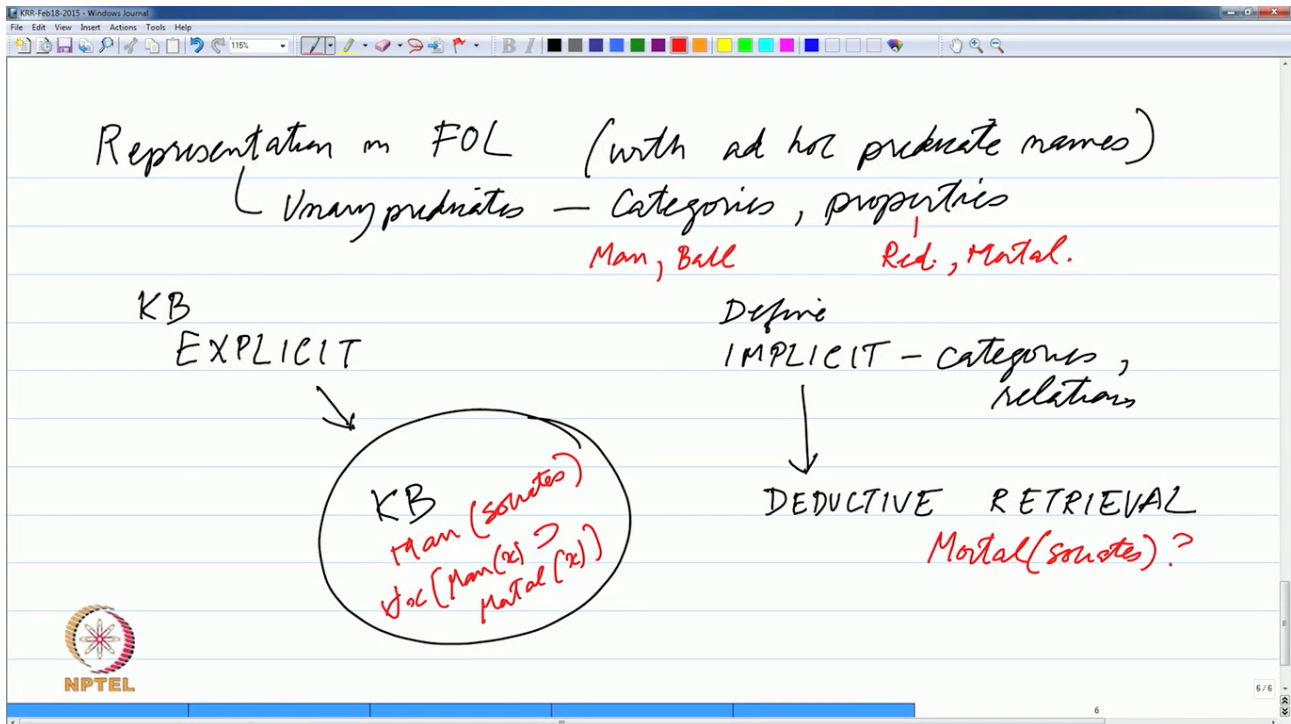


Artificial Intelligence:
Knowledge Representation in First Order Logic
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Module - 04
Lecture - 02

okay so we are looking at representation and we had stated when we were looking at reasoning that very often we work adhoc predicate names and we will see as we go along that if you really want to represent really complex things then you have to be little bit more careful than choosing adhoc predicate name. But lets stick with adhoc predicate names today and try to see what are the kinds of things that we can represent essentially. So we had said that unary predicates represent categories. Thats one of the things we said and we also said that they also represent properties. So categories for example Man, Ball, and so on and properties like Red, Mortal and so on. Everything we said we will represent using unary predicates. So lets see that even if we do that what are the kinds of things that we can express. So there is an idea in FOL that we have some things are stated explicitly, so a knowledge base for example may be expressive, its a collection of statements that we are making there may be implicit. Implicit could be categories or relations and so on.

So the whole idea here is that what we have in the knowledge base is an explicit statement of facts but once we know the facts there are certain other statements which can be implicit and we can do what Charniak and McDermott had called it as Deductive Retrieval. So to go back to the standard example we have been looking at. If the knowledge base has this Man Socrates and for all x Man x implies Mortal x. We can ask a query is Mortal Socrates. So to retrieve the answer to that we may need to do some deduction on the way and thats why its called deductive retrieval. So we may need to apply modus ponens and say that okay since all men are mortal and Socrates is a man yes Socrates is mortal. So this is an implicit fact its not stated explicitly. All that is stated explicitly is that Socrates is a man and all men are mortal and we want to essentially build systems which allow you to do deductive retrieval.

(Refer Slide Time: 04:24)



and we will see this can also allow us to ask a query like this that is there someone who is mortal? But keeping in mind that predicates represent either categories or properties for example if they are unary in nature or if they are binary they express relations between categories. So what are kinds of things that we can define explicitly and what can we define implicitly. So we can call upon what we sometimes call as Terminological Facts. How can we define a set of categories and how can we define more categories in terms of those categories. So let us say that we have these things explicit so in some sense I am giving you schemas for all the predicates. So we have as schema which says Man so in the style of databases we use the term man inside. So it takes one argument and the schema name is Man. So the meaning is since this is an adhoc predicate it basically is meaning as we understand among ourselves.

So this is also something you must always keep in mind. Whenever we use names like man, mortal so on and so forth, they make sense to us because you know they have an English meaning which we associate with it but in the language that we are representing it is just a subset essentially just a category. It does not have any connotations about man so if we say Socrates is a man it doesn't mean that the system knows in some sense that Socrates has got two eyes and Socrates has got two legs you know that kind of stuff nothing. All you know is that its the subset of the domain essentially, but anyway given a set of explicit sets like this say Man, Woman, Parent. So anyway we have a kind of a database in which we know who is a man or who is a male, who is a female, and who is a parent of who and may be one more fact which is Married so spouse1, spouse2. We have this in our knowledge base or database essentially so all are set. These are a set of statements. So you know lets say about a hundred people or something. All of them you know what are the in terms of these four relations which of them is male which of them is female, who is a parent of who and who is married to who. Supposing you know all this stuff we can now talk about implicit statements. So we can for example define the notion of a Mother. How do we do that so we want to be able to say what is the meaning of Mother. So we can say something like this for all x for all y Mother x y is equivalent to saying that Parent x y because the Mother is also a Parent and Woman(x).

so given the explicit predicates Man, Woman, Parent, Married, we can introduce a new predicate called Mother and maybe the language that we are talking about says that okay Yashoda was Ram's mother and so on so forth and if you want to listen to such stories we should be able to make sense of what do we mean by a Mother and here is a way of defining what is it. You can even define a category called Mother so obviously if you look at the set of human beings for example then there are some of them which are mothers so how do we define them. Any suggestions? I want to say for all x, x is a Mother is equivalent to saying that within the scope of the universal quantifier remember, there exists a y Mother x y. So we can say that somebody is a mother if she is a mother of someone essentially. And we are overloading the predicate name but we assume that that's only for our benefit. But as far as the system is concerned you can distinguish it by the fact that one of them has arity 1 and one of them has arity 2. if it is arity 1 and this one has arity 2. they are different relations in the domain. They are different relations in the domain, they just happen to be calling both of them as Mother essentially.

(Refer Slide Time: 10:35)

TERMINOLOGICAL FACTS

Explicit : Man (man), Woman (woman), Parent (parent, child),
 Married (spouse 1, spouse 2)

Implicit : $\forall x \forall y [Mother(x, y) \equiv (Parent(x, y) \wedge Woman(x))]$
 $\forall x [Mother(x) \equiv \exists y Mother(x, y)]$

arity 1 arity 2

NPTEL

7/7

so once you have defined Mother we can define other things, we can define for all x for all y Grandmother. We can define a version of Grandmother, how do we do that? A Grandmother is the Mother of a Parent essentially if we can remember that then we can simply write that. There must ofcourse be a third person involved now. We are talking about the relation between two people x and y but there must be a third person who stands between them in some sense. So you want to say that there exists a z such that Mother. We can reuse Mother now that we have defined it, x, z and you don't know if that z is a boy or a girl or man or a woman so we use Parent x y. So we can define new categories. So as an exercise you can say Define brother, sister, uncle, aunty, grandparent, cousin, niece, and so on whatever relations you can think of you can define them.

So the important point here is that our knowledgebase or database need not store who is who's brother and who is whos sister and so on. Once we define the four explicit predicates that we have. If we have data represented in terms of those four explicit

predicates we can answer queries about who is who's brother, who is cousin brother and so on. So if we ask a query is Suresh a brother of Sheela then ofcourse we can do some theorem proving to see that statement is true or not. And that will essentially use all these definitions that we are using.

What's more interesting is that you can define the notion of Ancestor. So we want to say, we want to be able to define Ancestor. Now defining Mother for example or Grandmother or even Great-grandmother for example is a straightforward process because we know how many people are in between. That between a person and his grandmother there is one parent sitting. Or between a person and the great-grandmother there is a parent and a grandmother or grandfather sitting. But when you define Ancestor which is little bit more vague how can we do that? And it can be done. So I want to say you must think recursion here. You can use recursion and whenever we talk about recursion we talk about a base clause. So we can say first thing is that our notion of Ancestor says that if x is an Ancestor of y it could be that x is the Parent of y so a Parent is also an Ancestor, immediate ancestor so but its ancestor.

But ofcourse what about other ancestors. Here we can say so ofcourse we must use for all x for all y. Otherwise how do we define an ancestor, we can say that Ancestor x y is equivalent to saying now thing the definition of Grandmother and we can use something similar. We can say that there exists a z and there are various choices here. At some later time we will consider which choices are better choices in terms of efficiency but right now at this moment we are not really bothered about that. So we are saying that okay x is an ancestor of y if there is a z such that x is a Parent of z and z is a Ancestor of y. So thats the other possible definition. So either you are an immediate parent which ofcourse is available in the database or you can use this second equation or second logical equation to say that okay you must be having the child who is an ancestor of, x is an ancestor of y, if x is a child of z and z is the ancestor of y. So eventually this will work essentially. And how do we combine this in terms of logic. We simply put disjunction between them and close the whole thing

(Refer Slide Time: 16:25)

The screenshot shows a Windows Journal window with the following handwritten text:

$$\text{Ancestor}(x, y)$$

$$\forall x \forall y \left[\begin{array}{l} \text{Ancestor}(x, y) \equiv \text{Parent}(x, y) \vee \\ \text{Ancestor}(x, y) \equiv \exists z [\text{Parent}(x, z) \wedge \text{Ancestor}(z, y)] \end{array} \right]$$

The window also shows a toolbar with various drawing tools and a playback control bar at the bottom with a progress indicator and a logo.


so for all x for all y either Ancestor x y is equal to Parent x y, well equal is not the right thing we should say equivalent to Parent x y or Ancestor x y is equivalent to saying that there is a child whose Parent you are and who is an ancestor of y. And as an exercise you should be able to see that just create a small database or few people and you would be able to answer this query using these inequalities. Remember that these inequalities are, not inequalities, these equivalences are basically two way implications. So whenever we write this it is equivalent to writing this and this essentially.

so I can write a statement like this or a statement like this. So a equivalence b is equal to saying a implies b and b implies a essentially. So you can really break it up into implications and you can use Modus Ponens and forward chaining, whatever algorithm that you are using essentially. So lets since we are computer science students lets take a small exercise and try to define a connected graph. We want to be able to say that a graph is connected. I will just outline the process here and leave it as a small exercise for you to do. Now if you remember what is a graph? A graph is basically a collection of nodes and edges and nodes is basically a set n_1, n_2, n_k . And edges is another set is also formed for example if i say e_{ij} then remember these are all the terms in a language so they denotes objects in a domain. So e_{ij} I could say is equivalent to saying that there is a function called Edge between n_1 and n_j . So an edge is basically a pair or a pair of nodes. Or you can explicitly call it an edge and then so on and so forth,

(Refer Slide Time: 19:19)

$Ancestor(x,y)$
 $x \neq y \left[\begin{array}{l} Ancestor(x,y) \equiv Parent(x,y) \vee \\ Ancestor(x,y) \equiv \exists z [Parent(x,z) \wedge Ancestor(z,y)] \end{array} \right]$

Exercise: Define a connected graph. — $\langle N, E \rangle$
 $\{n_1, n_2, \dots, n_k\}$
 $\{e_{ij} = Edge(n_i, n_j)\}$



and a graph is a term which will stand for, which is defined over a set of nodes N, E . Remember all these are terms essentially including Graph is a term, its an object in the domain essentially. And what you want to define is something called Connected. You want to define a predicate called Connected, we want to be able to say that the Graph is connected essentially. How do we do that?

We can replace this definition of Connected by saying there is a path between every pair of nodes. So you can see that this part, every pair of nodes, now you can see $n_1,$

n_2 , this is also a notation saying for all n_1 , for all n_2 , that for every node, which are these nodes? So these nodes must belong to the such that n_1 belongs to N and n_2 belongs to N . Remember N is the set of nodes that we have. I have written belongs to you could have chosen a more logical looking predicate, lets say $\text{Belong } n_1 \text{ comma } B$. Its the same thing, since we are more comfortable with this. So for all this we can say that there is a path n_1, n_2 . Any idea how to, now we need to define the predicate path, any suggestions? How can i define that there is a path from n_1 to n_2 . So essentially when I say define I am expressing a logical statement which will be true whenever there is a path. So i say $\text{path } n_1 \text{ } n_2$ means, in the domain which is in the graph which is a collection of nodes and edges, if I was to start from n_1 I will find a path to n_2 , which means I will find a sequence of edges which will go to n_2 . So any suggestions how?

Ok good. So thats like defining the Ancestor. Just as we defined Ancestor we can define Path. We can say that you have a path from n_1 to n_2 if there is an edge between n_1 and n_2 . In the other clause, recursive clause would be that there is a path between n_1 and n_2 if there is a third node such that there is a node between between n_1 and n_3 and there is a path between n_2 and n_3 . Which is analogous to this notion of defining Ancestor essentially.

Then we can define other kinds of properties. So we are talking about Terminological facts. We can define more properties about the domain and these examples that i am taking are from the book Reckman and Lewis which is one of the textbook that we are following for this course. So you can see for example Disjointness, so you can say for all x $\text{Man } x$. In other words we are saying that the set of Man and Woman are disjoint. We can talk about Subtypes. So you can say for all x $\text{Surgeon } x$ implies $\text{Doctor } x$. So Surgeon is a subtype of Doctor. And a little bit later in the course we will have a lot to say about this because when we talk about ontologies and description logics they are essentially talking about this kind of relation. One of the very important relation is the subtype relation essentially or subsumes relation. A set of Doctors contain a set of Surgeons and so on and so forth.

But we can express this in first order logic by a simple statement which say that All Surgeons are Doctors essentially. And we can say that our definition is Exhaustive, we can say that for all x if one is an Adult implies you must be either a Man or a Woman.

Then we can talk about Symmetry. We can make statements like. So notice that all this is trying to create a knowledge base in which the explicit facts will be minimal whereas the inferred or the implied facts will be a lot essentially. So whenever you say that Suresh is married to Sheela, you can also say that Sheela is married to Suresh. But since we are here dealing with all syntactic entities we have to do this explicitly. I mean for us it may be something obvious but not for a logic machine. Then we can have Inverses, for all x for all y .

Then we can restrict Type. Type is another word used for category for example. So we could say for example, if two people are married then they must be humans or persons. So in some sense we are saying only people can be married.

Then we can have Compound definitions. So we can say for example, for all x $\text{Richman } x$ implies $\text{Rich } x$ and $\text{Man } x$. In fact you can use an equivalence sign here. So we can define compound predicates like Richman, Democratic Country and so on and so forth by combining predicates in this manner essentially. Again we will look at more when we look at description logic essentially. So this gives us a flavour of how do we express relations and define new relations in terms of old. Even when we are using adhoc

predicates but we will see as we go along that adhoc predicates is not a very good idea because the number of predicates which you may end up creating is far too many essentially. So you need some degree of economy there, we will come to that later. But meanwhile in the next class we will look at the notion of properties, we have not seen how properties will be represented. So for example I will leave you with this question as to how do I express the fact that the ball is red essentially. And we will take it up in the next class.