Fundamentals of Artificial Intelligence

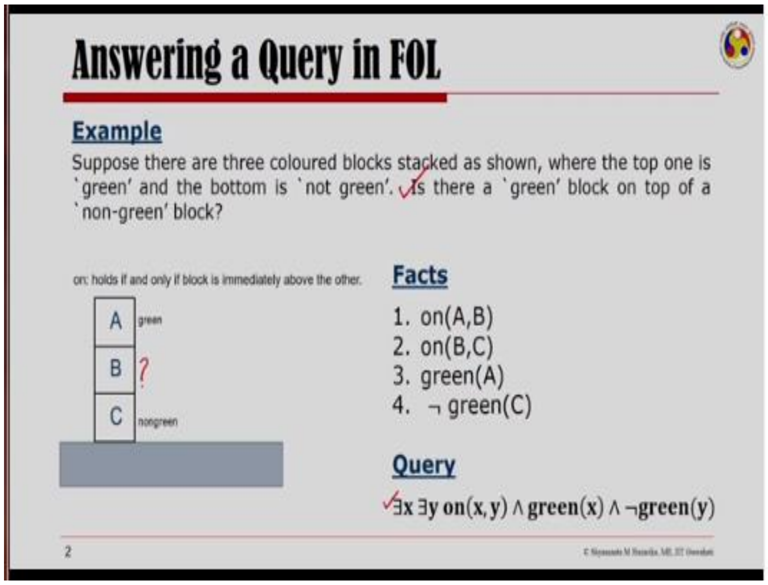
Prof. Shyamanta M Hazarika
Department of Mechanical Engineering
Indian Institute of Technology- Guwahati

Lecture-16
Answer Extraction

Welcome to fundamentals of artificial intelligence, we continue our discussion on knowledge representation and reasoning. In the last lecture we have looked at resolution in first order logic, and a walk through resolution refutation proofs, proof within a logical inference system can be applied for information retrieval or automatic problem solving. These applications have procedures to extract the information from the proof once they are found.

One such procedure is the procedure of answer extraction which extracts the information about an existentially quantified variable in the goal from proofs using the resolution principle. This is what we will look in more depth today in our lecture, so here is our blocks world problem that we have introduce in the last lecture.

(Refer Slide Time: 02:01)



Answering a Query in FOL

Example

Suppose there are three coloured blocks stacked as shown, where the top one is `green' and the bottom is `not green'. Is there a `green' block on top of a `non-green' block?

on: holds if and only if block is immediately above the other.

A
green
B
?
C
nongreen

Facts

1. on(A,B)
2. on(B,C)
3. green(A)
4. $\neg$ green(C)

Query

$\exists x \exists y \, on(x,y) \wedge green(x) \wedge \neg green(y)$

2

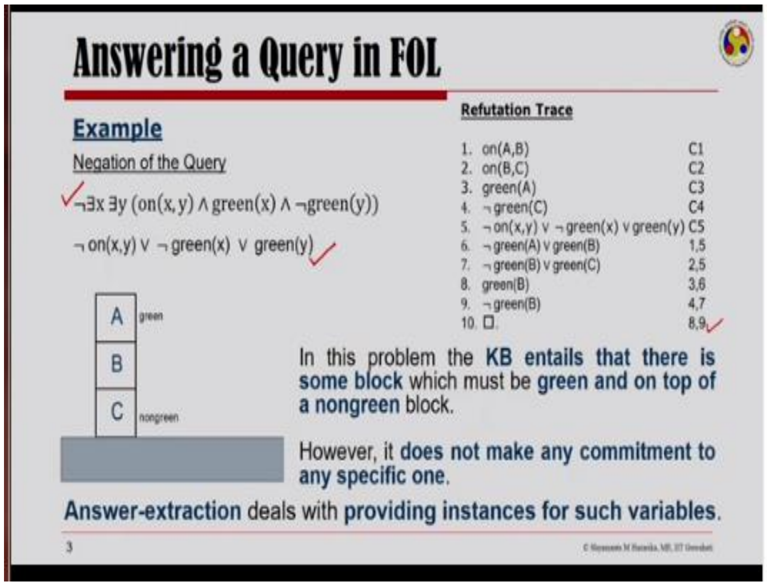
© Shyamanta M Hazarika, ME, IIT Guwahati

There are 3 colored blocks stacked as shown the top one is green and the bottom one is not green now the query is, is there a green block on top of a non green block. Now can a first order logic inference system answer such a query. So if you look at the query we can write it as there exists x, there exists y on x, y and green x and not green y where the predicate on holds if and only if a block x is immediately above another block y, a predicate green x is to mean that the block is green.

And a predicate negation of green of x would mean that the block is non green, now given these predicates and the scenario that is being shown here, where I have a green block at the top a block B which whether it is green or not green is not known and a block C which is non green is there a process to confirm that there is a green block on top of a non green block. Now intuitively if you argue in this system you could see that B could be either green or not green.

If B is a green block then this statement is true because I have a green block which is B on top of a non green block which is C. On the other hand if B is a non green block then also the statement is true because then I have A which is a green block on top of B which is a non green block. Now in order to show that this statement is true what we do in a resolution refutation proof is we

(Refer Slide Time: 04:36)


A
green
B
C
nongreen

Answering a Query in FOL

Example

Negation of the Query

✓ $\neg\exists x\,\exists y\,(on(x,y) \wedge green(x) \wedge \neg green(y))$

$\neg on(x,y) \vee \neg green(x) \vee green(y)$ ✓

Refutation Trace

1.	$on(A,B)$	C1
2.	$on(B,C)$	C2
3.	$green(A)$	C3
4.	$\neg green(C)$	C4
5.	$\neg on(x,y) \vee \neg green(x) \vee green(y)$	C5
6.	$\neg green(A) \vee green(B)$	1,5
7.	$\neg green(B) \vee green(C)$	2,5
8.	$green(B)$	3,6
9.	$\neg green(B)$	4,7
10.	$\square$.	8,9 ✓

In this problem the **KB entails that there is some block** which must be **green and on top of a nongreen** block.

However, it **does not make any commitment to any specific one.**

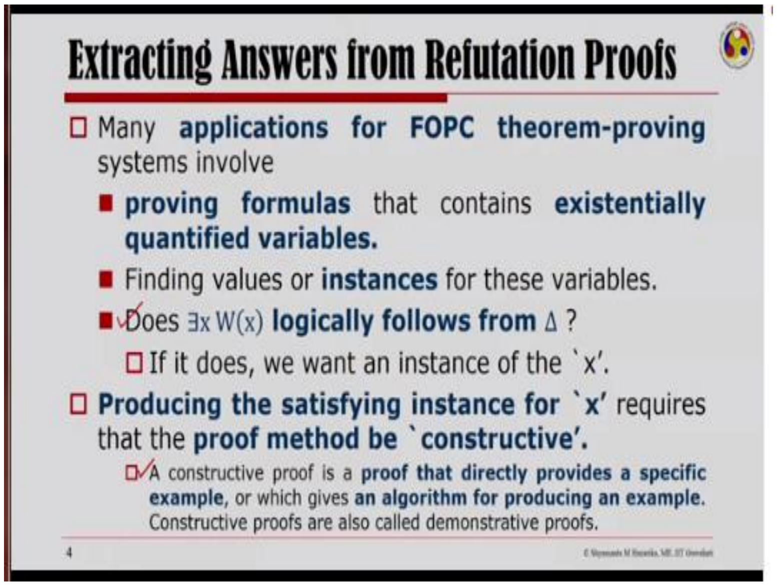
Answer-extraction deals with **providing instances for such variables.**

3 © Shyamanta M Hazarika, ME, IIT Guwahati

take the negation of the query, so we write here not there exists x there exists y on x, y and green x and not green y. And if you push the negation symbol inside we will have for all x for all Y not of this and then I can get to the class form. So this statement here is the clausal normal form of the negation of the query and what I have on the right of your screen is the refutation trace to finally arrive at the empty clause.

So the statement that there is a green block on a non green block is logically implied by the set of clauses that I get from the initial scenario. But one thing to note here is the following that in this problem the knowledge base entails that there is some block which must be green on top of a non green block. However what it does not do is it does not make any commitment to any specific one, so it does not tell us whether this statement is true because the block B is green or the statement is true because block B is non green. We have no answer to what the existentially quantified variables are. So answer extraction deals with providing instances for such variables.

(Refer Slide Time: 06:30)



Extracting Answers from Refutation Proofs

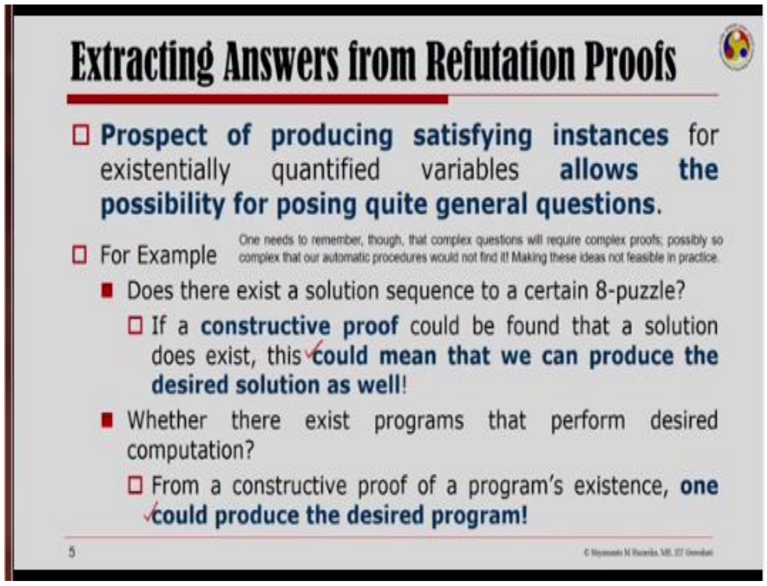
- Many **applications for FOPC theorem-proving** systems involve
  - **proving formulas** that contains **existentially quantified variables.**
  - Finding values or **instances** for these variables.
  - Does $\exists x\, W(x)$ **logically follows from** Δ ?
    - If it does, we want an instance of the `x'.
- **Producing the satisfying instance for `x'** requires that the **proof method be `constructive'.**
  - A constructive proof is a **proof that directly provides a specific example**, or which gives **an algorithm for producing an example.** Constructive proofs are also called demonstrative proofs.

4

Now extracting answers from refutation proves there are many applications of first order logic theorem proving systems which look at this proving formulas that contain existentially quantified variables. And finding values or instances of these variables would mean that I could not only talk of that there is a X such that W of x logically follows from the initial clause set. I would also be able to get an instance of the X doing this I can produce the satisfying instance for X.

Now this requires that the proof method is constructive let us take one minute to understand, what is a constructive proof. A constructive proof is a proof that directly provides a specific example or which gives an algorithm for producing an example. So in a sense constructive proofs are called demonstrative proofs because they demonstrate the existence of the existentially quantified variable by showing us an example of that particular object.

(Refer Slide Time: 07:55)


Extracting Answers from Refutation Proofs

- **Prospect of producing satisfying instances** for existentially quantified variables **allows the possibility for posing quite general questions.**
- For Example

  One needs to remember, though, that complex questions will require complex proofs; possibly so complex that our automatic procedures would not find it! Making these ideas not feasible in practice.
  - Does there exist a solution sequence to a certain 8-puzzle?
    - If a **constructive proof** could be found that a solution does exist, this **could mean that we can produce the desired solution as well**!
  - Whether there exist programs that perform desired computation?
    - From a constructive proof of a program's existence, **one could produce the desired program!**

5

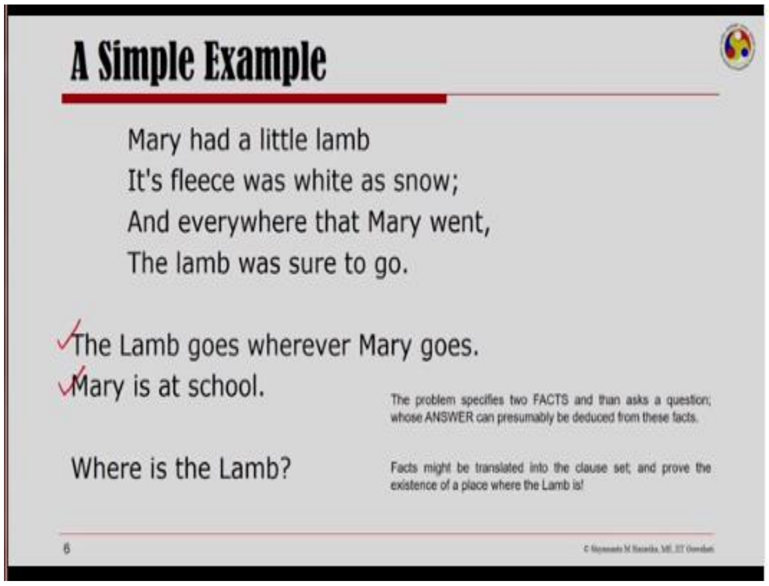
© Shyamanta M Hazarika, ME, IIT Guwahati

There is a huge consequence of the prospect of producing satisfying instances for existentially quantified variables. This allows the possibility for posing quite general questions like for example, I could ask a question like does there exist a solution sequence to a certain 8 puzzle. Now if I ask that question and I have a demonstrative proof that there exists a solution sequence to a certain 8 puzzle that would mean that I also have the path to the solution.

So if a constructive proof could be found that a solution does exist this could mean that we can produce the desired solution as well or it could even be stress further to I could ask whether there exist programs that perform desired computation. Now if I can have a constructive proof of a program's existence then one could produce the desired program. One needs to understand though that complex questions will in the first place require very complex proofs possibly.

So complex that our automatic procedures that we have in place would not find the proof in the first place. So making these ideas not feasible in practice however it is interesting to note that if I can extract answers from refutation proofs that is I can produce satisfying instances for existentially quantified variables. Then it allows the possibility for posing quite general questions and have wide implications.

(Refer Slide Time: 09:51)



A Simple Example

Mary had a little lamb
It's fleece was white as snow;
And everywhere that Mary went,
The lamb was sure to go.

✓The Lamb goes wherever Mary goes.
✓Mary is at school.

Where is the Lamb?

The problem specifies two FACTS and than asks a question; whose ANSWER can presumably be deduced from these facts.

Facts might be translated into the clause set; and prove the existence of a place where the Lamb is!

6

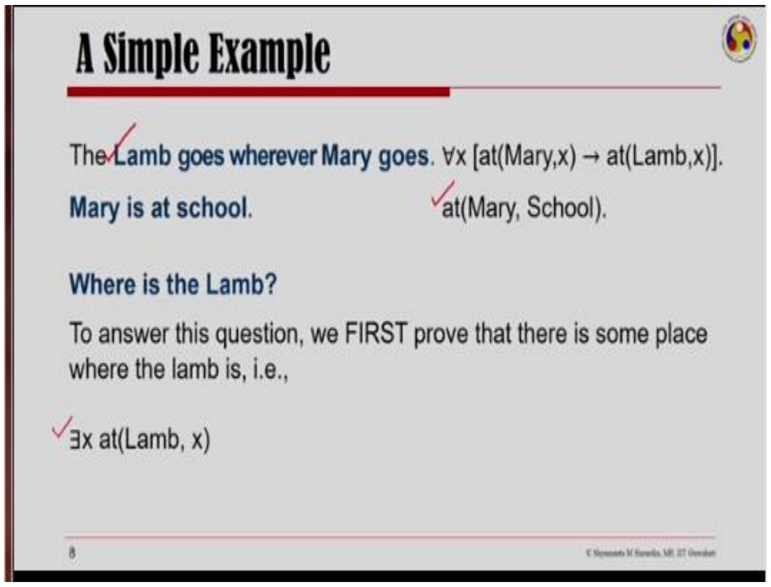
© Shyamanta M Hazarika, ME, IIT Guwahati

Let us take a simple example first and try to understand the answer extraction process here we will talk of resolution refutation being performed and then how that refutation tree is converted to a modified proof tree to get to the answer. But this discussion will be a little casual and then we will look at more depth on how this is done after we have done with this example. So here is a very popular nursery rhyme.

Mary had a little lamb it’s fleece was white as snow and everywhere that Mary went, the lamb was sure to go this if you hear then you know that the lamb goes wherever Mary goes and then at one point of time I am told that Mary is at school. So if I ask you where is the lamb one can very well answer that the lamb is at the school, how do you find such an answer within a first order logic resolution system.

So here if you see the problem specifies 2 facts one that the lamb goes wherever Mary goes, 2 that Mary is at school and from these facts the answer can presumably be deduce in order to do that I must first convert the facts into the clause set. And then I should be able to prove the existence of a place where the lamb is.

(Refer Slide Time: 11:46)



A Simple Example

The Lamb goes wherever Mary goes. $\forall x\, [at(Mary,x) \rightarrow at(Lamb,x)]$.

Mary is at school. $at(Mary, School)$.

Where is the Lamb?

To answer this question, we FIRST prove that there is some place where the lamb is, i.e.,

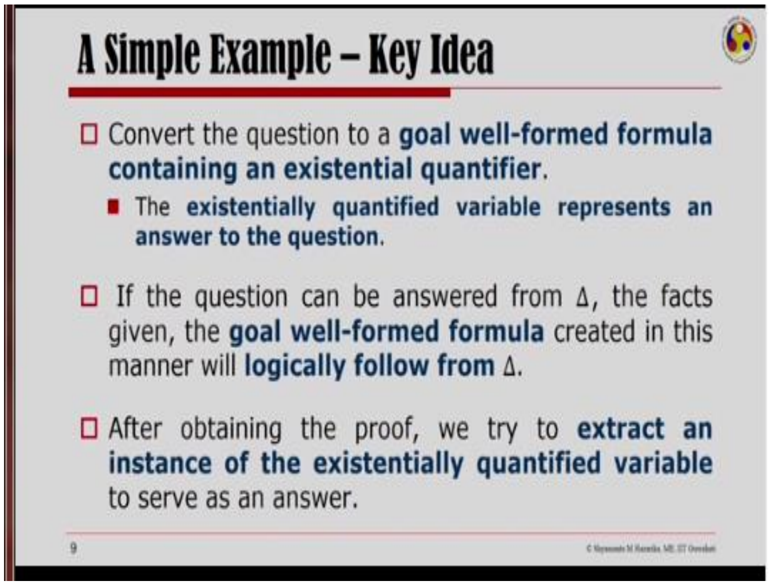
$\exists x\ at(Lamb, x)$

8

So the question where is the lamb to answer this question we first prove that there is some place where the lamb is. So we say there is an x at lamb, x now just to take a note here that the predicate being used here is at x, y to mean that x is at place y. So here the statement lamb goes wherever Mary goes instead of bothering about places and other things I have introduce a predicate at to say that for all x if Mary is a x the lamb will also be at x.

So this is the final result of that particular statement, now the next statement says Mary is at school. So I have a statement here at Mary, school where is the lamb I would have to have an existential that there is an x at lamb, x. I take the negation of the statement and then try to get a resolution refutation done but before that let us see the key idea.

(Refer Slide Time: 13:12)

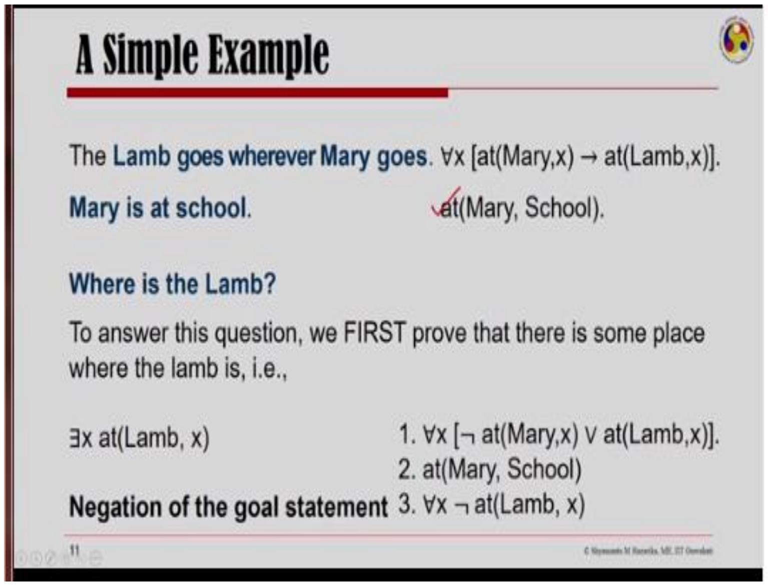

A Simple Example – Key Idea

- Convert the question to a **goal well-formed formula containing an existential quantifier.**
  - The **existentially quantified variable represents an answer to the question.**
- If the question can be answered from Δ, the facts given, the **goal well-formed formula** created in this manner will **logically follow from Δ.**
- After obtaining the proof, we try to **extract an instance of the existentially quantified variable** to serve as an answer.

9 © Shyamanta M Hazarika, ME, IIT Guwahati

The key idea is to convert the question to a goal well form formula which contains an existential quantifier. Now the existentially quantified variable actually represents an answer to the question if the question can be answered from the clause set that is the facts given then the goal well form formula created in this manner will logically follow from the clausal set. So after obtaining the proof we try to extract an instance of the existentially quantified variable to serve as an answer.

(Refer Slide Time: 13:57)


A Simple Example

The **Lamb goes wherever Mary goes.** $\forall x\, [at(Mary,x) \rightarrow at(Lamb,x)]$.

Mary is at school. $at(Mary, School)$.

Where is the Lamb?

To answer this question, we FIRST prove that there is some place where the lamb is, i.e.,

$\exists x\ at(Lamb, x)$

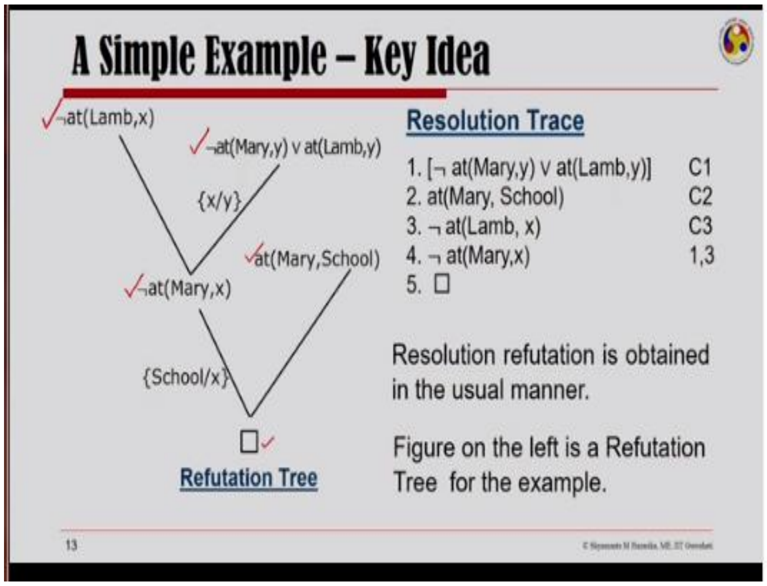
1. $\forall x\, [\neg\, at(Mary,x) \vee at(Lamb,x)]$.
2. $at(Mary, School)$
3. $\forall x\, \neg\, at(Lamb, x)$

Negation of the goal statement

11 © Shyamanta M Hazarika, ME, IIT Guwahati

So coming back to the example we had this existentially quantified statement there exists x at lamb, x we take the negation of the goal statement and also we convert the 2 facts given to us. So the implication is left out and I have this already in a clausal normal form, so I have these 3 clauses.

(Refer Slide Time: 14:26)



A Simple Example – Key Idea

Resolution Trace

1. [¬ at(Mary,y) v at(Lamb,y)]	C1
2. at(Mary, School)	C2
3. ¬ at(Lamb, x)	C3
4. ¬ at(Mary,x)	1,3
5. □	

Resolution refutation is obtained in the usual manner.

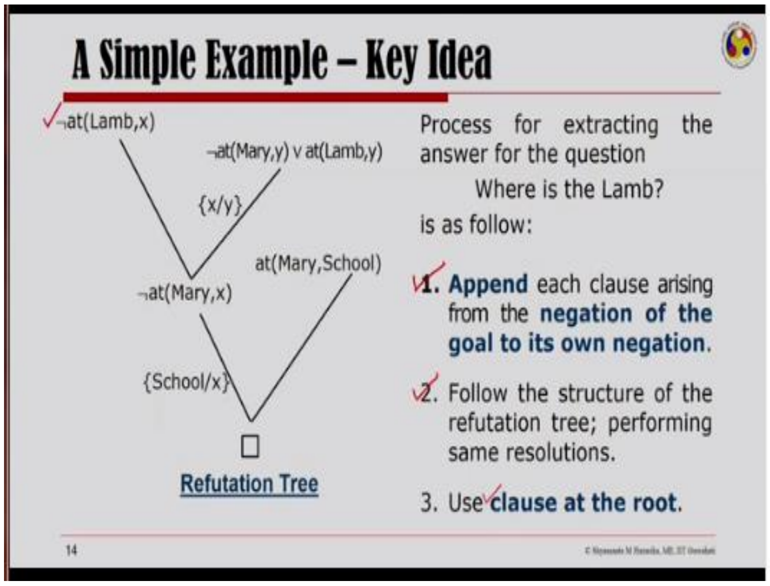
Figure on the left is a Refutation Tree for the example.

13

Here is the resolution trace of this problem, so the resolution refutation is obtained in the usual manner the figure on the left is the refutation tree. So here is the negation of the goal not at lamb, X and here is the statement saying the lamb goes wherever Mary goes, from that I have not at Mary, X and then I know Mary is at the school, so using a substitution which is school for X, I have the empty clause.

This was about getting to the proof of the statement that there is some place where the lamb is but then the idea is to get to the particular place that is what is the value of that existentially quantified variable X in the goal well-formed formula.

(Refer Slide Time: 15:41)



A Simple Example – Key Idea

Process for extracting the answer for the question

Where is the Lamb?

is as follow:

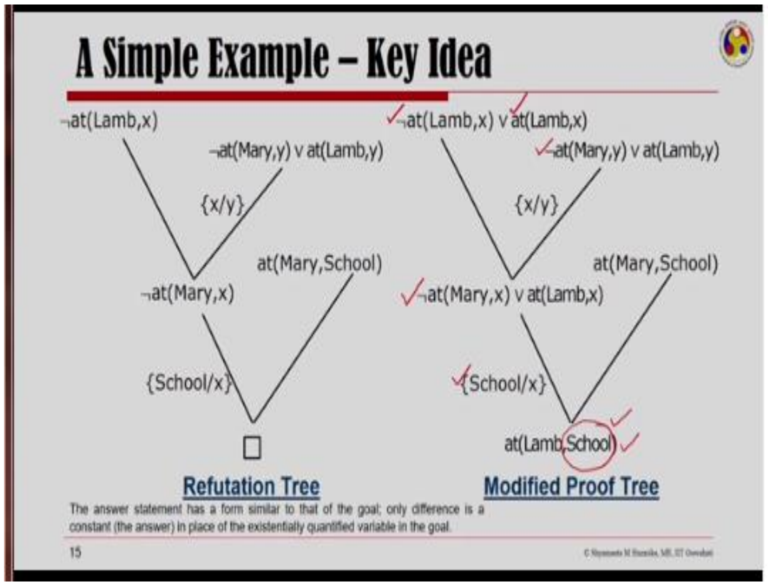
1. **Append** each clause arising from the **negation of the goal to its own negation.**
2. Follow the structure of the refutation tree; performing same resolutions.
3. Use **clause at the root.**

14

So the process for extracting the answer for the question where is the lamb is as follows I append each clause arising from the negation of the goal to it is negation, that is this was the statement which was negation of the goal. If I append this to its own negation that means I would be converting a tautology, so basically this statement here the goal well-formed formula is converted to a tautology and then you follow the structure of the refutation tree.

So when I mean follow the structure of the refutation tree I mean that you perform the same resolutions and use the same unifications there. Finally the clause that you have at the root will be the answer that we are looking for.

(Refer Slide Time: 16:45)



A Simple Example – Key Idea

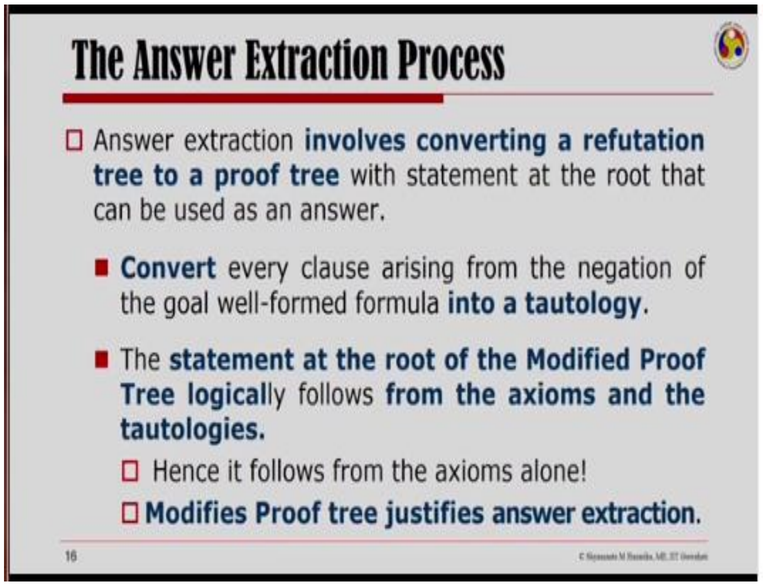
The answer statement has a form similar to that of the goal; only difference is a constant (the answer) in place of the existentially quantified variable in the goal.

15

So the figure on your right now shows the modified proof tree where the negation of the goal together with it is own negation has been added up and we have a tautology and we follow the same resolutions of the refutation tree. So first we resolve it wherever Mary goes the lamb goes and we use the same unification and here instead of at Mary x, I have not at Mary x or at lamb, x and then this substitution school for x instead of an empty clause at the root.

Now I have at lamb, school this is interesting to note that the answer statement that I have got here is of the same form to that of the goal only differences that now I have a constant here school in place of the existentially quantified variable in the goal. And this is the answer to the question where is the lamb, so I could say the lamb is at the school.

(Refer Slide Time: 18:14)



The Answer Extraction Process

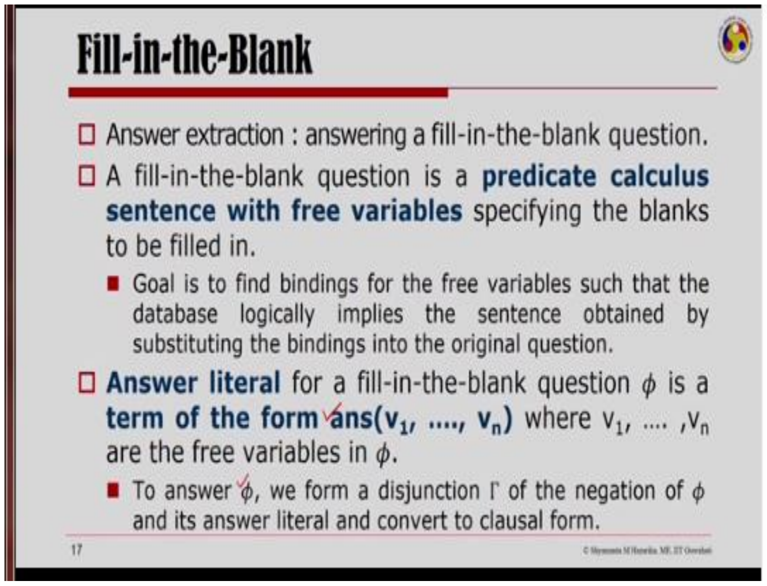
- Answer extraction **involves converting a refutation tree to a proof tree** with statement at the root that can be used as an answer.
  - **Convert** every clause arising from the negation of the goal well-formed formula **into a tautology.**
  - The **statement at the root of the Modified Proof Tree logical**ly follows **from the axioms and the tautologies.**
    - Hence it follows from the axioms alone!
    - **Modifies Proof tree justifies answer extraction.**

16

So let us now look at in more depth the process of answer extraction and try to understand why it works, answer extraction involves converting a refutation tree to a proof tree which statement at the root that can be used as an answer. You convert every clause arising from the negation of the goal well-formed formula into a tautology. So the statement at the root of the modified proof tree logically follows from the axioms and from the tautologies.

And hence we could say that it follows from the axioms alone, so the modified proof tree by the very fact that it works on tautologies and the axioms justifies answer extraction.

(Refer Slide Time: 19:14)



Fill-in-the-Blank

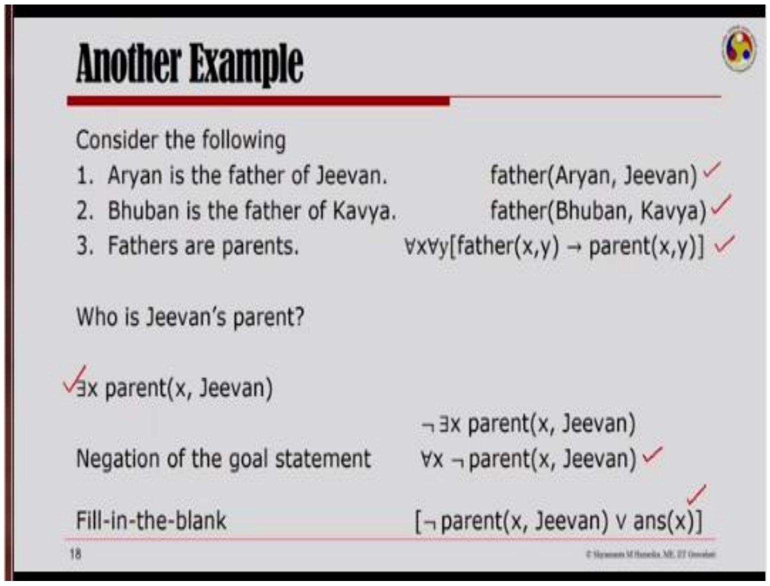
- Answer extraction : answering a fill-in-the-blank question.
- A fill-in-the-blank question is a **predicate calculus sentence with free variables** specifying the blanks to be filled in.
  - Goal is to find bindings for the free variables such that the database logically implies the sentence obtained by substituting the bindings into the original question.
- **Answer literal** for a fill-in-the-blank question ϕ is a **term of the form ans(v_1, ..., v_n)** where v_1, ... ,v_n are the free variables in ϕ.
  - To answer ϕ, we form a disjunction Γ of the negation of ϕ and its answer literal and convert to clausal form.

17

So answer extraction can also be looked at as answering a fill in the blank question like I could think of a fill in the blank question as a predicate calculus sentence with free variables which specify the blanks to be filled in. So the goal is to find bindings for the free variables such that when the database logically implies the sentence obtained by substituting the bindings into the original question the gap has been filled up and I know what is the answer to my question.

So answer literal for a fill in the blank question phi is a term of the form answer v1 to vn, where v1 to vn are the free variables in phi. Now to answer phi we form a disjunction of course we take the negation of the question being asked so we take negation of phi and we take the answer literals and convert it to a clausal form.

(Refer Slide Time: 20:29)



Another Example

Consider the following

1. Aryan is the father of Jeevan. father(Aryan, Jeevan) ✓
2. Bhuban is the father of Kavya. father(Bhuban, Kavya) ✓
3. Fathers are parents. $\forall x \forall y[\text{father}(x,y) \rightarrow \text{parent}(x,y)]$ ✓

Who is Jeevan's parent?

✓ $\exists x$ parent(x, Jeevan)

Negation of the goal statement $\neg \exists x$ parent(x, Jeevan)
$\forall x \neg$ parent(x, Jeevan) ✓

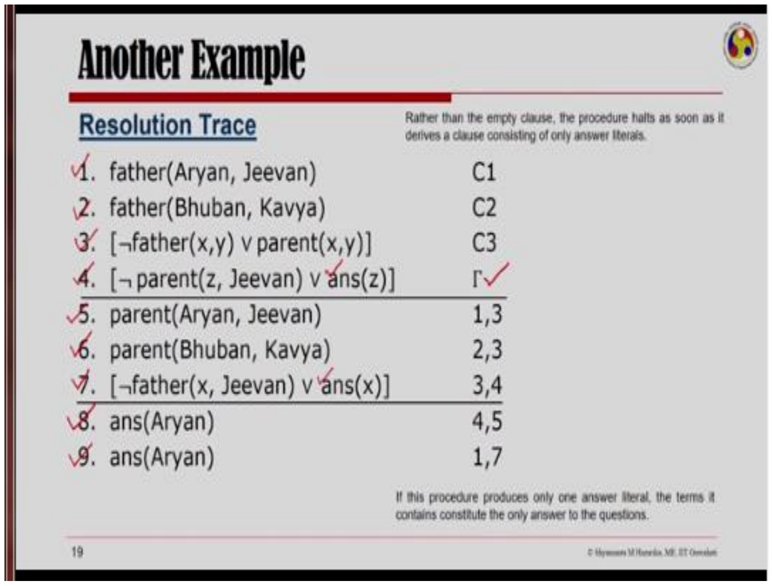
Fill-in-the-blank ✓ $[\neg$ parent(x, Jeevan) $\vee$ ans(x)$]$

18

Let us take an example to understand how this happens and how I arrived at the answer, so consider the following sentences Aryan is the father of Jeevan, Bhuban is the father of Kavya, fathers are parents. So if I ask you now who is Jeevan's parent then you know father's are parents and you know Aryan is father of Jeevan, so the answer could be that Aryan is the parent of Jeevan. In order to ask this question I will first say there exists an x who is a parent of Jeevan.

And then I could think of writing the facts of the problem given to me the first is father Aryan, Jeevan the next is father Bhuban, Kavya. And then a rule which is saying that for all X, for all Y, X is the father of Y means X is the parent of Y. Now in order to do a fill in the blank question answering I can take the negation of the goal which is not of there exist parent X, Jeevan. And this is not parent X, Jeevan I could add the answer literal answer of X.

(Refer Slide Time: 22:20)



Another Example

Resolution Trace

Rather than the empty clause, the procedure halts as soon as it derives a clause consisting of only answer literals.

1.	father(Aryan, Jeevan)	C1
2.	father(Bhuban, Kavya)	C2
3.	[¬father(x,y) ∨ parent(x,y)]	C3
4.	[¬ parent(z, Jeevan) ∨ ans(z)]	Γ ✓
5.	parent(Aryan, Jeevan)	1,3
6.	parent(Bhuban, Kavya)	2,3
7.	[¬father(x, Jeevan) ∨ ans(x)]	3,4
8.	ans(Aryan)	4,5
9.	ans(Aryan)	1,7

If this procedure produces only one answer literal, the terms it contains constitute the only answer to the questions.

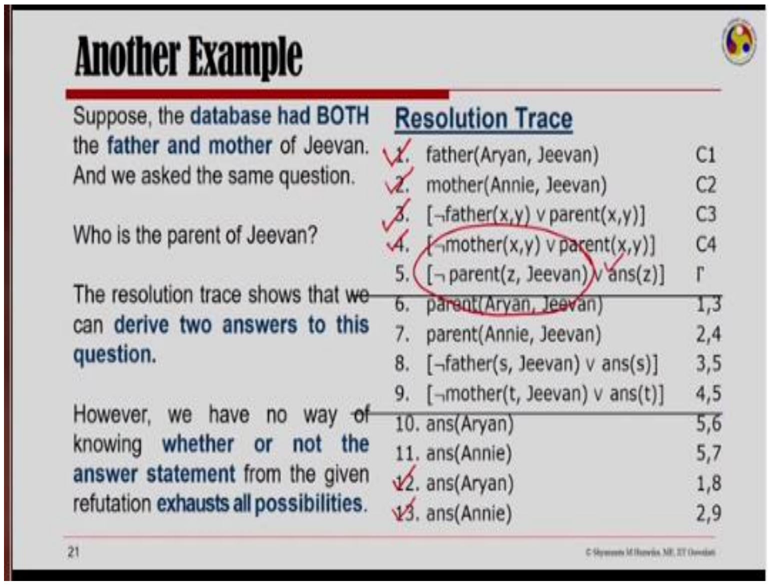
19 [illegible]

And thereafter I could do a resolution proof for this, so here is the resolution trace, where I have added gamma which is not only negation of the goal statement but also an answer literal. And I could resolve between 1 and 3 to have parent Aryan, Jeevan, I could resolve 2 and 3 to have parent Bhuban, Kavya. And then I could do a resolution between 3 and 4 to have not father X, Jeevan and the answer literal has answered X.

Now the resolution between these 2, here one needs to look at again see here I have X for Z and Jeevan for Y and that leads to not father X, Jeevan answer X. Now we could see that when I resolve 4 and 5, I have answer as Aryan or I resolve 1 and 7 I have answer as Aryan rather than the empty clause as would have come out if this was a resolution proof. The procedure here halts as soon as it derives a clause consisting of only the answer literals.

And this procedure produces only one answer literal if it does that the term it contains constitute the only answer to the question but this is not always the case.

(Refer Slide Time: 24:30)



Another Example

Suppose, the **database had BOTH** the **father and mother** of Jeevan. And we asked the same question.

Who is the parent of Jeevan?

The resolution trace shows that we can **derive two answers to this question.**

However, we have no way of knowing **whether or not the answer statement** from the given refutation **exhausts all possibilities.**

Resolution Trace

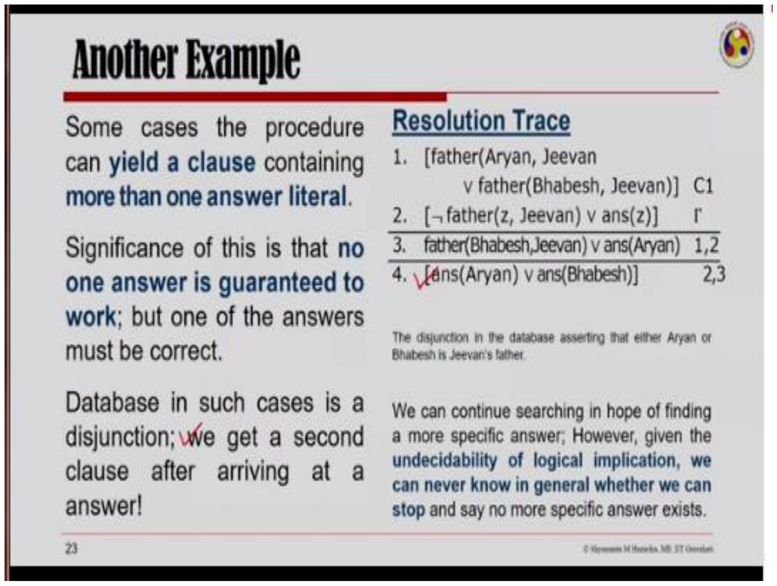
1.	father(Aryan, Jeevan)	C1
2.	mother(Annie, Jeevan)	C2
3.	[¬father(x,y) ∨ parent(x,y)]	C3
4.	[¬mother(x,y) ∨ parent(x,y)]	C4
5.	[¬ parent(z, Jeevan) ∨ ans(z)]	Γ
6.	parent(Aryan, Jeevan)	1,3
7.	parent(Annie, Jeevan)	2,4
8.	[¬father(s, Jeevan) ∨ ans(s)]	3,5
9.	[¬mother(t, Jeevan) ∨ ans(t)]	4,5
10.	ans(Aryan)	5,6
11.	ans(Annie)	5,7
12.	ans(Aryan)	1,8
13.	ans(Annie)	2,9

21

Suppose the database had both the father and the mother of Jeevan and we asked the same question who is the parent of Jeevan, in such a case the resolution trace if we see here can derive 2 answers to this question. One if I have a fact that any is the mother of Jeevan and another that Aryan is the father of Jeevan, I could have an answer that says Aryan and I could equally have an answer that says any.

Of course I have to have rules here saying every mother is a parent, every father is a parent and I am looking for the answer to equation which is who is the parent of Jeevan therefore I have added a answer literal. Now the resolution trace shows that we can derive 2 answers to this question, however we have no way of knowing whether or not the answer statement from the given refutation exhaust all possibilities, are there more answers there is no way of knowing whether there are more answers or not.

(Refer Slide Time: 26:02)



Another Example

Some cases the procedure can **yield a clause** containing **more than one answer literal**.

Significance of this is that **no one answer is guaranteed to work**; but one of the answers must be correct.

Database in such cases is a disjunction; we get a second clause after arriving at a answer!

Resolution Trace

1. [father(Aryan, Jeevan v father(Bhabesh, Jeevan)] C1
2. [¬ father(z, Jeevan) v ans(z)] Γ
3. father(Bhabesh,Jeevan) v ans(Aryan) 1,2
4. [ans(Aryan) v ans(Bhabesh)] 2,3

The disjunction in the database asserting that either Aryan or Bhabesh is Jeevan's father.

We can continue searching in hope of finding a more specific answer; However, given the **undecidability of logical implication, we can never know in general whether we can stop** and say no more specific answer exists.

23

© Shyamanta M Hazarika, ME, IIT Guwahati

Now instead of that if I have cases that have disjunctions in the database I can yield a clause containing more than one answer literal, significance of this is that no answer is guaranteed to work but then one of the answers must be correct. So let us try to understand this with an example here is an example, where I have a disjunction in the database. So the database here has a statement saying Aryan is the father of Jeevan or Bhabesh is the father of Jeevan.

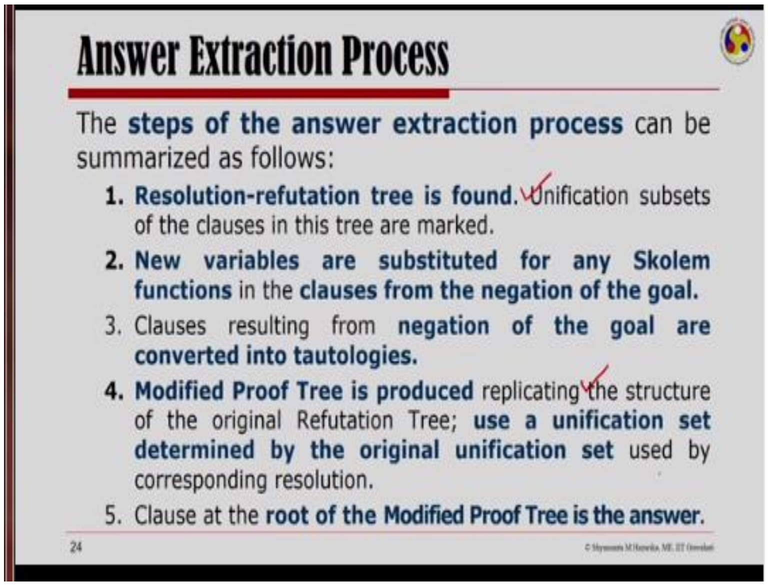
Now if I was looking for an answer who is Jeevan's father I would have added the negation of the goal which is not father Z, Jeevan and as I was doing fill in the blank I would have added an answer literal saying answer Z. Now given such a scenario, where I have a disjunction I can always get a second clause like let us say what we mean by a second clause after arriving at an answer, like if you resolve 1 and 2 then you could see that here Aryan would be for Z.

And these would go away that is the complimentary pair I would be left with father Bhabesh, Jeevan and my answer literal. I know Aryan has been substituted for Z. So I have an answer which says Aryan, so this is my first answer which says who is the father of Jeevan and that is Aryan. But now if you look at clause 3 and clause 2 you could see that you could resolve these clauses, here is my complementary pair this one and this one father Bhabesh, Jeevan not father Z, Jeevan.

So Bhabesh for Z, I can resolve this and then I would have an answer which would be Aryan or Bhabesh and this is something that one needs to understand that some cases the answer extraction procedure can yield a clause containing more than one answer literal. And such a case no one single answer is guaranteed to work but then we need to realize that one of the answers must be correct this is precisely because we get a clause second time after arriving at answer.

We can continue searching in hope of finding a more specific answer, however given the undecidability of logical implication. We can never know in general whether we have exhausted all possibilities, whether we can stop and say no more specific answers exist this is something that eludes us in answer extraction.

(Refer Slide Time: 29:52)



Answer Extraction Process

The **steps of the answer extraction process** can be summarized as follows:

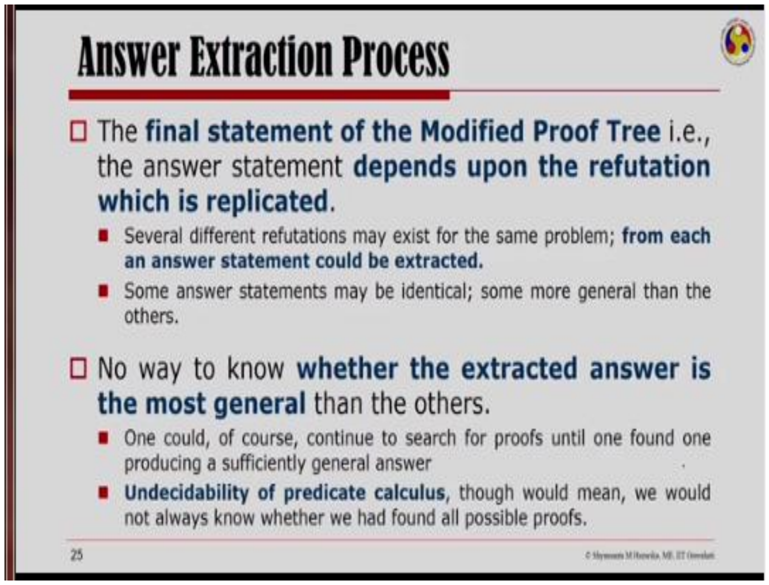
1. **Resolution-refutation tree is found.** Unification subsets of the clauses in this tree are marked.
2. **New variables are substituted for any Skolem functions** in the **clauses from the negation of the goal.**
3. Clauses resulting from **negation of the goal are converted into tautologies.**
4. **Modified Proof Tree is produced** replicating the structure of the original Refutation Tree; **use a unification set determined by the original unification set** used by corresponding resolution.
5. Clause at the **root of the Modified Proof Tree is the answer.**

24

So the answer extraction process that we have discuss today, to summarize the steps of the answer extraction process are the following. One first the resolution refutation tree is found, so here the most important fact to be marked is the unification subsets of the clauses. Because when I am trying to get the modified proof tree I have to go through the same resolutions that is I use the same unification subsets of the clauses.

Next if I have Skolem functions new variables are substituted in the clauses from the negation of the goal and clauses resulting from negation of the goal are converted into tautologies. Modified proof tree is produce replicating the structure of the original refutation tree, so the structure must remain the same and use a unifications set which is determine by the original unification set and therefore we have to mark the unification set of the clauses while we are doing the resolution refutation, clauses at the root of the modified tree is the answer to the question that we are looking at.

(Refer Slide Time: 31:26)



Answer Extraction Process

- The **final statement of the Modified Proof Tree** i.e., the answer statement **depends upon the refutation which is replicated.**
  - Several different refutations may exist for the same problem; **from each an answer statement could be extracted.**
  - Some answer statements may be identical; some more general than the others.
- No way to know **whether the extracted answer is the most general** than the others.
  - One could, of course, continue to search for proofs until one found one producing a sufficiently general answer
  - **Undecidability of predicate calculus**, though would mean, we would not always know whether we had found all possible proofs.

25

The final statement of the modified proof tree that is the answer statement depends upon the refutation which is replicated. Now, one needs to remember that several different refutations may exist for the same problem and from each and answers statement could be extracted. Some of these answers that I have extracted it may be identical some more general than the others. But then as I had emphasize we do not know if we could get a more general answer and we do not know when to stop looking for more general or more specific answers.

This is because of the very fact that predicate calculus is undecidable, so the undecidability of predicate calculus would mean that we would not always know whether we have found all possible proofs. But one could of course continue to search for proofs until one has found one producing a sufficiently general answer. So this is what we have covered in answer extraction, in our next lecture we would look at certain strategies that we use while we are working through resolution refutation proof systems, thank you very much.