization up to order. Since, we are considering list as a data structure actually there is an order. But, order but further the I imposable order on the atoms I impose an artificial lexico graphic order on the atom because atoms are all strings. So, I impose an artificial lexico graphic ordering therefore actually I get list which are unique. Because, there order according to this lexico graphic ordering there is a salting also which happens in order to get a unique form. But, those are internal details we primarily regard to view need to view it us sets of sets of literance.

A set of literance and they have a unique forms simply because of a artificial lexico graphic ordering on the strings that I impose. There are other possible ways one possible way is to just store all the atoms in the order in which they appear use the indexes as your sorting order. For example, and use that throughout I mean that might actually simplify the I mean simplify carrying around strings. I mean carrying around strings is not particularly expensive but it is easier to carry a indexes interior indexes than indexes than carry strings for example that is one thing. But I did not do that I just use the string ensures I used to lexico graphic ordering and I sorted. But, the best way to check disjointness I required sorting because the best way to check disjointness was to actually look at these two sorted list  $P_i$  and  $N_i$ .

And just scan them once linearly to check this jointness that's it. So, I do imposes sorting without you an I do a sorting without duplicates because, anyway here conjunction into disjunction are idempotent so having duplicate case around this useless it just occupied by space. So, there is there are these cleaning up and book keeping operation at you need to do. So, basically do as sorting without duplicate while you are creating CNF if itself and so that your list of liter's are already in a certain order. Also I think my sorting I had to write a sorting algorithm because I also consider all the positive literal as being less than all the negative liter's right.

You have to have a total ordering on literals so which means you have to do something about either I did that or I put the I interferences the interleave the positive letters and the negative literals that is another total order. They can always you can claim that  $P$  is less than not  $P$  for example. And  $T$  is less than  $Q$  and not  $P$  is also less than  $Q$ . So, you can impose various kinds of total

ordering whatever is convenient for the algorithm to check this jointness. But, the crucial things this jointness what is the easiest simplest way in which can do checking this jointness without doing very complicated algorithm is like the once you learn in algorithm course on algorithm. Because, sorting is always a convenient thing for other purposes. I mean which just like a when you do on coursing on algorithm you have to find in the k media or some certain thing. But, you not allowed to sort it but supposing I sort it then I know k th largest element immediately but that has a complexity which is the at least the complexity of the sort algorithm. Where, as finding the k th percentile whatever can be done in less than the complexity of asortic. But, assorting has other uses which over which do not allow me to which I can use. So, for example just the fact that I want a unique normal form absorbing helps me to do that.

And then, I checking disjointness lineal time you can that so when you start going down deeper and deeper into the data structures is not enough to just look for the most efficient algorithm which will do certain job. But, whether the most convenient form can be used for other things so one thing is because of this lexico graphic ordering there is a unique representation. Because, I do sorting I can use that unique representation without and sorting without duplicate I sort of this reduce the sizes of I remove unnecessary redundant stuff and it also helps me to do disjoints in checking its linear time and so on. Those are decision you take in a typical course on analysis of l algorithm in use you do not look at the problem in contest. Here, you have to look at the problem in contest is it really worth doing less than  $N \log N$  is it really worth not sorting them. Data structure is a question that you have to ask there is if sorting radiator structure gives you some other benefits. It is better to do the sorting and then use that sorting itself.

So, then let us go back briefly. So, the other kinds of technique that are normally used are these two. So, one of this many of you have must actually used this one. This b part of this theorem you use it essentially in resolution theorem so systems like prolog use resolution theorem proving. And a this is actually much more common because what it does if you already seen if you already know how the prologs works is that it allows you to do directed computation leading to a contradiction. Where is a proving things otherwise can take you of the main direction so, the whole idea of doing a directed sub well compotation and prolog is to eventually reach that contradiction in some directed fashion. And so this is very useful for that we will also see the another method call the tab tablo method which, originally invented by famous by a magician

come mathematician called Raymond's Malian. But, now tablo methods are extremely popular in the thermo proving error because, they also use this theorem 4.3 b and tries to a give you a contradiction but, in a certain sense they there are very fast must faster than resolution. So, it is not clear to me why for example but resolution also comes with other things make unifications and so on and therefore value bindings prolog program but more and more people are using tablo methods to prove things if you just looking at proof things provability than and tabloo methods are very good. So, what I will do next is actually a tabloo method I start with a tabloo method and this tabloo methods are also very useful for all other kinds of logics. Because, they first they have same advantages resolution in providing in you a directed means and secondly they some or seem in practice in to work faster than resolution.

So I will essentially stop here. Are they any other questions.

Student: (Refer Time: 48:00)

There are I separated the two terms of adequacy and function completeness many book treat them as the same. So, what I did was I used the existing I used what I thought was set of operator's standard set of operators and I defined adequacy in terms of that and function completeness I defined as a level above. There is most book treat them books both has the same concept and they just ask the question whether you got a set functionally set of operators. So, one thing is clear that any adding set of operator is also functionally complete.

Student: (Refer Time: 48:57)

That is because omega is functionally complete but I separated the two of this because I thought we start with some standard and work downwards and up wards from this standard that is all any other questions. I think in some of my calls home pages you will find this tautology checker code you can see it written in ML. And it essentially you can see there are also common that code you can see how actually the development takes place.