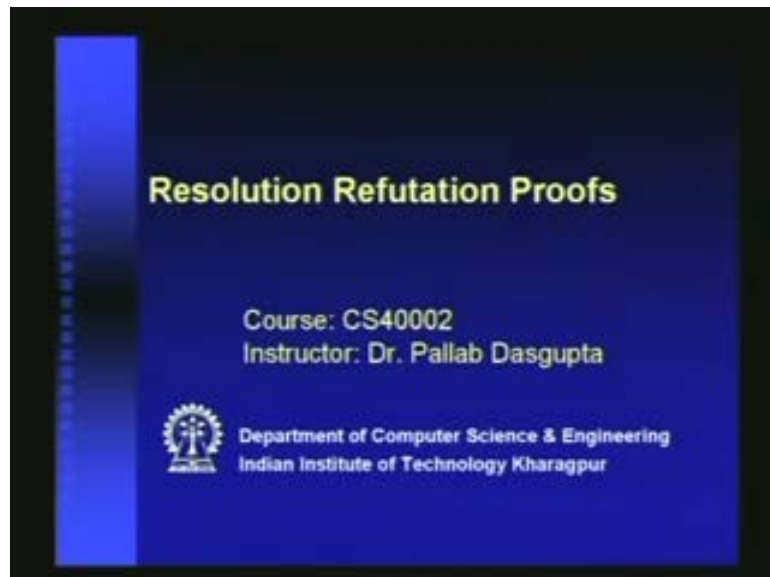


**Artificial Intelligence**  
**Prof. P. Dasgupta**  
**Department of Computer Science & Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture- 12**  
**Resolution Refutation Proofs**

We will continue with our analysis of first order logic. Today, we will study a kind of proof mechanism called resolution refutation proof.

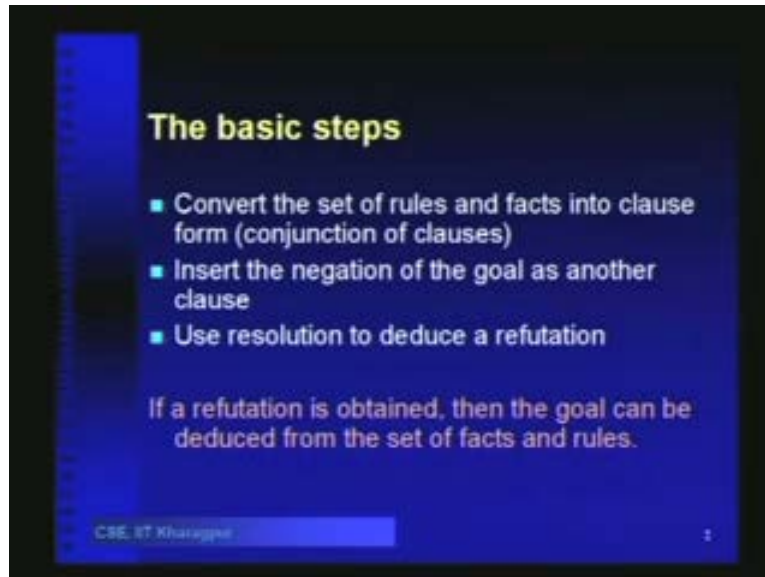
(Refer Slide Time: 00:01:02)



Today's lecture will be on resolution refutation proofs. The basic idea of resolution refutation is as follows: we will first convert the set of rules and facts into clause form. Recall that in the last class, we had discussed that a clause form is a conjunction of clauses, and each of these clauses can be a disjunction of predicates or their negations.

That is the clause form. In the clause form, we have only the universal quantifiers and all of those universal quantifiers are pushed to the left.

(Refer Slide Time: 00:04:11)

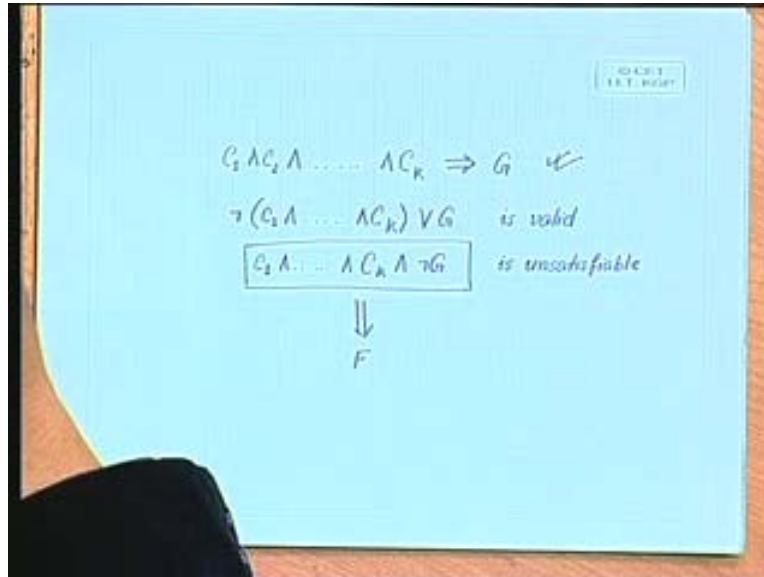


So, any variable that we have in the clause form is universally quantified from the outside. For the existential quantifier, recall that we use the thing called Skolemization, to replace the existentially quantified variable by skolem functions. After converting the set of rules and facts into clause form, we insert the negation of the goal as another clause and we try use resolution to deduce a refutation. Let us see- I think we had discussed this in the context of propositional logic. The idea is that if I have a set of clauses-  $c_1$  and  $c_2$  and some  $c_k$ - suppose this is the set of  $k$  clauses that are given to us, and we want to deduce whether this implies the goal.

Then, if this has to be valid, **then the neg** then, it is equivalent to writing not of this  $c_1$  and  $c_k$  or  $g$  is valid. Or the negation of this, which means  $c_1$ ,  $c_k$  and not of  $g$ , is if this is valid, then, the negation of this is unsatisfiable. If we can deduce that this thing is conflicting, in the sense that from this, if we are able to deduce false, that means that this

set of clauses will be inconsistent and therefore unsatisfiable, and therefore, this original thing that we wanted to prove will be proved.

(Refer Slide Time: 00:04:05)



So, if we use resolution to deduce refutation, then if a refutation is obtained, then the goal can be deduced from the set of facts and rules. But recall, that because of the semi decidability of first order logic, it may also be possible that we continue forever, and we are not able to deduce false and that resolution continues forever. And that will happen for the kind of formulas that I had shown you previously, **that** which has **functors** functions within the predicates, and there is a recursion in the function which never terminates. In the last class, we had studied the conversion to normal form, where we had seen a set of steps that converts any given formula to normal form.

So, like taking the existential closure and eliminating redundant quantifiers, then renaming any variable that is quantified more than once. Eliminating implication, then moving negation all the way inwards to the predicates, pushing the quantifiers to the right, then eliminating existential quantifiers, as I was mentioning, through a skolem function, which is also called Skolemization, followed by moving all universal

quantifiers to the left, and finally distributing the and over or, to get the conjunctive normal form.

(Refer Slide Time: 00:04:51)

**Conversion to Normal Form**

- A formula is said to be in clause form if it is of the form:  
$$\forall x_1 \forall x_2 \dots \forall x_n [C_1 \wedge C_2 \wedge \dots \wedge C_k]$$
- All first-order logic formulas can be converted to clause form
- We shall demonstrate the conversion on the formula:  
$$\forall x \{ p(x) \Rightarrow \exists z \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,y))] \wedge \forall y [q(x,y) \Rightarrow p(x)] \} \}$$

CSE, IIT Kharagpur

(Refer Slide Time: 00:04:57)

**Conversion to Normal Form**

- *Step1: Take the existential closure and eliminate redundant quantifiers. This introduces  $\exists x_1$  and eliminates  $\exists z$ , so:*  
$$\forall x \{ p(x) \Rightarrow \exists z \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,y))] \wedge \forall y [q(x,y) \Rightarrow p(x)] \} \}$$
  
$$\exists x_1 \forall x \{ p(x) \Rightarrow \{ \neg \forall y [q(x,y) \Rightarrow p(f(x,y))] \wedge \forall y [q(x,y) \Rightarrow p(x)] \} \}$$

CSE, IIT Kharagpur

(Refer Slide Time: 00:05:07)

**Conversion to Normal Form**

- Step 2: *Rename any variable that is quantified more than once. y has been quantified twice, so:*

$$\exists x, \forall x \{ p(x) \Rightarrow [ \neg \forall y [ q(x,y) \Rightarrow p(f(x,)) ] \wedge \forall y [ q(x,y) \Rightarrow p(x) ] \}$$
$$\exists x, \forall x \{ p(x) \Rightarrow [ \neg \forall y [ q(x,y) \Rightarrow p(f(x,)) ] \wedge \forall z [ q(x,z) \Rightarrow p(x) ] \}$$

CSE, IIT Khargpur 1

(Refer Slide Time: 00:05:11)

**Conversion to Normal Form**

- Step 3: *Eliminate implication.*

$$\exists x, \forall x \{ p(x) \Rightarrow [ \neg \forall y [ q(x,y) \Rightarrow p(f(x,)) ] \wedge \forall z [ q(x,z) \Rightarrow p(x) ] \}$$
$$\exists x, \forall x \{ \neg p(x) \vee [ \neg \forall y [ \neg q(x,y) \vee p(f(x,)) ] \wedge \forall z [ \neg q(x,z) \vee p(x) ] \}$$

CSE, IIT Khargpur 2

And finally, an optional step to simplify, so that in the end we get a set of clauses. Here, for example, this not px or qx gx is 1 clause, and not of pfa is another clause. Let us take

an example. Then, we had resolution, which in terms of the- I have rewritten the resolution rule.

(Refer Slide Time: 00:05:13)

**Conversion to Normal Form**

- Step 4: Move  $\neg$  all the way inwards.

$$\exists x_1 \forall x \{ \neg p(x) \vee \{ \neg \forall y [ \neg q(x,y) \vee p(f(x,y)) ] \wedge \forall z [ \neg q(x,z) \vee p(x) ] \} \}$$
$$\exists x_1 \forall x \{ \neg p(x) \vee [ \exists y [ q(x,y) \wedge \neg p(f(x,y)) ] \wedge \forall z [ \neg q(x,z) \vee p(x) ] \} \}$$

CSE, IIT Kharagpur

(Refer Slide Time: 00:05:17)

**Conversion to Normal Form**

- Step 5: Push the quantifiers to the right.

$$\exists x_1 \forall x \{ \neg p(x) \vee [ \exists y [ q(x,y) \wedge \neg p(f(x,y)) ] \wedge \forall z [ \neg q(x,z) \vee p(x) ] \} \}$$
$$\exists x_1 \forall x \{ \neg p(x) \vee [ [ \exists y q(x,y) \wedge \neg p(f(x,y)) ] \wedge [ \forall z \neg q(x,z) \vee p(x) ] \} \}$$

CSE, IIT Kharagpur

We had seen the resolution rule in the implicational form in the last class. I have just rewritten it, so that we can associate resolution rule with the conjunctive normal form that we are looking at. Suppose, this is 1 clause which is a disjunction of some of these predicates, and this is another clause, which is another disjunction of predicates.

(Refer Slide Time: 00:05:20)

**Conversion to Normal Form**

- Step 6: *Eliminate existential quantifiers (Skolemization).*
  - ◆ Pick out the leftmost  $\exists y B(y)$  and replace it by  $B(f(x_1, x_2, \dots, x_n))$ , where:
    - a)  $x_1, x_2, \dots, x_n$  are all the distinct free variables of  $\exists y B(y)$  that are universally quantified to the left of  $\exists y B(y)$ , and
    - b)  $f$  is any  $n$ -ary function constant which does not occur already

CSE, IIT Kharagpur

(Refer Slide Time: 00:05:27)

**Conversion to Normal Form**

- Skolemization:

$$\exists x_1 \forall x \{ \neg p(x) \vee [ [\exists y q(x,y) \wedge \neg p(f(x_1))] \wedge [\forall z \neg q(x,z) \vee p(x)] ] \}$$

$$\forall x \{ \neg p(x) \vee [ [q(x,g(x)) \wedge \neg p(f(a))] \wedge [\forall z \neg q(x,z) \vee p(x)] ] \}$$

CSE, IIT Kharagpur

And then, theta is a substitution which unifies  $z_j$  and not of  $q_k$ , which means that if we use the substitution theta, then  $z_j$  here and not of  $q_k$  here, become identical. If that happens, then the resolution tells us that we can deduce the or of  $z_1$  through  $z_i$  minus 1,  $z_i$  plus 1 through  $z_m$ , and likewise,  $q_1$  through  $q_k$  or  $q_k$  plus 1 through  $q_n$ .

(Refer Slide Time: 00:05:30)

**Conversion to Normal Form**

- Step 7: Move all universal quantifiers to the left

$$\forall x \{ \neg p(x) \vee [ [q(x, g(x)) \wedge \neg p(f(a))] \wedge [ \forall z \neg q(x, z) \vee p(x) ] ] \}$$
$$\forall x \forall z \{ \neg p(x) \vee [ [q(x, g(x)) \wedge \neg p(f(a))] \wedge [ \neg q(x, z) \vee p(x) ] ] \}$$

CSE, VT Kharagpur 11



(Refer Slide Time: 00:05:35)

**Conversion to Normal Form**

- Step 8: *Distribute*  $\wedge$  over  $\vee$ .

$$\forall x \forall z \{ [-p(x) \vee q(x, g(x))] \wedge [-p(x) \vee \neg p(f(a))] \wedge [-p(x) \vee \neg q(x, z) \vee p(x)] \}$$

- Step 9: (Optional) *Simplify*

$$\forall x \{ [-p(x) \vee q(x, g(x))] \wedge \neg p(f(a)) \}$$

CSE, IIT Khargpur 13

Now, that is that is easy to see, because if you have something here- if you have some a here and not of a here, then we know, that if you merge the 2, if you take and of this clause and this clause, you can get this clause, by eliminating that, right? This is the same resolution rule, but only shown in the clause form. So, what we are going to do is, we will take 1 full example and work out a complete resolution refutation proof, and then, we will study some of the optimizations that we can do in the proof procedure.

(Refer Slide Time: 00:06:00)

**Resolution**

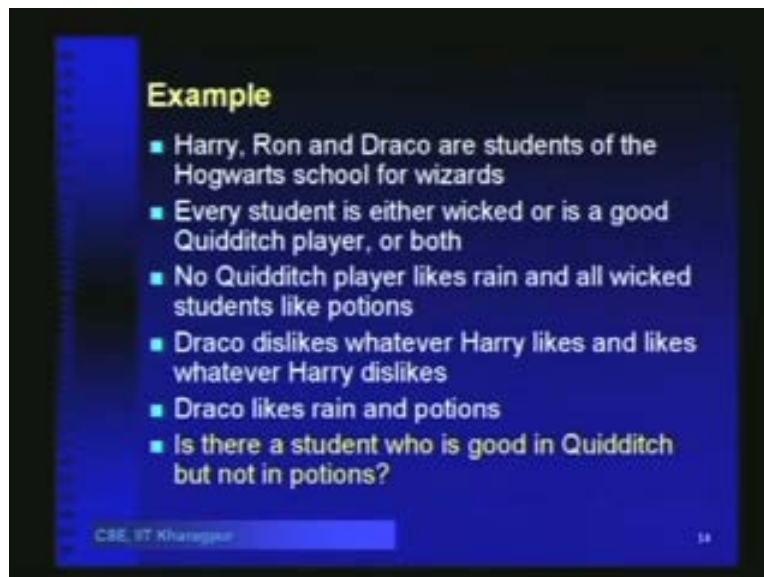
■ If  $\text{Unify}(z_j, \neg q_k) = \theta$ , then:

$$\frac{z_1 \vee \dots \vee z_m \quad q_1 \vee \dots \vee q_n}{\text{SUBST}(\theta, z_1 \vee \dots \vee z_{j-1} \vee z_{j+1} \vee \dots \vee z_m \vee q_1 \vee \dots \vee q_{k-1} \vee q_{k+1} \vee \dots \vee q_n)}$$

CSE, IIT Khargpur 18

See, essentially, the proof procedure is a search procedure, because we are going to have a set of clauses, then, you can take a pairs of clauses, use resolution to deduce new clauses. Now, which pairs will you take? There are there are some pairs existing in the knowledge base, some which you have deduced. If you continue using them in the haphazard way, then we will have to do a lot of work. So, the search is essentially to choose those pairs, on which we will apply resolution and eventually be able to deduce false. And, we will see some heuristics, which will help us in choosing the clauses, so that we are able to deduce false more quickly. This example is inspired from some very well-known stories.

(Refer Slide Time: 00:09:22)



**Example**

- Harry, Ron and Draco are students of the Hogwarts school for wizards
- Every student is either wicked or is a good Quidditch player, or both
- No Quidditch player likes rain and all wicked students like potions
- Draco dislikes whatever Harry likes and likes whatever Harry dislikes
- Draco likes rain and potions
- Is there a student who is good in Quidditch but not in potions?

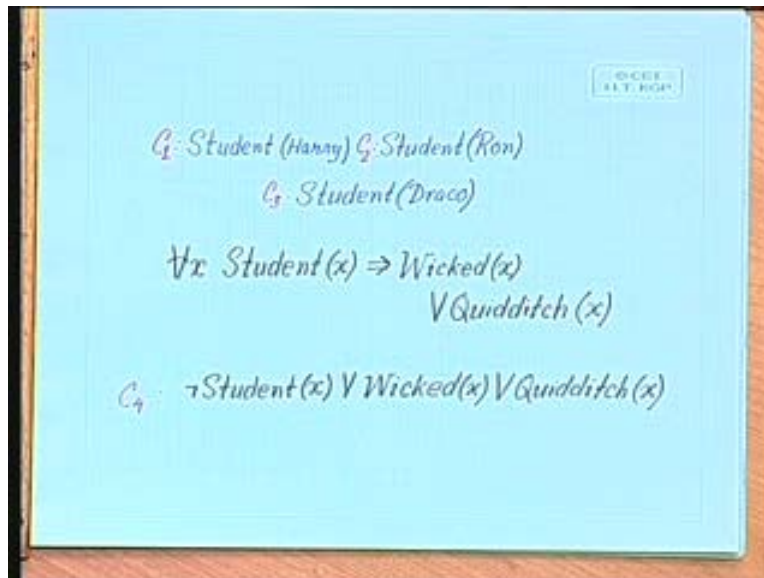
CSE, IIT Kharagpur 14

The first sentence says that Harry, Ron and Draco are students of Hogwarts school of wizards. Every student is either wicked or is a good quidditch player or both. No quidditch player likes rain, and all wicked students like potions. Draco dislikes whatever Harry likes and likes whatever Harry dislikes. Draco likes rain and potions, and the goal is that is there a student who is good in quidditch but not in potions. And, we are trying to find out whether this is true or not, whether this is valid or not, whether the set of clauses helps us in deducing, that there is a student who is good in quidditch but not in potions, right?

So, first thing that we are going to do is, to write each of these sentences in first order logic, and reduce them to clause form. Now, the nice thing about the clause form is that you have all the sentences- each of them are going to give you some clauses, and that entire clause form is just a conjunction of all these clauses. So, we can take 1 sentence at a time, convert it to clause form, take the next sentence, convert it to clause form, and then attend all the clauses together to get the full clause form. So, first sentence is, Harry, Ron and Draco are students of Hogwarts school of wizards.

So, let us say, we have a predicate called student, so the first 3 clauses that we will have, are student Harry, then, we will have student Ron, and we have- so, these are the first 3 clauses.

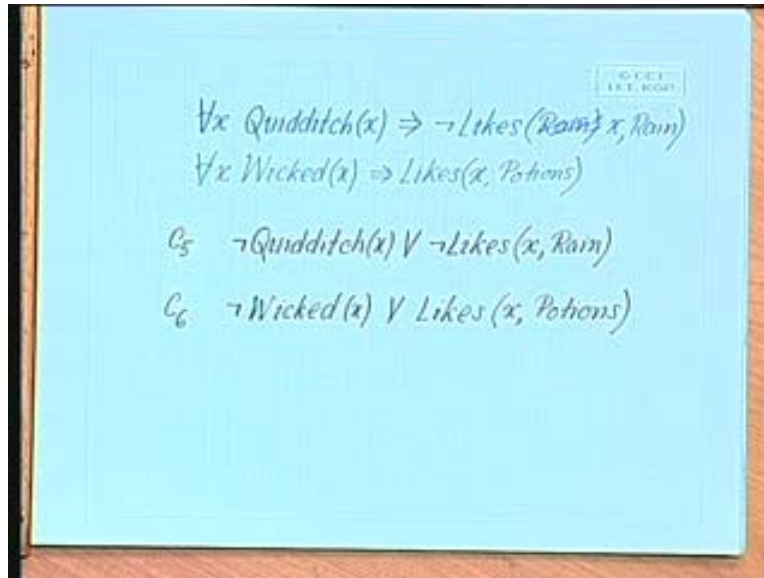
(Refer Slide Time: 00:13:02)



Then, next sentence says, that every student is either wicked or is a good quidditch player or both. Every student is either wicked or is a good quidditch player or both. So, we can write this as- for all x, student x implies either wicked- so, wicked x- or is a good quidditch player- so, let us call that predicate quidditch itself, but this is not in clause form. So, if we convert it into clause form, what we will get is- we will convert this to remove this implication. So, we will have not student x or wicked x or-, right? So, let us name these clauses. This was c1, this is c2, this is c3, this is our c4.

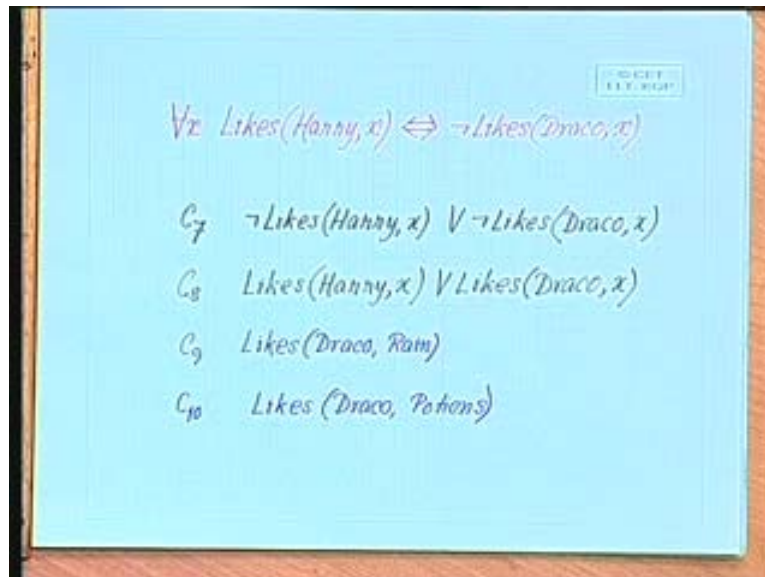
We are not writing the for all x outside, because it is implied- because, in clause form, all the variables will be existential, will be universally quantifier. Then, let us look at the third statement. It says, no quidditch player likes rain and all wicked students like potions. So, this is actually 2 sentences and let us write them down. The first sentence says that no quidditch player likes rain.

(Refer Slide Time: 00:15:42)



So, which means, that for all x, yes, implies, right, implies- not of-, right? And the other 1 is- sorry, let me write this as x comma rain, and this predicates says that x likes rain. So, likes xy is x likes y. And then, we have that all wicked students like potions, so for all x- right? Again, if we convert these to clause form, then, we will have c5, which is not of- quidditch x or, then, not this is c5, and then we have c6, which is not of. Then, we have Draco dislikes whatever Harry likes, and likes whatever Harry dislikes. This is going to be like, for all x- and in clause form, we will have again 2 clauses. So, let us call them c7, which says that not likes Harry x or not likes Draco x and c8, which says- just check out.

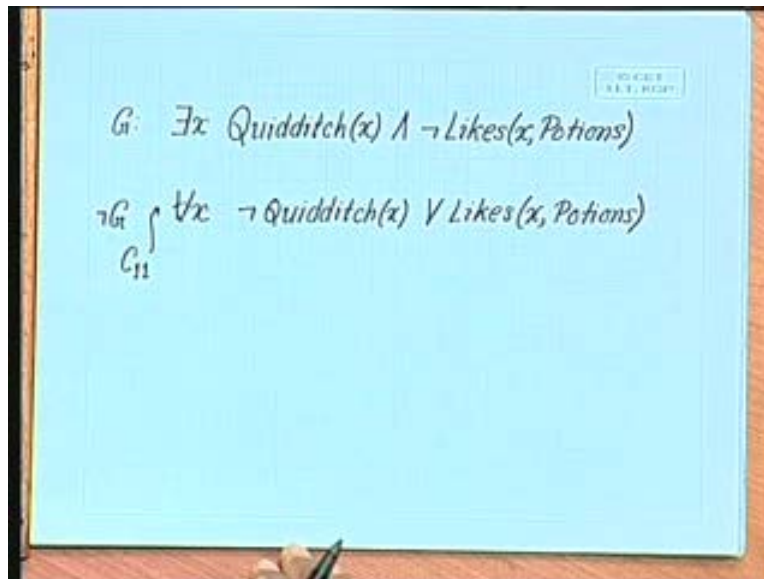
(Refer Slide Time: 00:18:27)



This is for 1 side of the implication; this is for the other side of the implication. Then, finally, we have Draco likes rain and potions. So, that is going to give us another 2 clauses. So, c9, which says likes- and c10. These 10 clauses form the knowledge base. The knowledge base currently consists of these clauses- c1 through c10. Now, we will encode the goal, and insert the negation of goal into the knowledge base and try to deduce a refutation. The goal says, is there a student who is good in quidditch but not in potions?

If we have to write that in a first order logic, we will say, there exists x, who is good in quidditch and does not like potions. If we convert this, if we take the negation of this, **this is goal** this is the goal g. Then, not of g is **[noise] not of g is [students: noise]** for all x-okay. So, this is the goal clause, not g clause, which we will insert into the knowledge base as c11.

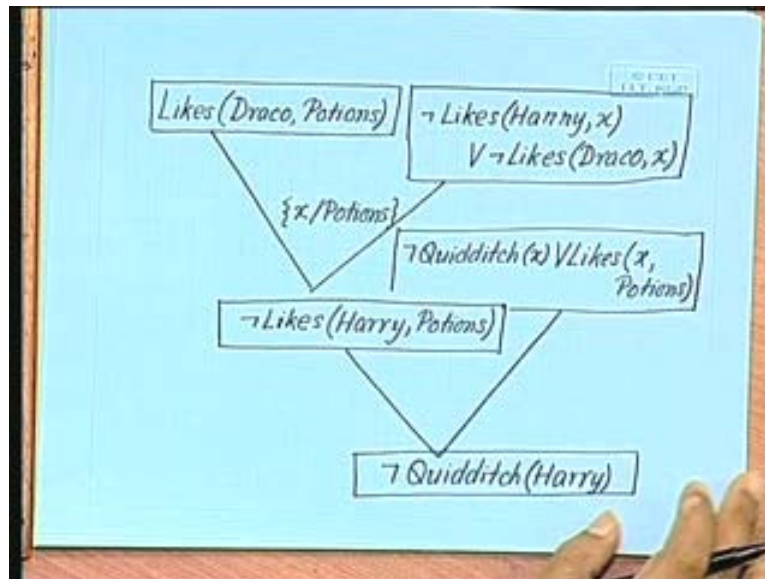
(Refer Slide Time: 00:20:30)



This is going to be our  $C_{11}$ , and then we will try to deduce false, and for deducing false, we will use resolution. And if you recall, that in the last class, we had mentioned that resolution is a complete proof procedure, so if false can be deduced at all, then, if we repeatedly apply resolution, we should be able to deduce it. If we systematically apply resolution, we should be able to deduce it. Let us look at 1 way of deducing this. We will start with 2 clauses- we will start with likes- I am just showing 1 way of doing deduction.

Then, we will see, that what are the strategies for systematically doing this deduction? So, likes Draco potions is given to us, and we have- this is 1 clause, and then we have not likes Harry  $x$  or not likes Draco  $x$ . If we apply resolution on this, then we see that this and this will unify with a substitution of  $x$  by potions, and when this unifies, because this is the negation of this, therefore, we will just have not likes Harry potions.

(Refer Slide Time: 00:24:50)

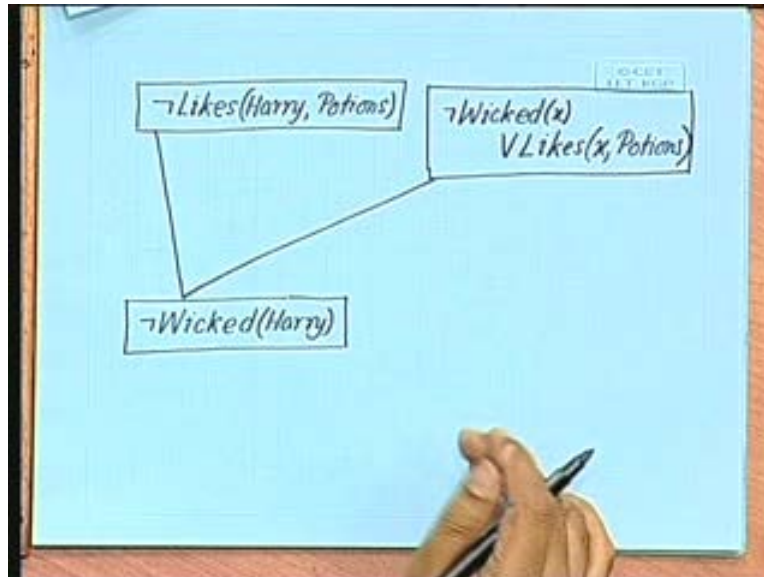


So, what resolution helps us in deducing from these, by substituting  $x$  by potions is not likes-. Then, we also have not quidditch  $x$  or likes  $x$  potions. This is another clause that we have, recall? We had this clause- not quidditch  $x$  or likes  $x$  potions- this is which clause? Yes, this is not of the goal, right? So, we use that clause with this, and again, with  $x$  substituted by potions, we will have unification between these 2, and so, with resolution, **we will be able to** we will get not-. Now, because of lack of space, I am not being able to draw the full tree on the other side, so, I will break it up and draw it separately in another.

What **do we** I am going to do is, I will use this not likes Harry potions to unify with other things also. I will carry over this 1- this clause- into the next page, and I start with not likes Harry potions. See, this is something which we have already deduced, so I can use it again to resolve with other clauses. This we have already deduced from these 2, so now, **we can** it is now there in the knowledge base, so I can use that with, say, we have this clause- not of wicked  $x$  or likes  $x$  potions.



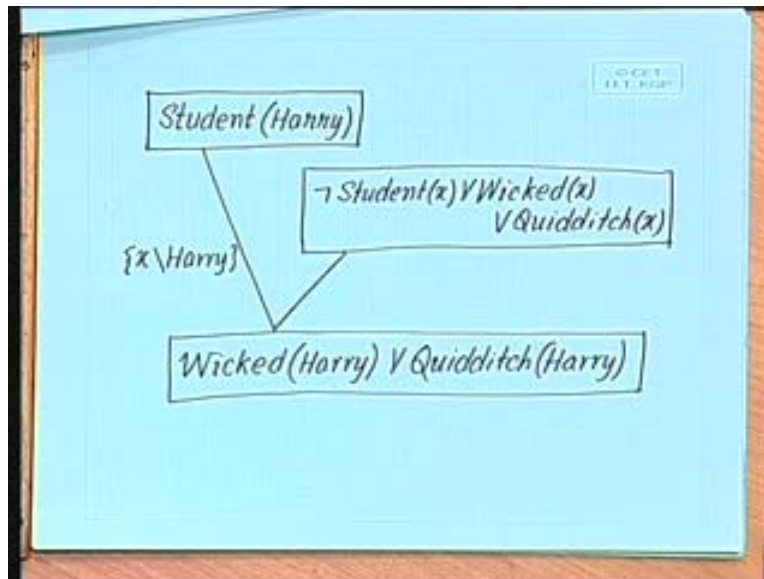
(Refer Slide Time: 00:26:47)



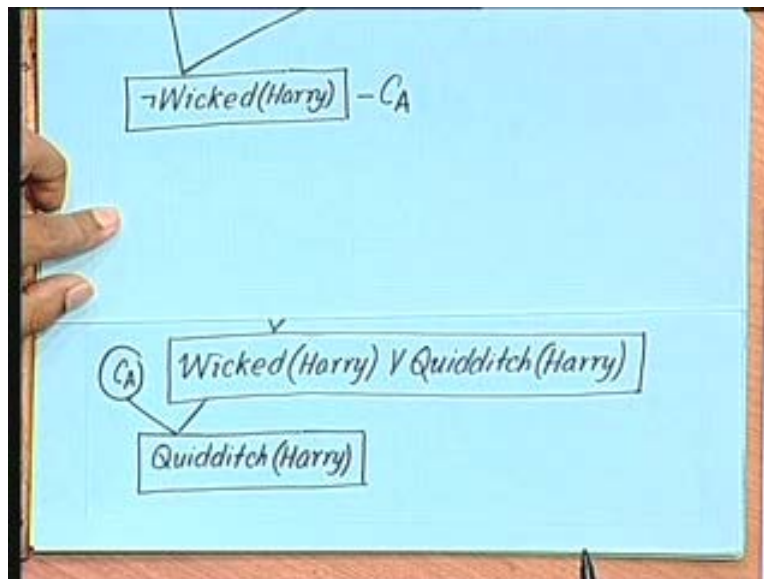
We can resolve these 2 to get- what will we get? So, we will leave this up to here for the time being, and then pick up 2 other clauses from the knowledge base and deduce something else and then come back to this. We have deduced not wicked Harry; previously, we had deduced here that not quidditch Harry. Now, let us see- we have in the knowledge base, student Harry, and we also have in the knowledge base not of student x or wicked x or-. This was another clause that was there, in fact, the first clause is told as that a student is either wicked or good in quidditch.

So, if we use these 2 with x substituted by Harry, then, what do we get? We have wicked Harry or quidditch. Now, having deduced this, we will go back to the previous one, where we had deduced not wicked Harry. So, we have not wicked Harry here, and we have wicked Harry or quidditch Harry here. Let us say, that if we use these 2, then resolving with these 2 will give us quidditch Harry.

(Refer Slide Time: 00:28:49)



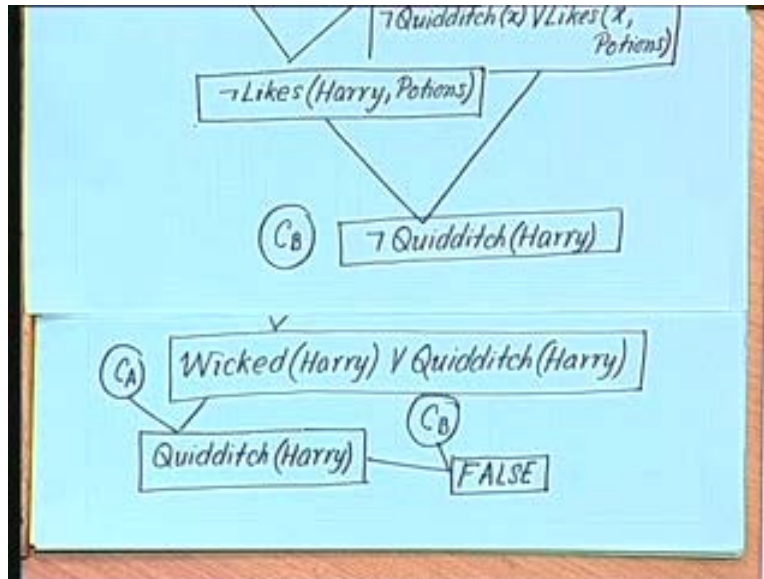
(Refer Slide Time: 00:29:45)



If we use this, and let us give this clause a name- let us say  $ca$ , then with  $ca$  and this, I am able to deduce quidditch Harry, and then, we have not quidditch Harry here. If we use these 2, so, let us call this  $cb$ ; then, with  $cb$ , and this gives us false. (Student speaking). Yes, you can do that also. Depending on how your theorem prover can handle

clauses, **it can** it may be able to resolve more than pairs of clauses at the same time. That is quite acceptable, and in fact, some theorem provers will do that.

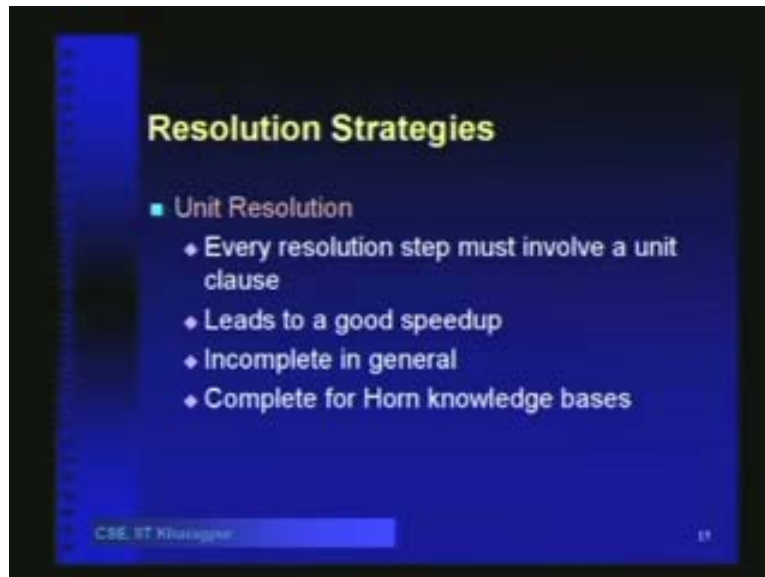
(Refer Slide Time: 00:30:12)



What we have seen so far is, that we converted the rules and facts into clause form, and then saw 1 way to deduce false. Now, we can use this resolution in many different ways also. This is 1 particular proof tree, and the proof tree of this whole thing is actually the concatenation of all these 3 pages that I have done here. So, it is a tree which eventually leads down to false. Now, what we want is a short proof tree. We do not want to derive things which we are not going to use for deducing the false, but how do we do that? How do we direct the proof mechanism towards the false?

There have been several suggestions of doing this. These are called different resolution strategies and these strategies are meant for making the search for the false more efficiently. The first 1 that we are going to look at, is unit resolution. Now, what this tells us is that every resolution step must involve a unit clause. What is a unit clause?

(Refer Slide Time: 00:34:40)



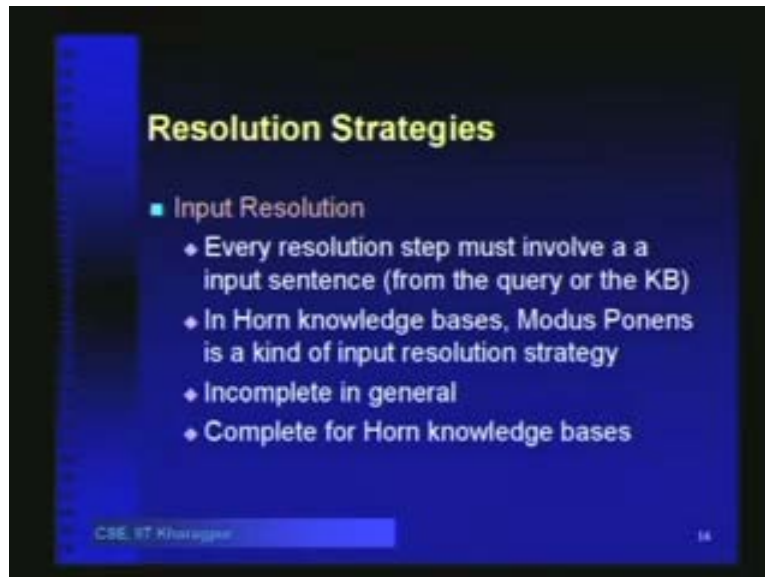
A clause which does not have any or; it just has 1 predicate, or its negation. Those are the unit clauses. Every resolution step must involve a unit clause. Let us look at the resolution that we did just now, and then you will see that this is a case of unit resolution. If you look at the- the text please. See, we started with these 2, and here, this was a unit clause. Then, after doing that, the second step was these 2, and again, this is a unit clause. Then, we deduce this- this is also unit clause. Here also, this is a unit clause, right? What we deduce was another unit clause.

Then this is a unit clause and then, ca was this one, which is a unit clause, and cb was another unit clause- this is also unit clause. At every step of the resolution, at least 1 of the clauses was a unit clause. That is what unit resolution aims to do, and as it turns out, that when this was originally proposed- unit resolution- it gave a dramatic speedup in the proof mechanism. So, it usually leads to a good speedup, but at the same time, it is incomplete in general. What do you mean by incomplete? (Student speaking). No, no. By incomplete, we mean that there will be examples, there will be cases, where you should be able to deduce the proof or deduce the goal, but using unit resolution, you will not be able to do so.

There can be cases where, if you just use a unit clause in every step of resolution, you will not be able to prove the result. So, that is what is meant by the incompleteness of unit resolution. Recall, that whenever we are talking about completeness, it is always completeness on 1 side. So, if it can be deduced, we should be able to deduce it. If it cannot be deduced, we may not terminate, we have to live with that for first order logic. So, unit resolution is incomplete in general, but it is complete for horn knowledge bases. You remember what is meant by horn knowledge bases, where you do not have- in the implication form you have on the left hand side, you have on the right side, you have only 1 literal, or you do not have any ors on the right hand side. So, that is horn logic. So, if we use horn logic- (Student speaking). Yes, because if you have ands, you can break it down into 2 rules.

Suppose I have  $a \text{ implies } b \text{ and } c$ , then I can break it down into  $a \text{ implies } b$  and  $a \text{ implies } c$ . And those 2 individually are fine, so having ands on the right hand side is not a problem. But we cannot allow  $a \text{ implies } b \text{ or } c$ , that is something which horn logic will not allow. We will study first order horn logic called prolog. In the following lecture, we will start prolog. It is a completely new kind of language that we will study and that uses this horn logic for doing the deduction. It is not as powerful as full first order logic, but it is something which is a fragment of first order logic. Then there is a strategy called input resolution. What this says is, every resolution step must involve a input sentence from the query or the knowledge base. What does this mean?

(Refer Slide Time: 00:39:28)



It means that whenever we use resolution, we must use 1 of the clauses which was there in the original knowledge base, including the query. (Student speaking). No, you cannot resolve- this prevents you from resolving 2 derived clauses, and the notion behind this is- let us see what we have in the knowledge base, **that is going** that is what is going to lead us to false. We are going to always try to use those, rather than trying to resolve 2 derived clauses. Typically, we will start with the goal. We will start with the goal clause, because assuming that the knowledge base itself was consistent; addition of the goal is expected to make it inconsistent if we have to derive false.

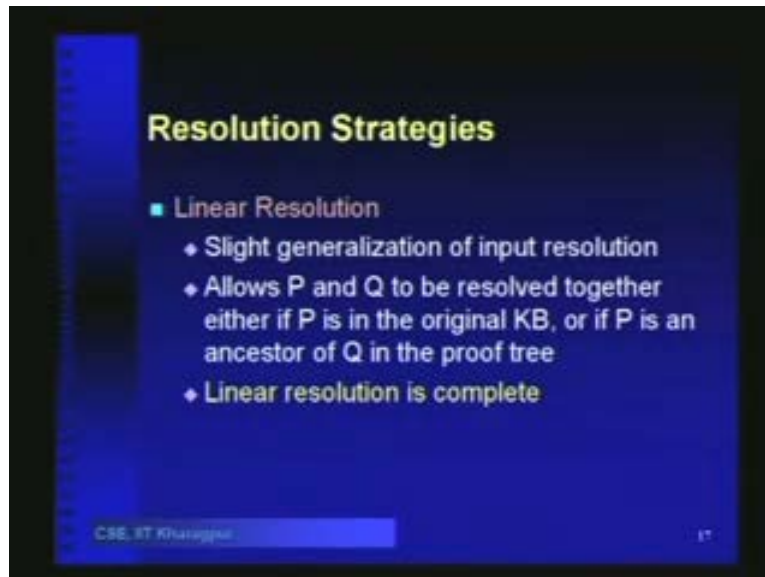
So, whatever derivation takes place, must take place around this goal that we have inserted. We will start from the goal and keep on doing this resolution. At each step, at least 1 of the clauses should be from the original knowledge base or the goal. Now, if you look at the derivation that we did just now this, was not input resolution. Why? Because if you see that in some places, let us look at this- here it is fine, because these were there in the original knowledge base. Here, this was in the original knowledge base, so this step is also fine. But when we use, say, okay, this is also fine, but then, when we do this, this is a derived clause ca, which is not of wicked Harry, and this is also a derived clause,

which is wicked Harry and quidditch Harry or quidditch Harry, and this type of resolution is not is against input resolution. Input resolution will not allow you to do this.

You have to use always at every step, **1 of the** at least 1 of the clauses from the original knowledge base. Now, in horn knowledge bases, modus ponens is a kind of input resolution strategy, because modus ponens is what you are- something implies something; a implies b, and you are given a, then, you deduce b. So, 1 of those rules are part of the knowledge base. Every time you are using a rule to deduce something, it is a kind of input resolution. It is again incomplete in general and complete for horn knowledge bases. But there is generalization of input resolution, which is complete. Let us see what that is. (Student speaking). Yes, yes. That is why this is incomplete; that is why input resolution is incomplete.

There are **there are** examples, where, by using derived clauses, you will be able to deduce false. But if you just stick to input resolution, you will not able to deduce it. That is why it is incomplete. But it is complete for horn knowledge bases, because you always have that set of clauses implies something else, which is essentially modus ponens. For horn clauses, **this is always** this is a complete proof procedure, otherwise, it is not. So, the generalization of input resolution which is complete, is called linear resolution. Now, what generalization do we do? We will allow clauses p and q to be resolved together. Either if p is in the original knowledge base, which is as in input resolution, or if p is an ancestor of q in the proof tree. So, in the proof tree, we are taking clauses, deriving something out of them, then using this 1 with some other clauses, deriving something out of them, and so on.

(Refer Slide Time: 00:41:50)

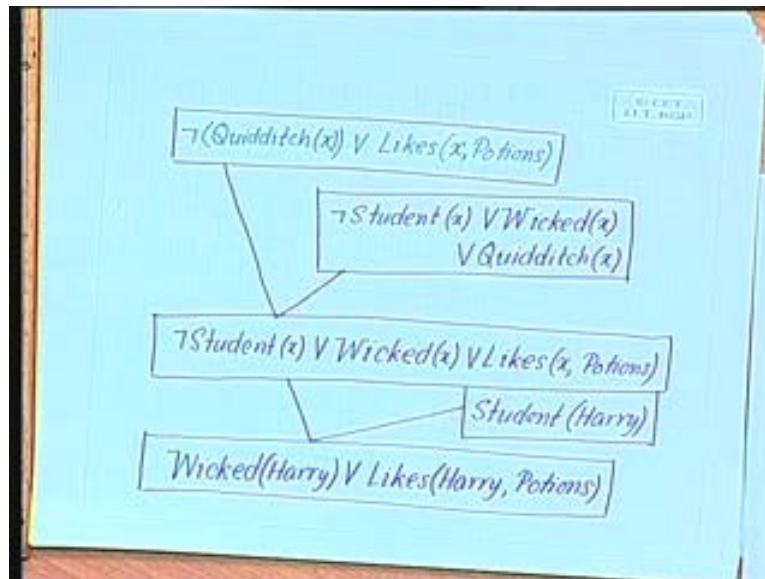


So, at every point of time, the 2 clauses that you will resolve- 1 of them has to be either from the original knowledge base, or it can be 1 which we have derived on this path of the proof tree. It is an ancestor in the proof tree. If we use that, that is fine, and we will be able to- as it turns out, that linear resolution is complete. Let us quickly see, whether the example that we have done can be derived using linear resolution. Actually, this particular case can be derived using input resolution also, but in general, input resolution is not complete, but linear resolution is complete. Please take some time to think about why linear resolution is complete, but input resolution is not.

So, this simple generalization, which allows us to use the ancestors clauses in the proof tree, makes the resolution procedure complete. How will you show that it is complete? Take any resolution proof and try to convert it into an equivalent linear resolution proof. If you are able to do that, then, that means the linear resolution is complete. Just take some time off to think and work this out on your own. Let us quickly apply linear resolution on this.



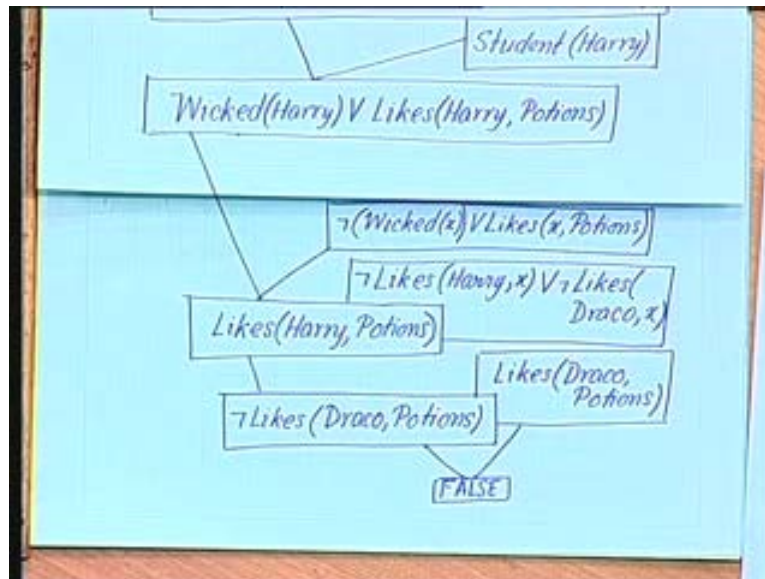
(Refer Slide Time: 00:46:00)



We will start with the goal. The goal clause was not of quidditch or-, right, this was the goal clause and we will resolve it with not student x or- and that is going to give us-; again, this linear resolution can be done in multiple ways. I am just showing 1 way of doing. Then, we will resolve again with student Harry. So far, we are using 1 clause always from the input, so it is input resolution as well. These were from the original knowledge base. Now, we are using this, which is from the original knowledge base. So, with this, we derive wicked Harry or-. See, what this linear resolution helps us in doing is that, because you are not going to use 2 derived clauses, unless they are on the same path of the tree, so, the tree is not going to branch out a lot.

The tree is not going to branch out a lot; it is going to be something like a linear tree. That also helps us in keeping the knowledge base- size of the knowledge base- contained, because if the knowledge base size starts growing, then we will end up in trouble. So, here, we have wicked Harry and likes Harry potions, and we will resolve this with- text please.

(Refer Slide Time: 00:49:03)



We will resolve this with- again, this is from the original knowledge base. We use this and this to get- and then, we will use these 2 together, will give us- this is not likes- and we have in the knowledge base- we have likes Draco potions, this gives us-. So, this whole thing is the proof tree starting right from here, when we had this derivation, then, we end up to false. Now, this whole derivation- at every step, we have used 1 of the clauses from the original knowledge base. We started with the goal and kept using 1 clause from the knowledge base, until we deduce false. So, this is both- yes, both input resolution as well as linear resolution.

If any proof satisfies input resolution, it by default satisfies linear resolution, but the reverse is not true. And probably, I will reserve it for your exam, to give you some case which will work for linear resolution but not for input resolution. So, please try to think, that how to discover that this is going to be the case. With that, we come to the end of this particular lecture, and we also come to the end of our analysis of first order logic. In the next lecture, I will start with the language prolog, which is logic programming language, and we will see how to write different kinds of programs with prolog. After finishing off with prolog, then, we will start the **planning the** chapters on planning.

The whole of- a major part of the analysis of artificial intelligence is centered around search; that is why we started with search. Search is the most **basic the** basic mechanism, which allows us to solve these things. As you can see, now, this is also a kind of search. Deduction is also a kind of search, where you have to search within the rules and facts and clauses, and to search and find out the appropriate order in which to apply them to deduce this. We will see later on- a little bit of constraint logic programming, where with deduction, we will be able to use also constraints. And there has been recent annotations to constrain logic programming, which allows you to solve optimization problems, as we did in heuristic search, using a detecting frame work.

So, the detective framework is just an interface to write what we want, to do using a logic, and the solver is at the back end. In procedural languages, the solver that you have is the 1 or Norman architecture which does this fetch-decode-execute cycle. We know that after this statement, this statement is going to be executed. That is 1 flow that we understand and which we believe is the core of computation. Now, logic programming **is** **is** does the computation in a different way. So, you have to write the program knowing that the computation will happen that way, right? So, in the next lecture, we will start off with logic programming.