

**Artificial Intelligence:
Incompleteness of Forward Chaining and Backward
Chaining**

Prof. Deepak Khemani

**Department of Computer Science and Engineering
Indian Institute of Technology, Madras**

Module - 07

Lecture - 01

So let's come back to the topic of theorem proving and remember we had these basic notions of entailment which says that a knowledge base entails a formula α . And then we had the notion of Proof which said that knowledge base can derive a formula α . And then we had the notions of soundness and completeness. In particular we are interested in completeness which says that if something is entailed then we should be able to derive it. The opposite is soundness which is that if we can derive it, it must be true or it must be entailed.

Now we have looked at two methods so far, we have looked at forward chaining and what forward chaining does is that given a knowledge base of some facts. So when we say facts I mean two sentences which in the sense of rule based systems means include both facts and rules but you can keep adding new things. In a forward chaining system which is data driven you have something in your knowledge base, you can make some inferences and then you can keep adding those inferences to the knowledge base so that the knowledge base tends to grow essentially and the hope is that eventually you will add the formula that you are looking for. If there was some formula called α that you wanted to show to be true then eventually that will be added essentially.

That's the notion of completeness essentially. In backward chaining you try to find rules which will make alpha true which means we have things like you try to look for rules of a kind Beta prime implies alpha prime and you try to unify this with this and then produce a subgoal beta or you can have another subgoal beta prime depending on how many rules you have. You try to work backwards from the thing you want to show and hopefully you will reach the set of facts that you have.

(Refer Slide Time: 03:40)

Theorem Proving.

Entailment $KB \models \alpha$

Proof $KB \vdash \alpha$

Completeness IF $KB \models \alpha$ THEN $KB \vdash \alpha$

Forward Chaining

Backward Chaining

$\beta \leftarrow \beta' \supset \alpha'$

α

KB

NPTEL

So our example was you have Man Socrates and Man X implies Mortal X and the query was Mortal Socrates. So in backward chaining you reduce this to Man. So if there is a query you reduce this query to Man Socrates and this formula will match this formula essentially.

(Refer Slide Time: 4:25)

krn-march4 - Windows Journal

File Edit View Insert Actions Tools Help

100%

3/4/2016

Theorem Proving. Entailment $KB \models \alpha$
 Proof $KB \vdash \alpha$
 Completeness IF $KB \models \alpha$ THEN $KB \vdash \alpha$

Forward Chaining

Backward Chaining

$\beta \leftarrow \beta' \supset \alpha'$
 β'' $? \text{Mortal}(\text{socrates})$

$\text{Man}(\text{socrates})$
 $\text{Man}(\text{?x}) \supset \text{Mortal}(\text{?x})$
 KB

$\text{Man}(\text{socrates})$
 $? \text{Mortal}(\text{socrates})$

NPTEL

1/1

In forward chaining you will add Mortal Socrates at some point may be other things you could have added first and hopefully when you add Mortal Socrates then you can terminate. So one advantage of backward chaining is low branching. And which is why very often people prefer to do things something like. But we saw in both the cases you may have to write the rules carefully.

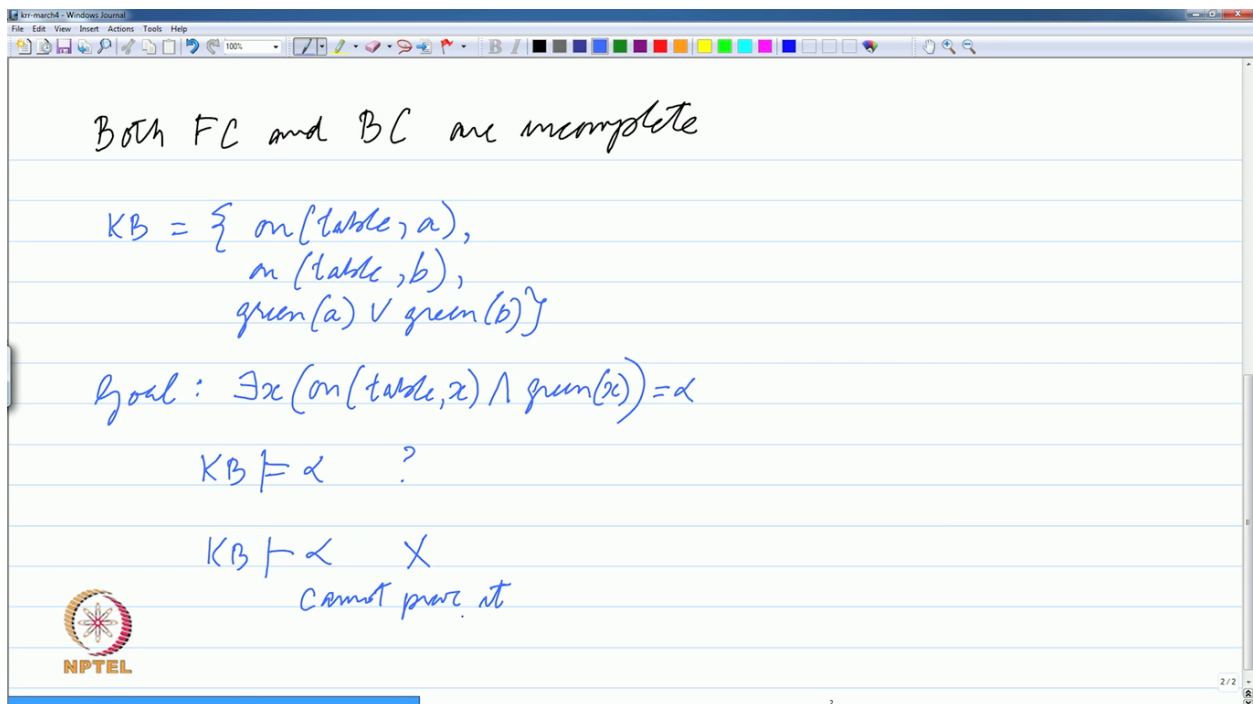
Now lets look at a couple of examples which show that both Cs are incomplete. So the first example is consider this set of formulas. We have very small knowledge base here. You have three statements, you are saying that a is on the table and b is on the table and one of them atleast is green, so green a or green b. and lets say our goal is to show there exists something which is on the table and which is green. So first of all lets call this formula alpha, we want to show that alpha is true. Alpha is a statement that there exists an alpha such that ontable x is true and green x is true.

So let's ask the question. Is this the case, is this statement entailed by the knowledge base. Does the knowledge base necessarily imply that there is something on the table which is green. The answer is yes because we know that there are these two blocks and we know that at least one of them is green so this statement that there exists something in the knowledge base which is green is true.

7:49

Long pause

(Refer Slide Time: 8:08)




Both FC and BC are incomplete

$$KB = \{ on(table, a), \\ on(table, b), \\ green(a) \vee green(b) \}$$

Goal: $\exists x (on(table, x) \wedge green(x)) = \alpha$

$KB \models \alpha$?

$KB \not\models \alpha$ X
cannot prove it

 NPTEL

2 / 2

But unfortunately this case doesn't hold, cannot prove it. Irrespective of whether you want to do forward chaining or backward chaining you cannot prove this fact. In fact there are hardly any rules in the knowledge base, in fact the best you can try to do is convert this into a rule. You could say not green a implies green b. you can rewrite it as implication and think of it as rule. But it doesn't help much really, because there is nothing to say that you know whether a is green or b is green, we know one

of them is green not both. So this is an example which illustrates that while forward chaining and backward chaining can work well in many situations, these are not methods which are complete and there are true facts which you cannot derive.

This example is from a book by Sharniac and McDevott. The second example that I am showing you is from this our book that we are following which is Reckman and Lev is essentially. So that was example 1. So example 2 is also from a blocks world kind of a domain and I have 3 block lets say A, B, C. and I know that this is green, and this is not green which ofcourse I can write as a knowledge base. On A B, on B C, green A, not green B. so I have these 3 blocks which are stacked up. A is on B, B is on C and I know that A is green and I know that B is not green. And my goal let me call it beta is there exists a x, there exists a y such that on x y and, may be I should use And symbol directly. And green x and not green y. So does KB2 entail beta?

What is the query? That is there a green block sitting on top of a non green block. So is it true?

(Refer Slide Time: 11:49)

Both FC and BC are incomplete

Ex 1

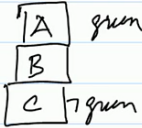
$$KB = \{ on(table, a), \\ on(table, b), \\ green(a) \vee green(b) \}$$

Goal: $\exists x (on(table, x) \wedge green(x)) = \alpha$

$KB \models \alpha$?

$KB \not\models \alpha$ X
cannot prove it

Ex 2



$KB2 = \{ on(A, B), \\ on(B, C), \\ green(A), \\ \neg green(B) \}$

$\beta = \exists x \exists y [on(x, y) \wedge green(x) \wedge \neg green(y)]$

$KB2 \models \beta$?

NPTEL

So A is green C is not green. Actually I drew the figure. I want to show that there is a green block lying on block which is not green. In fact Reckman and Lewis, use an example to show that entailment sometimes is not this obvious. But if you give it a thought you will see that this statement is indeed true because there are two possible cases: one is either B is green or the other is that B is not green. If B is green then B is on C A and C is not green. If B is not green then A is on C and it is green.

Again it turns out that we are not able to derive it and may be you can just give it a bit of try. So this gives us a motivation to look for another method and we have already seen this method except we saw it in the case of propositional logic. And its called the Resolution method. So just to highlight this fact let me just work with these two examples to start with. And then we will go to the details, revisit the algorithm in a little bit more detail. So lets take Example 1. What are the facts given to us? On table A. The other fact given to us is on table B. and the third fact given to us is one of them is green.

So you might recall that the resolution method requires the things to be in conjunctive normal form, all the formulas. In this case, all these three formulae are already in conjunctive normal form, they are very simple formulae so we can work with them directly. We will see later on, we will see the method of converting them. So this is given to us and the goal is that there exists x on table x , and green x . So if you remember the resolution refutation method we negate it, so which means we get this formula. Or let me use the original color to suggest that this is what we are working with. So you should remember that the resolution refutation method suggests to negate the goal and add the negation of goal to set of facts or the set of formulae that we have.

But we must first convert them into clause form. To convert into clause form we know that this is equivalent to for all x , I will push the not inside twice in one step, on table x or not green x . which when I convert into implicit quantifier form I get this not on.

So that is the clause that I add to the set. And I need to show that the set of four clauses that I have is unsatisfiable, and if you remember that in the resolution refutation method it amounts to being able to derive the null clause essentially. So how do we do that? So let's resolve this with this first. We have already looked at unification, so we will need to apply unification to make two formulae true. So given that this not green x we can match either with green B or green A , it doesn't matter so let's say we say that x is equal to B . then we will get not green B here or not on table B here. So when we resolve these two clauses we will get not on table B or green B . Just let me know if I am making any error. In a similar fashion we can also say that x is equal to A and we will get not on table A or green B .

(Refer Slide Time: 19:01)


Resolution Refutation is Complete!

Ex 1
 Given $\boxed{on(table, A)}$ $\boxed{\neg on(table, B)}$ $\boxed{green(A) \vee green(B)}$

Goal:
 $\exists x [on(table, x) \wedge green(x)]$
 \downarrow negate
 $\neg \exists x [on(table, x) \wedge green(x)]$
 \equiv
 $\forall x [\neg on(table, x) \vee \neg green(x)]$

Resolution process:
 $\neg on(table, A) \vee green(A)$ (from $on(table, A)$ and $\neg \exists x [on(table, x) \wedge green(x)]$)
 $\neg on(table, B) \vee green(A)$ (from $\neg on(table, B)$ and $\neg \exists x [on(table, x) \wedge green(x)]$)
 $\neg on(table, ?x) \vee \neg green(?x)$ (from $\forall x [\neg on(table, x) \vee \neg green(x)]$)

Resolution steps:
 1. $\neg on(table, A) \vee green(A)$ and $\neg on(table, B) \vee green(A)$ resolve to $green(A)$ (labeled $?x=A$).
 2. $green(A)$ and $\neg on(table, B) \vee green(A)$ resolve to $\neg on(table, B)$.
 3. $\neg on(table, B)$ and $\neg on(table, ?x) \vee \neg green(?x)$ resolve to $\neg on(table, ?x)$ (labeled $?x=B$).
 4. $\neg on(table, ?x)$ and $\neg on(table, ?x) \vee \neg green(?x)$ resolve to the empty clause.



Now using this and this one, because on table A is there we get green B. And if I use this and this by saying that x is equal to B again I will get not green B. and then from this and this I will get the empty clause. So this is one way of deriving empty clause. You see that we were successful in deriving empty clauses. And this is something we could not prove using forward chaining or backward chaining.

(Refer Slide Time: 20:44)

Resolution Refutation is Complete!

Ex 1

Given $m(\text{table}, A)$ $\neg m(\text{table}, B)$ $green(A) \vee green(B)$

Goal: $\exists x [m(\text{table}, x) \wedge green(x)]$

\downarrow negate

$\neg \exists x [m(\text{table}, x) \wedge green(x)]$

\equiv

$\forall x [\neg m(\text{table}, x) \vee \neg green(x)]$

Diagram illustrating the resolution process:

- From $m(\text{table}, A)$ and $\neg m(\text{table}, B)$, we derive $green(B)$.
- From $green(A) \vee green(B)$ and $green(B)$, we derive $\neg m(\text{table}, A) \vee green(B)$.
- From $\neg m(\text{table}, A) \vee green(B)$ and $green(B)$, we derive $\neg m(\text{table}, B) \vee green(A)$.
- From $\neg m(\text{table}, B) \vee green(A)$ and $\neg m(\text{table}, B)$, we derive $\neg m(\text{table}, ?x) \vee \neg green(?x)$.

NPTEL

So I will leave the second example I gave which was three stacked up blocks as exercise for you to work on to show that it can be shown in the refutation method. So one thing that we need is to convert formula into clause form. and when I say clause form I mean that the formula should look like this, for all x_1 , for all x_2 , for all x_n , and there should be a CNF formula here. So this particular form is called is called a clause form where there are only universal quantifiers all outside to the left and inside the bracket we have the formula in conjunctive normal form. there are no quantifiers inside, there are no quantifiers no implications, only the CNF form and or and not and everything should be in that form.

So one of the first things we need to do is to convert formulas into clause form. so lets look at a way of doing that. What is the algorithm for converting. So first step is 1. Take existential closure. So it basically means that if x is free then introduce there exists x . because we want to talk about sentences and we had said that formulas with free variables are not sentences so we will first convert it into a sentence by saying that whenever we have a

formula like with a free variable, so let's say x is greater than 13 for example, then what I really mean is that there exists an x such that x is greater than 13. So I will just add an existential quantifier for that particular variable essentially.

2. Then we have already seen this Standardised variables apart. Because one of the goals of converting to clause form is we don't want to handle quantifiers so once we have it in clause form and since all the quantifiers are outside the square brackets we can just throw them away. But if we do that we have to be careful that whenever we used two instances of the same quantifier we must use a different variable name. and that's what we had said when we were looking at forward chaining as well.

3. Eliminate all connectives and we have seen how to do that. And we will see one more example. The implication can be P implies Q can be replaced by $\text{not } P$ or Q . P equivalent to Q can be replaced by P implies Q and Q implies P and so on and so forth.

You push the negation inside because you have seen that negation can create havoc for example with the nature of variables. So if we have formula like there does not exist x which something something then you should push inside and convert it into a for all x formula.

Push the quantifiers also inside. So the scope of each quantifier is as tight as needed. This is again good for clarity.

Eliminate the existential quantifier, this we have already studied. This is a process of Skolemization. How to eliminate a existential quantifier so if you remember every existentially quantifier variable we either replace it with the skolem constant or with a skolem function of universally quantified variables. We will may be see an example.

Now that you have eliminated the existential quantifier you can bring all the universal quantifier outside without changing the meaning of the formula essentially.

Since we want it to be in CNF we have to distribute AND over OR. We will revisit it if necessary

Simplify. Sometimes you find that you have redundant set of formulae and you can simplify them to reduce the actual occurrences. And finally rename clauses.

(Refer Slide Time: 28:27)

Resolution Method - input = clause form.
 $\forall x_1 \forall x_2 \dots \forall x_n$ [CNF formula]

Conversion to Clause Form

1. Take existential closure of x to free - introduce $\exists x$
2. Standardize variables apart
3. Eliminate all connectives except \wedge, \vee and \neg
4. Move \neg inside
5. Push \forall and \exists to right (so that scope is as tight as needed)
6. Eliminate \exists - Skolemization.
7. Move \forall to the left
8. Distribute \wedge or \vee
9. Simplify
10. Rename clauses with different variable names:

NPTEL

4/4

Because you may end up using clauses again and again it is best to rename all the different clauses with different variable names. If you do this you will end up with a formula which is in clause form. so in the next class I will start with an example of converting a formula into clause form and then we will look at an example of resolution method again. And if we have time we will revisit the earlier example which we did not show.