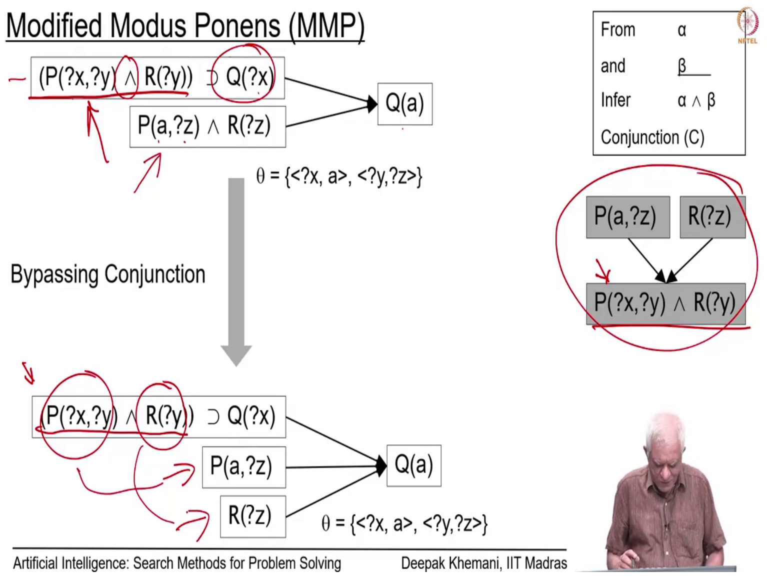
Artificial Intelligence: Search Methods for Problem Solving

Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 11
A First Course in Artificial Intelligence
Lecture – 85
Deduction as Search Backward Reasoning

(Refer Slide Time: 00:17)



Modified Modus Ponens (MMP)

From	α
and	β
Infer	α ∧ β
Conjunction (C)	

(P(?x,?y) ∧ R(?y)) ⊃ Q(?x)
P(a,?z) ∧ R(?z)
Q(a)
θ = {<?x, a>, <?y,?z>}
Bypassing Conjunction
P(a,?z)
R(?z)
P(?x,?y) ∧ R(?y)
(P(?x,?y) ∧ R(?y)) ⊃ Q(?x)
P(a,?z)
R(?z)
Q(a)
θ = {<?x, a>, <?y,?z>}
Artificial Intelligence: Search Methods for Problem Solving
Deepak Khemani, IIT Madras

So, welcome back. This week we have been looking at logic and in the last few videos we looked at the process of finding proofs through forward chaining. We have seen that when we talk about logic we want to talk about truth values, but truth values are not directly accessible to us because they will depend upon what is true in the domain. So, instead we rely on the process of making inference or deduction or proofs and we need to search to find proofs essentially.

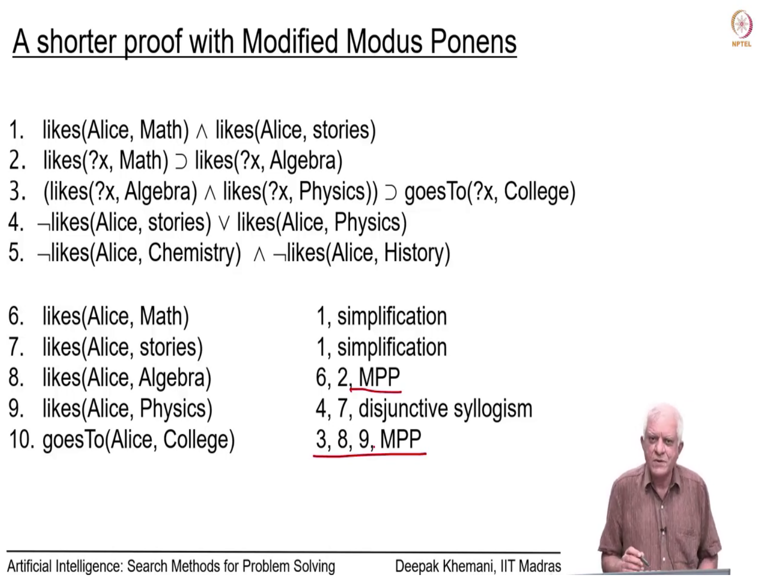
So, in this course our focus is on showing that underneath the logical reasoning process where you make inferences and jump to conclusions or draw conclusions, there is still a layer of search and in some sense that is a characteristic of intelligent behavior that it is not just flat behavior at one level, but there are layers upon layers upon layers.

So, for example, when we talk about city map route finding we can think of it as a at a higher layer asking a query give me a route or what is the best route from point a to point b and beneath answering that query there is search setting. So, search is in some sense fundamental to problem solving. And, now we are seeing how search plays a role in logical reasoning, in particular we are looking at deduction using first order logic.

And, so far we have seen forward chaining and in forward chaining we saw that we are given a set of facts which you can see at the bottom of the thing or you can see on the top here that there are two formulas and essentially you are applying something called modified modus ponens in which if you can match the given fact which is this one with the left hand side which is that one, then you can infer the right hand side.

And, we saw that we can extend the idea of modified modus ponens to say that those two facts which is a conjunction of two formulas can in fact, be allowed to be present individually and our rule will take care of the fact that it find both these components one is here and the other is here and makes the conclusion.

(Refer Slide Time: 03:04)



A shorter proof with Modified Modus Ponens

1. likes(Alice, Math) ∧ likes(Alice, stories)
2. likes(?x, Math) ⊃ likes(?x, Algebra)
3. (likes(?x, Algebra) ∧ likes(?x, Physics)) ⊃ goesTo(?x, College)
4. ¬likes(Alice, stories) ∨ likes(Alice, Physics)
5. ¬likes(Alice, Chemistry) ∧ ¬likes(Alice, History)

6. likes(Alice, Math)	1, simplification
7. likes(Alice, stories)	1, simplification
8. likes(Alice, Algebra)	6, 2, MPP
9. likes(Alice, Physics)	4, 7, disjunctive syllogism
10. goesTo(Alice, College)	3, 8, 9, MPP

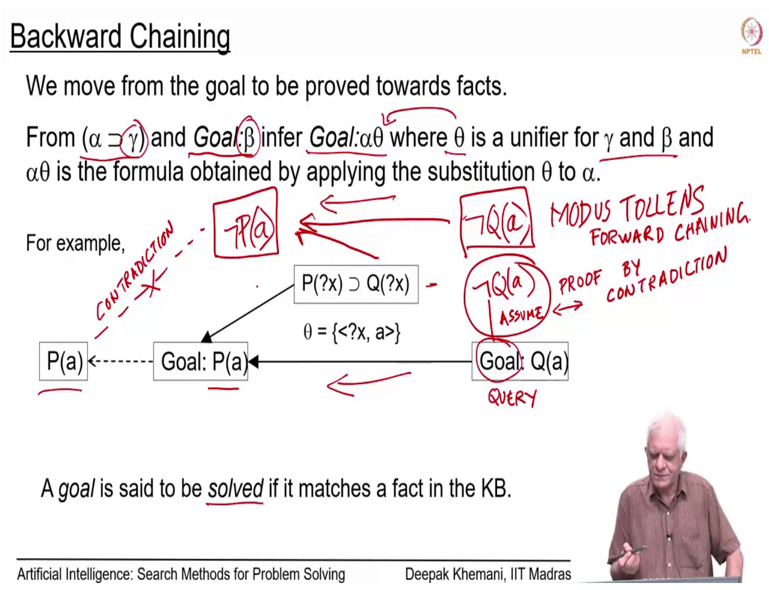
Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

And, then we saw that this gives rise to a much shorter proof for the problem that we have been looking at which is a problem of figuring out whether Alice will go to college or not and we had used the modified modus ponens as can be seen here essentially.

Now, we shift our attention to the other way of searching which is from the desired goals or queries towards facts essentially. So, we have already seen a flavor of this when we looked at planning we saw forward state space planning and we saw backwards state space planning.

So, this one is in some sense very similar to backward state space planning that you are moving from the goal towards the fact and asking whether the goal is true or not. And, then moving back over the rules of inference towards the facts and if you can reach a set of facts, then we can say that yes the goal is true. So, let us look at that process now.

(Refer Slide Time: 04:09)



Backward Chaining

We move from the goal to be proved towards facts.

From $(\alpha \supset \gamma)$ and *Goal*:β infer *Goal*:$\alpha\theta$ where θ is a unifier for γ and β and $\alpha\theta$ is the formula obtained by applying the substitution θ to α.

For example,

$$\begin{tikzcd} & & P(?x) \supset Q(?x) \arrow[dl] & \\ P(a) & \text{Goal: } P(a) \arrow[l, dashed] & & \text{Goal: } Q(a) \arrow[ll, "\theta = \{<?x, a>\}"'] \end{tikzcd}$$

A *goal* is said to be *solved* if it matches a fact in the KB.

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, if you remember in forward chaining, given this formula alpha implies gamma and given a fact beta, if you could unify alpha with beta remember the unification is the application of a substitution which makes alpha and beta the same. It basically means substituting values for variables then we could have moved forward to gamma and inferred gamma times delta which means you apply the same substitution not delta sorry theta the same substitution to gamma and that is your conclusion.

In backward chaining, we have to now add is include an additional class of formulas. So, in forward chaining you just have the knowledge base and you kept adding new sentences to the knowledge base so that the nature of the knowledge base was simply that it was a set of sentences and to that we kept adding new sentences which were the conclusions of the inferences that we are making or which are the deductions we were making.

But, now we are talking about goals or queries. So, we have a different class of formulas. So, we will just annotate them here with the symbol called goal to alert us that goals are different from facts. Goals are something that you want to show to be true or you are asking whether it is true whereas, facts are something which are known to be true or even theorems that you have proved.

So, to distinguish between facts and queries or facts and goals we use the annotation goal here. So, now, we are saying in backward chaining that if you are given a formula of the kind alpha implies gamma and you are given a goal beta and you can find a unifier theta which will make gamma and beta the same. So, beta is the goal that we are interested in and gamma is the consequent of the rule that we are hoping will be instrumental in justifying that beta is true.

So, if you can find the unifier theta which makes gamma and beta same, then we can apply the same unifier to alpha and we can say that instead of showing that beta is true we have reduced it to a sub goal which says that show that alpha is true; alpha with the theta applied to that essentially. So, we are moving from goals to sub goals to goals to sub goals and so on and that is the process of backward chaining.

So, remember that the goal is a different set of formulas of course, if you are implementing them you may you may just keep them in separate compartments so that you do not mix up goals with facts. But, as far as discussing is concerned we will use annotation goal in front of beta. Goal beta is not a sentence in our logic, it simply says that beta is a goal essentially and it is an annotation we are using.

So, this is the situation. You are given the universal formula remember that this is implicit quantifier form. It says all Ps are Qs or in other words for all x P x implies Q x and now, we are given some individual a; remember that a is a constant here in the language, x is a variable, a is a constant and we are asking whether Q of a is true or not essentially.

In forward chaining, we would have simply kept adding new formulas to the knowledge base until Q of a was added. But, now we are starting with Q of a and going in the backward direction and saying that is this is this formula true or not. So, we refer to Q of a also as a query in the style of databases that you are asking whether something is true or not. And, what happens in backward chaining is this process of moving as we just described above from the query Q of a to the query P of a.

So, when we were we started with the question that is Q of a true backward chaining says that there is a rule which says that all Ps are Q and therefore, let us use that rule chain backwards. So, remember the arrows are pointing from left to right here and generate a sub query and the sub queries is P of a true. Now, a goal like this is said to be solved if it matches some fact in the knowledge base.

So, if we have a knowledge base we have p of a as given here then we can say yes, the goal is solved essentially. So, this has a flavor of goal trees that we saw and we will see now that indeed there is a lot of similarity between goal case that we studied earlier and backward chaining process in logic essentially. So, we are moving from Q to P.

So, one should not confuse this with the rule of modus tollens which let me try to insert here would have said something like this that if you have naught of Q a, then from this and this you can infer naught of P a. This was the rule of modus tollens. We have not paid too much attention to individual rules of inference, but hopefully you remember modus tollens. It is one of the more common rules used in logic.

The reasoning in modus tollens is still forward. This is a fact naught of Q of a is a fact the rule is given to you it is also a sentence in the knowledge base and we sometimes differentiate between rules and facts, but it is a sentence in the knowledge base.

So, in that sense it is a fact and from there we infer a new fact not of P a modus tollens is forward chaining whereas, backward chaining goes from query to sub query they are two different things of course, backward chaining ends if the goal is so primitive or solved, such that it is there in your knowledge base automatically.

So, first of all you must distinguish between these two things modus tollens is different from backward chaining. Modus tollens uses for forward reasoning or forward chaining to move from a fact which is that Q of a is false or not Q of a is true to another fact which says that P of a is false or naught P of a is true that is a forward reasoning process. But, that is the interesting part coming up now.

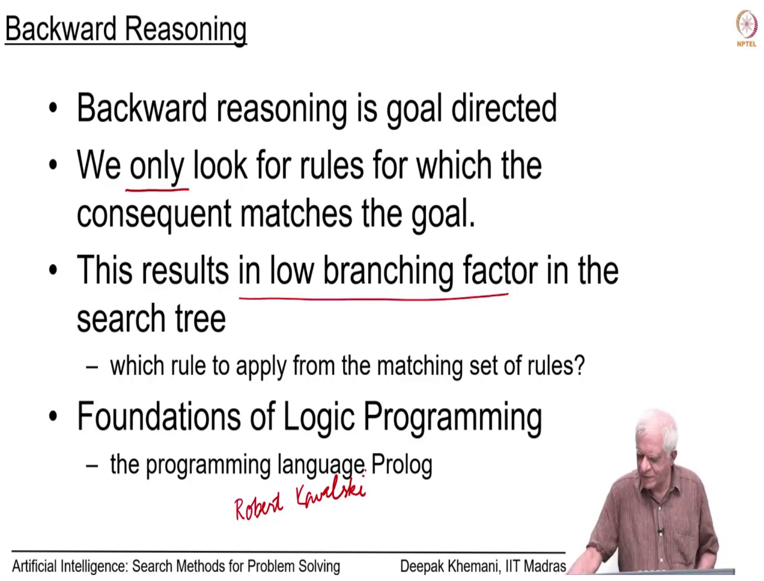
What if we think of this goal annotation that we have done to stand for negation essentially then this formula goal Q of a becomes the formula Q of a that we are talking about essentially. So, I hope that reminds you of one proof procedure that you must have studied at some point which says that proof by contradiction.

How do you do proof by contradiction? You are given some set of statements or premises or axioms or facts as we call it a knowledge base and you are given a query show that something is true.

In this case we want to show that Q of a is true and in proof by contradiction we say let us assume that Q of a is false. So, this is an assumption. And that is a process of proof by contradiction that you assume that what you want to show is false and then show that that leads to a contradiction.

So, if you take now naught Q of a as a fact because we have assumed that it is true, we now go back through this process of modus tollens to come to this. What have we shown? That naught of P of a is true or P of a is false, but you can see here that this is a contradiction. So, if you assume that Q of a is false you end up showing that P of a is false and that leads to a contradiction and then you go back and say yes, that assumption is wrong and therefore, Q of a is true.

(Refer Slide Time: 14:20)


Backward Reasoning

- Backward reasoning is goal directed
- We only look for rules for which the consequent matches the goal.
- This results in low branching factor in the search tree
  - which rule to apply from the matching set of rules?
- Foundations of Logic Programming
  - the programming language Prolog

Robert Kowalski

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, if you give some thought to this so, backward reasoning is goal directed we move from goal to facts. We only look at rules for which the consequent matches the goal for the same reason why we said that backwards state space planning has a small branching factor for in a similar way backward reasoning or backward chaining also has a small branching factor because you are only interested in showing that this particular goal is true essentially.

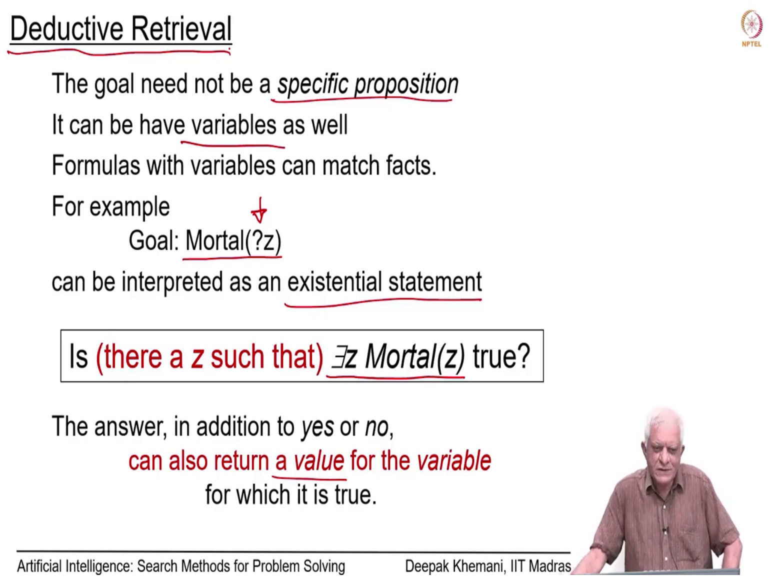
Whereas, in forward chaining you keep indiscriminately adding new facts to your knowledge base and hopefully wait for the time when your goal is added to the knowledge base. So, this indiscriminate addition of facts can lead to very high branching and unless you have some means of deciding as to which branch is better you are going to end up doing a lot of work in that.

So, backward chaining leads to low branching factor and in the search tree and the search tree is the tree in which you have to choose between which rules to apply essentially. Now, this idea was put forward by Robert Kowalski, whose name we have mentioned earlier.

He was the person who said that the program is equal to logic plus control and that the user or the programmer should only have to specify the logic and the control is left to some inference engine. We looked at that process, but we looked at forward chaining at that point and we looked at rule based production systems.

Now, we are moving towards backward chaining and the same process in which the logic is specified in a declarative fashion and some influence engine or some search program takes care of finding the proof essentially and he was one of the two people who devised this language prologue which is based on the idea of logic programming essentially. And, that is what we are going to see very quickly in the rest of this week I think.

(Refer Slide Time: 16:28)



Deductive Retrieval

The goal need not be a *specific proposition*

It can be have variables as well

Formulas with variables can match facts.

For example

Goal: Mortal(?z)

can be interpreted as an existential statement

Is (there a z such that) $\exists z$ *Mortal(z)* true?

The answer, in addition to *yes* or *no*,

can also return a *value* for the *variable*

for which it is true.

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

Now, one interesting thing about backward chaining or backward reasoning which is goal directed that you are focused on the goal is that you do not have to necessarily ask for a specific proposition we are not we are not saying whether Q of a is true you can ask is there some element for which Q is true and that kind of a thing.

So, we can have variables as well and of course, formulas with variables can match facts. So, if you have a goal which says mortal z with a variable so, observe that there is a variable here. This goal with a variable can be treated as an existential statement.

So, this is something which again you should be aware of that when you are talking about goals remember that we had partition the formulas into facts and queries and facts and goals. In fact, if you use a question mark with a variable we have a universally quantified variable which is a universally true statement. In goals this process is reversed if you have a question mark in a variable then you should treat it as an existential variable.

So, if you have a goal which says mortal z, where z is a variable it is asking two things. The first thing is what is written in the black here that is the formula there exists z mortal z true it is an existential statement that is the first thing we are asking from the logic perspective, but from the literal perspective it is also asking is there such a z which will make this fact true.

And, the nice thing about backward chaining or backward reasoning is that indeed when we run this query with our Socratic argument which said that all men are mortal and if you have at least one man in your knowledge base, then it will return yes there exist somebody who is mortal. And, not only that it will tell you as to who is it who is who you are talking about when you are giving us this proof, yes.

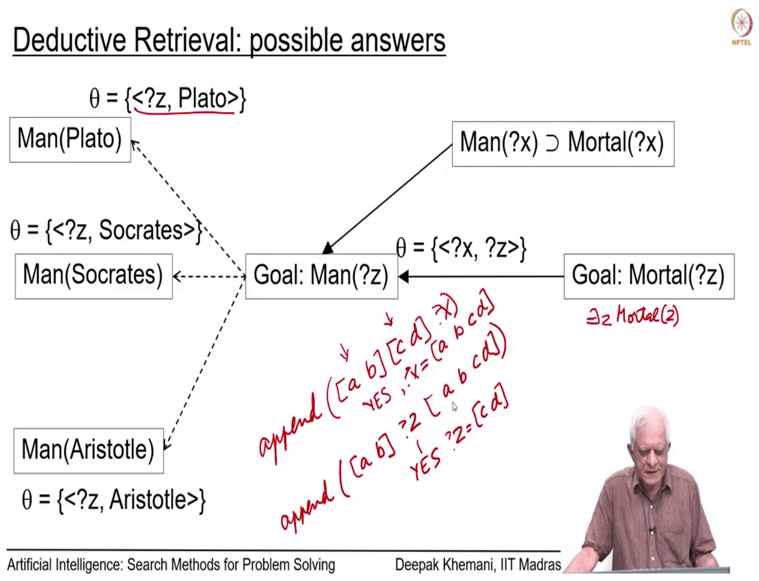
So, again you can see that this has a flavor of databases. So, if you have looked at relational databases and the corporate world is full of such databases you often ask a query is there an employee who has worked for 3 years and who has got two increments and who has got so much something and so on and the database of course, retrieves it.

Backward chaining extends the capability of a database to not only answer questions which are present in the database answer facts which are present in the database, but also those facts which can be concluded by a process of deduction. So, that is why we call this process as deductive retrieval. You can extend retrieval to the process of making deduction as well essentially.

And, it turns out that once we do that once we add this capability of doing reduction logic can be seen as a programming language and that is what led to the idea of logic programming.

And, not only is it a programming language it is a language which is turing complete as well and anything you can do in any programming language you can do in logic. Unfortunately, in this course we do not have so much time to look at the foundations of logic programming, but this much is enough to start with.

(Refer Slide Time: 20:32)



Deductive Retrieval: possible answers

Artificial Intelligence: Search Methods for Problem Solving Deepak Khemani, IIT Madras

So, let us look at this process of deductive retrieval again. This is our favorite statement all man all mortal expressed in implicit quantifier form and we also have a query which says is there someone who is mortal; expressed as a sentence mortal z where z is a variable essentially. Again, of course, by now you have figured out that z in the goal is an existential variable and x in the fact is a universal variable.

So, when we say all men are mortal we are talking about every element in the domain and if for every element if x is a man then x is mortal. When we talk about is there someone who is mortal then we are asking is there at least one person whose mortal which as we just saw is equivalent to asking the question is this following statement true there exists z such that mortal z. So, we are asking an existential query and we are talking about deductive retrieval now.

So, of course, we can now go back and do our process of backward chaining and when we chain backward chain over this rule you come to the sub goal which says that is there someone who is a man essentially and then if in your knowledge base you have a statement which says that Plato is a man then by saying that make z equal to Plato which is done by this substitution here, you can say yes that the query is true that there is somebody who is a man and not only that in fact, that someone is Plato.

You can also return yes with Socrates if man Socrates is there in your knowledge base and you can also answer it yes with Aristotle if that is totally your knowledge base. So, you can see that it is a deductive query process yes I asking is there a mortal person in my knowledge base it says yes, Plato and then if you have worked with prolog anytime you can ask for more answers. So, the second time it will say yes, Socrates is mortal; third time it will say yes, Aristotle is a mortal and so on and so forth.

So, that is an additional power that you get out of backward chaining is that you can ask existential queries essentially. You know you can ask queries like this. I will just leave it as a teaser for you to look at we will not go into the details here.

Is that is this formula true? Where append is a predicate which becomes true if when the first argument which is a list containing a and b, appended to the second argument which is a list containing c and d and x is just a variable and, now we are asking is this formula true.

Then, if you have written a prolog program which is a program to do append which incidentally is only two lines prolog will come back and say YES and not only that it will say x equal to a list a b c d. So, in effect you have a program to append two list.

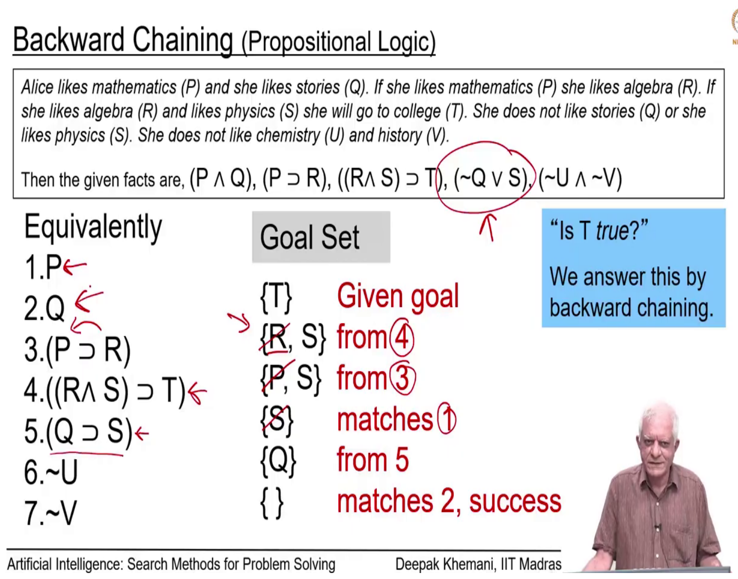
The nice thing about prolog is that you can even ask a query like this or at least pure prolog that if I take this list a b and append it to some list z can I get a b c d again prolog will search through the possibilities as I said there is search underlying reasoning and come back with an answer YES and it will say that s z is equal to a list made up of c and d.

So, there is a lot of power in logical deduction and that is the foundation of this whole idea of logic programming that logic and reduction can be used as a programming language. We can we have seen here that we can you know do append of course, I have not given you the program maybe as an exercise that you should try to write it.

The program has basically two clauses one is the base clause which says that an empty list appended to any list will give you that same list back and the other one is the recursive clause which says that if you have a list let us say with two elements and two elements like shown here which will give you an list of four elements.

Then, if you want to take a list of two elements and three elements you will get a list of five elements or if you take a list of three elements and two elements you will get a list of five elements and that gives us the basis for recursion. So, from one list we keep taking out elements till we come to the empty list clause. So, eventually the proof process will find that, but anyway this is just something which will hopefully motivate you to look at prologue in a little bit more detail.

(Refer Slide Time: 26:30)



Backward Chaining (Propositional Logic)

Alice likes mathematics (P) and she likes stories (Q). If she likes mathematics (P) she likes algebra (R). If she likes algebra (R) and likes physics (S) she will go to college (T). She does not like stories (Q) or she likes physics (S). She does not like chemistry (U) and history (V).

Then the given facts are, $(P \wedge Q)$, $(P \supset R)$, $((R \wedge S) \supset T)$, $(\sim Q \vee S)$, $(\sim U \wedge \sim V)$

Equivalently
1. P
2. Q
3. $(P \supset R)$
4. $((R \wedge S) \supset T)$
5. $(Q \supset S)$
6. $\sim U$
7. $\sim V$

Goal Set	
$\{T\}$	Given goal
$\{R, S\}$	from 4
$\{P, S\}$	from 3
$\{S\}$	matches 1
$\{Q\}$	from 5
$\{ \}$	matches 2, success

"Is T *true*?"

We answer this by backward chaining.

Artificial Intelligence: Search Methods for Problem Solving   Deepak Khemani, IIT Madras

Let us get back to the process of finding proofs and we will start with our favorite problem which is the Alice problem and we will look at the proportional version to understand what the how back backward chaining works and then we will extend it to first order case.

So, to remind you this is the same problem that we have that Alice likes mathematics and she likes stories, if she likes maths, then she likes algebra, if she likes algebra and she likes physics, then she will go to college and a couple of other facts which of course, expressed in proportional logic are given here as a set of sentences P and Q, P implies R and so on.

Now, we want to show that T is true that is a query that we have and we want to answer this by backward chaining essentially. So, this is our knowledge base which is expressed as a set of facts or clauses as we sometimes call them. Only difference you might see is that we have replaced not Q or S with Q implies S and if you remember your rules of substitution you will recall that they are equivalent and you could have place it at any time.

We have chosen this particular format because you know we just want to work with the implication and it kind of makes backward chaining process looks more intuitive essentially. Though we could have used that naught Q or S also here. So, backward chaining works with the set of goals that you want to be true.

So, you maintain a set of goals and say that these are the goals that you want to be true or these are the goals that you want to be solved if you use the terminology of goal trees that we had seen earlier and initially we start off by putting the goal that we are interested in which is the statement T; T if you remember stands for Alice will go to college. And, that is a given goal to us then we do backward chaining we look at this rule this says from 4 right and what is 4? 4 is here.

It says that R and S implies T. So, from the goal T, we have moved to two goals R and S this notation is a set notation for these formulas and this basically says that there are two goals R is a goal and S is also a goal essentially. So, that comma represents and in this case and we will adopt this procedure by which we are so familiar with now.

We will always take the first element in any set essentially or I mean if you are representing this as a list you would just take the head of the list. So, we will always address the first goal first sub goal that is to be taken care of and that in our case is the goal R.

So, our goal is to try and make R true and how does that happen? It happens because statement 3 says that P implies R and from R we can backward chain to the goal P. So, we have removed R from our goal set and added P instead to the goal set. Why because we could move over rule number 3 in a backward direction and now, we have the goal P and goal S.

Then we look at the first goal remember we are taking the first one at each time and that is present in statement 1 itself. So, we can cancel one because there is P already in 1. So, that is solved trivially and we are left with one goal which is S. How do we solve for S we have in our statement 5 that Q implies S. So, we remove S and we add Q to our set of goals. We would have done the same thing if we had used the disjunctive form of the formula.

And, we would have instead of using modus ponens we would have used disjunctive syllogism, but you know just to make life simple we have converted in into we have converted naught Q or S into Q implies this and now we have this goal Q.

What about Q? Q is present in our knowledge base. So, we are done with this. So, it matches us fact 2 and we have the empty goal set which means you have solved the problem that you wanted to solve there are number goals to be achieved. So, we have shown that T is true essentially.

We will take a break now and come back and look at the first order logic version of the same formula and look at how backward chaining results in our old friend the goal tree or the android tree and how that a particular way of searching the tree is the basis for the language prologue essentially.

So, we will do that in the next session.