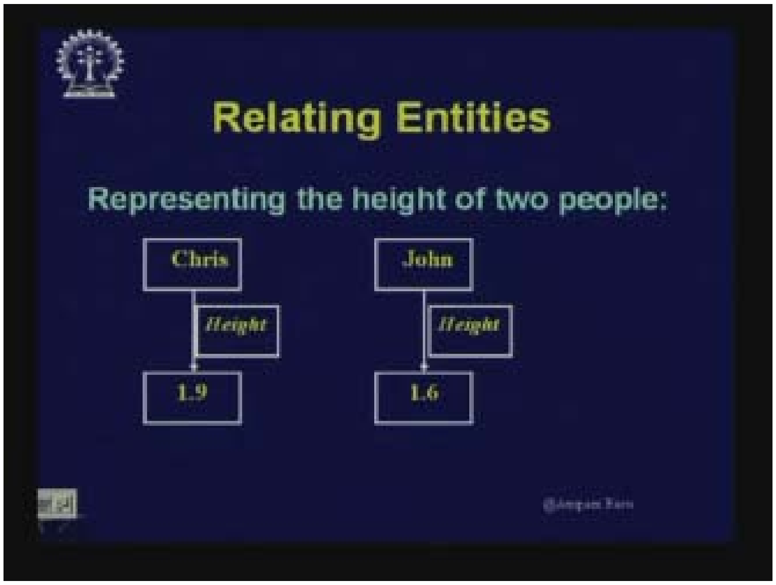
Artificial Intelligence
Prof. Anupam Basu
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture - 19
Reasoning in Semantic Net

In the last lecture we saw a new knowledge representation scheme called semantic net. We introduced the basic concept of semantic net in that lecture and prior to that we have seen that knowledge can also be represented using rules. There are some differences between the rule based representation and the semantic net representation. In semantic nets the word concepts are represented as nodes and the relationships between the different concepts are denoted as edges between the two nodes. Now there are some varieties of such representations.

Any entity or any concept we represent from the world can be represented as a node. The relationships can be of different types but most important relationship that is occurring in semantic nets is for example, cat is a mammal, tom is a boy. Now essentially it turns out to be very similar to a graph or a network of concepts and relationships. But as you will recollect that no knowledge representation scheme is worth its name unless we have a mechanism using which we can reason and infer new facts of answer questions and really do some useful work of inference. Only in that case if we can do those then we can certify that to be a valid knowledge representation scheme. In the last lecture what we discussed is just the representation the graphical structure and we also gave a couple of examples.

(Refer Slide Time 04:00)



Relating Entities

Representing the height of two people:

Chris

Height

1.9

John

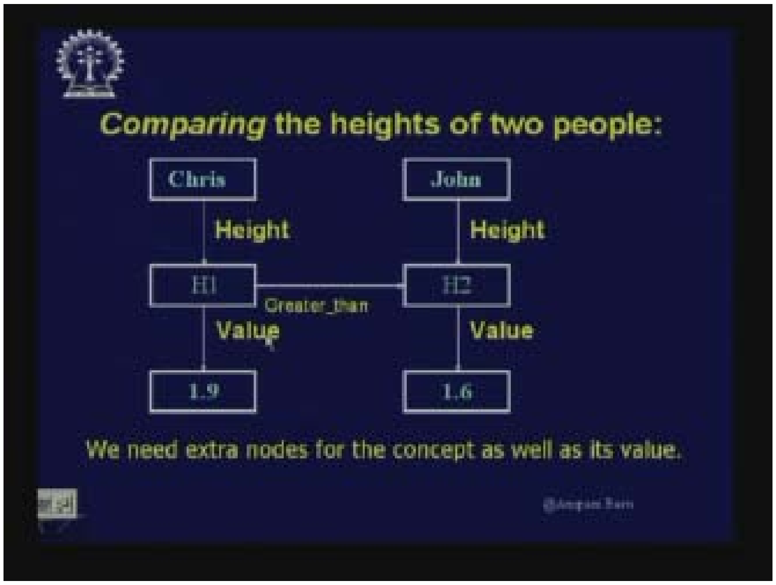
Height

1.6

©Anupam Basu

Today we will be extending that discussion and will include in our discussion how semantic nets can be used to reason so that we can infer new things or can answer questions. Today our objective is to look into reasoning in semantic net. Here is one more example, a new example which is little different from what we did in the last class. In this case we are trying to represent the heights of two people. Chris is one person and John is another person so the heights might be in meters or whatever but these are the values. but these are again in semantic nets we are not making any distinction between the types of the concepts and these two concepts are linked by the relationship height therefore it is standing to represent the sentence the height of Chris is 1.9 and the height of john is 1.6. These are two independent statements, height of Chris is 1.9 and height of John is 1.6 but if we want to state that the height of Chris is greater than height of John in that case we will have to add some extra nodes to designate that concept. So you can see how semantic nets can be extended. Here we can see that we are now bringing in two other concepts, these are constants 1.9 and 1.6, now the height of Chris is H1 and the height of John is H2 and the value of H1 is 1.9 and the value of H2 is 1.6 so we have already brought in one new relationship called value over here.

(Refer Slide Time 05:50)



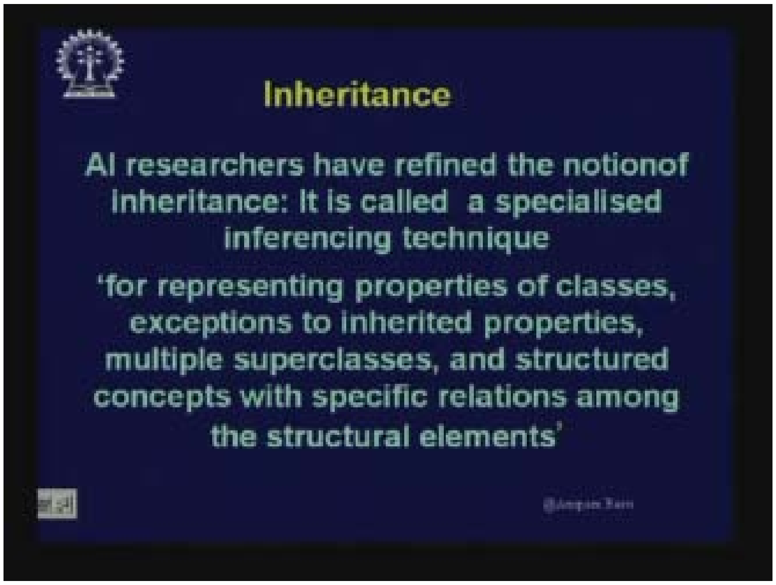
Comparing the heights of two people:

We need extra nodes for the concept as well as its value.

@Anupam Basu

For the sake of comparison what we can do is we can establish another new relationship between H1 and H2 and state that H1 is greater than H2. So, from this augmented semantic net now we can say that height of Chris is H1 and the height of John is H2 and H1 is greater than H2 therefore I can compare the height of Chris and John. But you can se that in order to achieve this we had to incorporate extra nodes for the concept as well as its value. We had to add two new nodes H1 and H2 and also we had to incorporate another relationship called value. At this point of time let us try to see how we can utilize semantic nets to reason the basic mechanism of reasoning using semantic nets is the concept of inheritance. Before going into the reasoning mechanism let us have a look at what we mean by inheritance.

(Refer Slide Time 07:38)



Inheritance

AI researchers have refined the notionof inheritance: It is called a specialised inferencing technique

'for representing properties of classes, exceptions to inherited properties, multiple superclasses, and structured concepts with specific relations among the structural elements'

©Anupam Basu

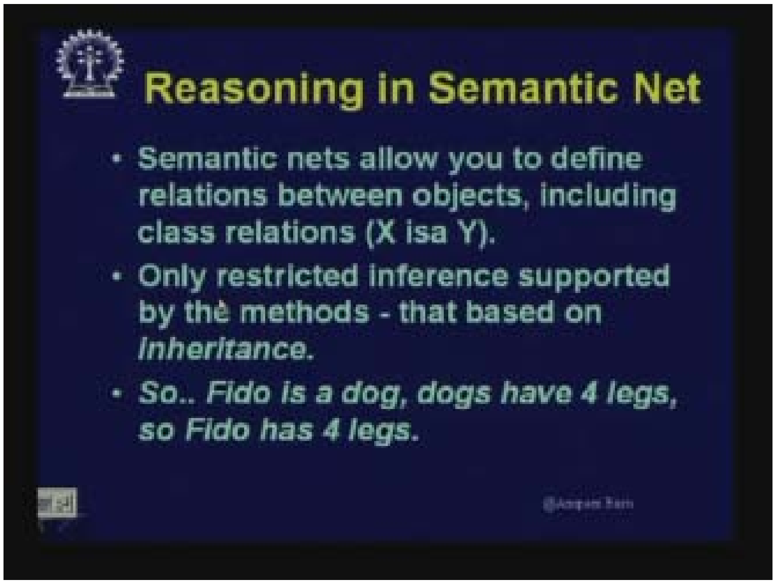
Inheritance is a word that is very common in our vocabulary. People inherit property from their parents, from their grandparents etc so it is getting some things from their predecessors parents etc. In other words we can extend it for example, if we say cat then cat is a concept and we say cat is a mammal then cat must share or inherit some properties that mammal has. Now the concept mammal will have some properties so cat being a mammal cat will also inherit some properties of mammal. So, using this idea, AI researchers have refined the basic notion of inheritance called a specialized inference technique for representing properties of classes.

We have got the notion of classes and we have got the notion of objects. You will find very close similarity and this is no magic that there is a similarity; they are actually the same concept being refined over years of research. So, if we think of some class and in the earlier example as I said mammal is a class, mammal designates a class of animal including human beings and cat is a subclass of mammal. Since it is a subclass of mammal it inherits some basic properties of mammal.

In addition, it can have some specialized features which cats have but human beings also being mammals do not have. For example, we have got two legs whereas cats have four legs but mammals have got some feature that mammals drink milk so that property is inherited both by cats and human beings. So there is a notion of classes and subclasses in the inheritance mechanism. The is-a relationship as we defined in semantic nets is nothing but a class subclass relationship.

Cat is a mammal so cat is a subclass of mammal. We want to represent properties of classes and exceptions to inherited properties. Some times the mechanism of inheritance will also allow us to make exceptions to the properties inherited because all the properties that are there in the class may not be there in subclass. It is also possible that one particular concept or one particular class will have multiple parents so there can be multiple super classes. Each concept can have a structure within itself and the elements of that structure can have specific relationships among them. That is what is known as inheritance.

(Refer Slide Time 11:52)



Reasoning in Semantic Net

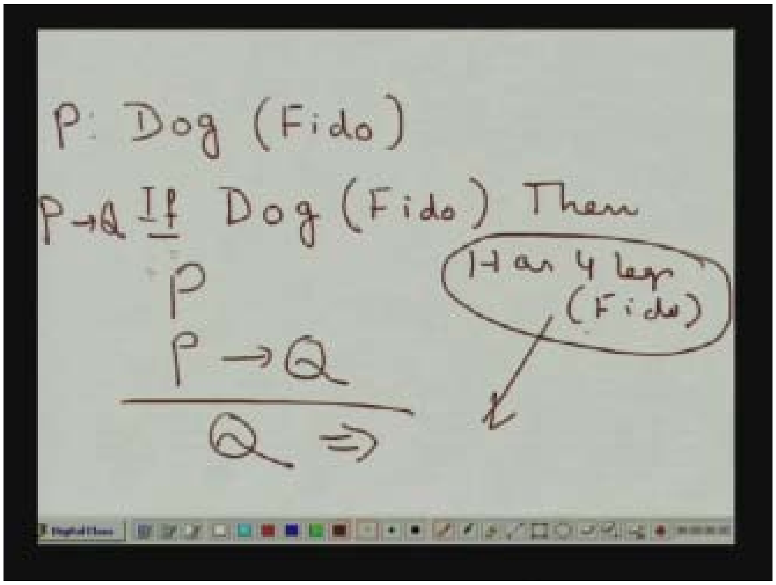
- Semantic nets allow you to define relations between objects, including class relations (X isa Y).
- Only restricted inference supported by thè methods - that based on *Inheritance.*
- *So.. Fido is a dog, dogs have 4 legs, so Fido has 4 legs.*

©Anupam Basu

Here are some examples: As we know in semantic nets the most powerful relationships or the most common relationship we use is the is-a relationship. In order to reason in semantic nets how this is-a relationship can be used? Semantic nets allow us to define relations between objects including class relations. For example, X is a Y. The first step is we can define this class subclass relationship X is a Y so X is a subclass of Y. The only restricted inference supported in semantic nets is the methods that are based on inheritance. So here is an example, Fido is a dog and dogs have four legs so we can infer Fido has four legs. Now, if you just think a while you could have also represented this using logic.

So how did it look?
Fido is a dog and dogs have four legs so Fido has four legs. That can very well be inferred using our well known Modus ponens method, Fido is a dog and dogs have four legs that means I am writing in propositional form. If dog Fido then has four legs Fido. Therefore if I say this is P and this is P implies Q then according to modus ponens if you recall you would have inferred Q using this and Q is nothing but has four legs Fido. So you could have inferred this using logic also or using rules.

(Refer Slide Time 15:05)



P: Dog (Fido)

P→Q If Dog (Fido) Then Has 4 legs (Fido)

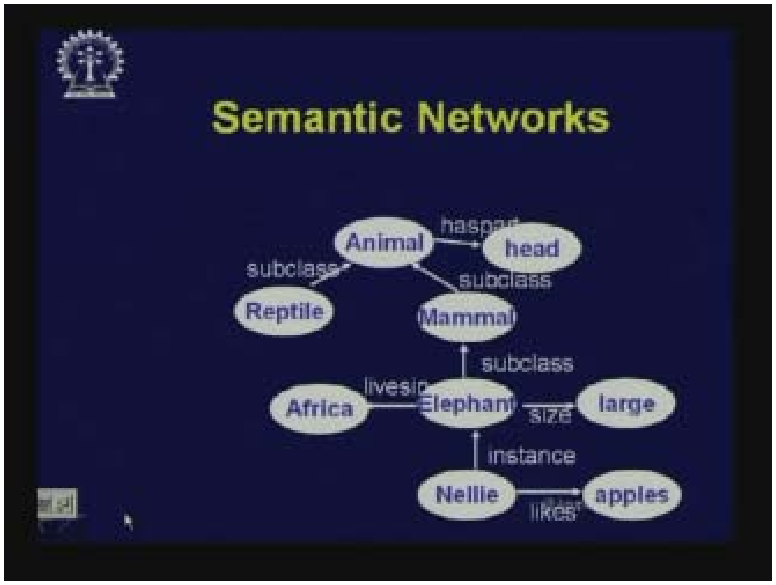
$$\frac{\begin{array}{c} P \\ P \rightarrow Q \end{array}}{Q \Rightarrow}$$

But these sorts of notations are not very common to our experience and it has been introduced by logicians as a very powerful tool to handle. But using semantic nets we can achieve the same thing. For example, if I just try to draw the semantic nets equivalent to this we could see that, here is a concept dog and dog number of legs is equal to four and Fido is a dog. Now this picture very clearly shows the relationship Fido is a dog and the number of legs of dog is four. So now from here we can inherit, you see how inheritance can be used. Since Fido is a dog and dog has number of legs to be four therefore Fido will also have number of legs to be four. Now this property of number of legs has not been stored with Fido because I could have had for example Jim as another dog. Now I need not store the fact that Fido has four legs Jim has four legs. If I had to store that all the time if I had to store this here again Jim has four legs then that would be redundant storage which is not require.

On the other hand, I can very well infer from this structure in a logical way. So where is the inheritance? The inheritance is here that Fido being a dog is inheriting the property of number of legs being four to it therefore we can infer it. But there are some exceptions to it. For example, if now somebody says that Fido is a lame dog then might be Fido has got three legs and not four legs. therefore whenever we infer something using inheritance that means from the parent we extract that and use that information that is fine unless anything is known to the contrary. So, as we go down the hierarchy as we go down to the specialized cases like Fido if there is an exception then I could have said Fido has three legs. If nothing is told I will infer Fido has four legs because Fido is a dog. That is why in the earlier case we said that there can be exceptions to such inheritances.

Here is another example semantic net.

(Refer Slide Time 18:51)



Semantic Networks

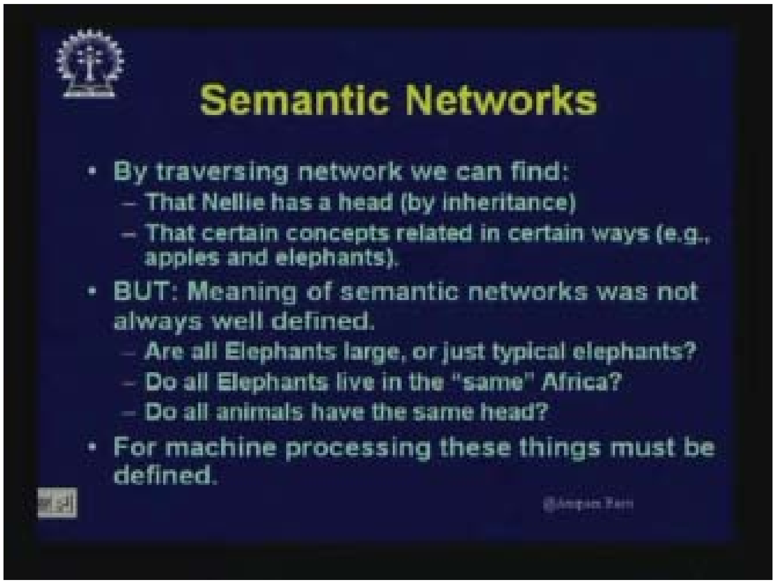
Here instead of is-a I have used specifically the relation subclass because by now you know that is-a essentially denoting a subclass. So we have got a root node animal and animal has a path head. Reptile is a subclass of animal, mammal is a subclass of animal, elephant is a subclass of mammal. Now elephant size is large and elephant lives in Africa so if I club together each of these relationships are equivalent to some sentence and the meaning of the sentence is very clear. That is why the name semantic which means meaning is used here, the word semantic net is present.

For example, we can say if I cluster these two nodes together what it means?
It means reptile is an animal, mammal is a type of animal, elephant is a mammal, elephant lives in Africa, size of elephant is large. And now Nellie is a instance of elephant so Nellie is a particular elephant just like Fido was a particular dog Nellie is a particular elephant and we could have done it with is-a and now if we say is-a then there is a problem. That is why we need to specify it a little bit. If I say all these are is-a then Nellie is an elephant or is an elephant and what would that mean? That would mean, is Nellie a class of white elephants which are rather small or which have got three tusks or something of that sort? Is it another variety of an elephant or is it a specific elephant?

Now merely if we use the relation is-a that is not very much clear. Therefore is-a is being substituted here by two distinct relations; one is instance and one is subclass. There is a class, there can be subclasses of classes and there can be objects belonging to these classes and these objects are nothing but instances like Tom is an instance of a boy; boy is a subclass of person or subclass of male etc. So here instead of is-a we have used another relationship called instance so Nellie is an instance of elephant, Nellie is a particular elephant and Nellie likes apples. Now over here if we try to ask some questions as where Nellie lives we write a procedure that will search through this graph and we first start with Nellie because the query itself asks where Nellie lives. But right now we will search here for an answer and find whether it is lying at this level or whether there is any relation with Nellie saying Nellie lives in somewhere, but it is not available. Then we think, is Nellie a instance of some other class or is it a subclass of something?

We find that Nellie is an instance of elephant so although I did not find the answer here I go one level up and see whether the answer is available then I can say Nellie is an instance of elephant and elephant lives in Africa so my answer will be Nellie lives in Africa. Now, if the question was asked does Nellie have a head then let us see how we can do it? We would have to start all over again from Nellie. Now, Nellie is an instance of an elephant and here we do not find any answer to this because here we do not find any relationship called as has part so I do not find the answer here whether it has got a head or not. I go up one level high because elephant is a subclass of mammal and I search for this and I do not have an answer here so I again go up to animal and there I find that animal has a part head therefore Nellie has a part head.

(Refer Slide Time 24:45)



Semantic Networks

- By traversing network we can find:
  - That Nellie has a head (by inheritance)
  - That certain concepts related in certain ways (e.g., apples and elephants).
- BUT: Meaning of semantic networks was not always well defined.
  - Are all Elephants large, or just typical elephants?
  - Do all Elephants live in the "same" Africa?
  - Do all animals have the same head?
- For machine processing these things must be defined.

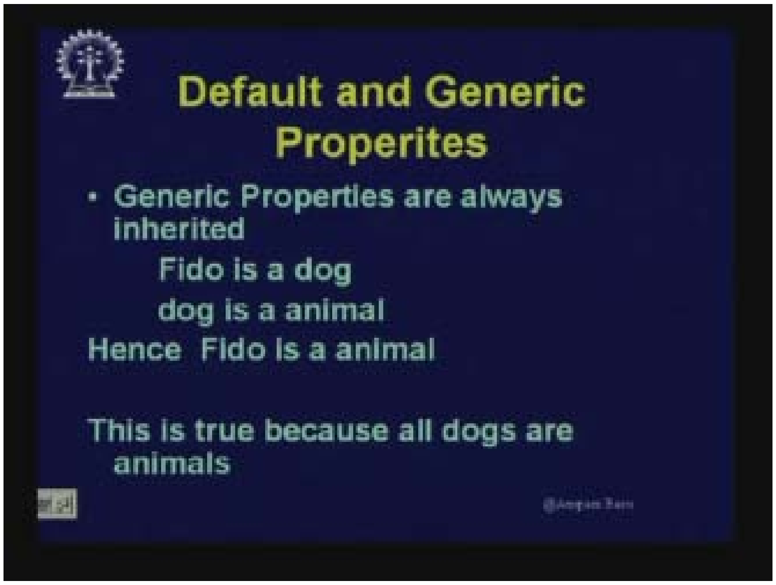
Therefore how did we infer? We inferred by traversing the semantic network and with that we found that Nellie has a head and that was purely by inheritance. We inherited the property of Nellie having a head. The property was much higher and that has been inherited along this arc to Nellie. And also we can find that certain concepts are related in certain ways like apples and elephants.

How are apples and elephants related?
Nellie likes apples but it really does not say whether all elephants like apples. So, as we go up it is a generalization and as we go down the hierarchy it is a specialization. Although Nellie is an instance of elephant and is a particular elephant Nellie likes apple but that does not mean all other elephants will like apple. But meaning of the semantic network was not always well defined. Even in this case all the meanings are not very much clear from this description we have given. For example, we cannot really say are all elephants large or just typical elephants are large.

Probably in the real world may be, all elephants may not be very large but as we have done over here it is apprehend that we will infer all elephants to be large but we have got no other option here. Do elephants live in the same Africa are some complicated meanings which you may like to ponder on. When we really write a program that will perform such inferences we cannot allow such ambiguities and we will have to do away with such ambiguities. So, in general I hope it is clear how the mechanism of inheritance works.

To summarize; a subclass can inherit the properties of the parent class. Now there can be instances or subclasses now instances or subclasses may have some additional properties which may be changed. For example, somebody inherits small house from his parent and then he can change it to a larger partner. Similarly the specializations or the subclasses can override or change the basic properties from those that were there with the ancestors.

(Refer Slide Time 28:06)



Default and Generic Properites

- Generic Properties are always inherited

    Fido is a dog

    dog is a animal

Hence Fido is a animal

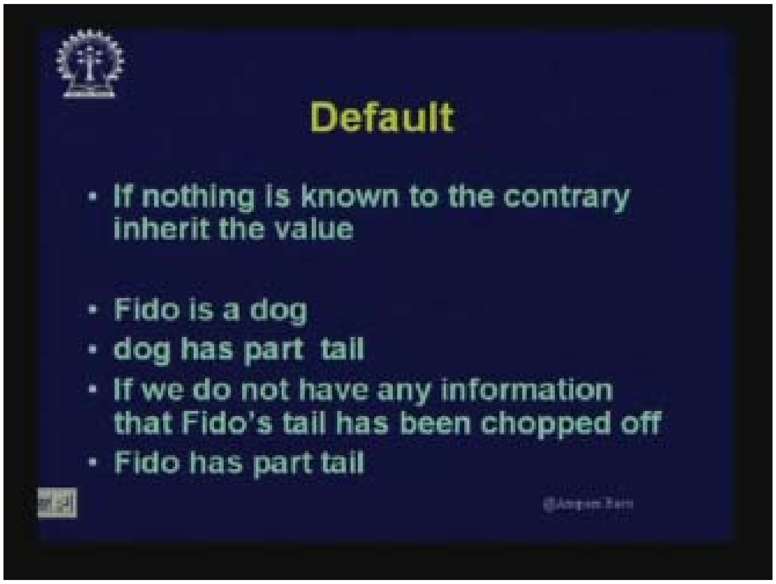
This is true because all dogs are animals

@Anupam Basu

Now it is necessary to distinguish between two very important categories of properties. One is the default property and another is the generic property. Properties are always inherited for example Fido is a dog, dog is a animal but this is not a grammatical error because we are talking of is-a relationship, in English it should be dog is a animal. So hence Fido is a animal this is inherited. These properties are generic to the super class the original class and whatever is there in the original class if that property is generic then that value will be inherited and we do not change that. There can be three types of properties you can think of namely generic, default and other. Now generic properties are not usually changed unless there are some special circumstances.

For example, if we have mammals then we know mammals are warm blooded species. Therefore for anything namely cat, dog, may be even human beings for that matter the generic property will be inherited. Besides that there can be default properties. What is the meaning of default property? Default means if nothing is known to the contrary inherit the value. Here is an example, if nothing is known to the contrary, I know Fido is a dog and suppose dog has a part tail.

(Refer Slide Time 30:03)



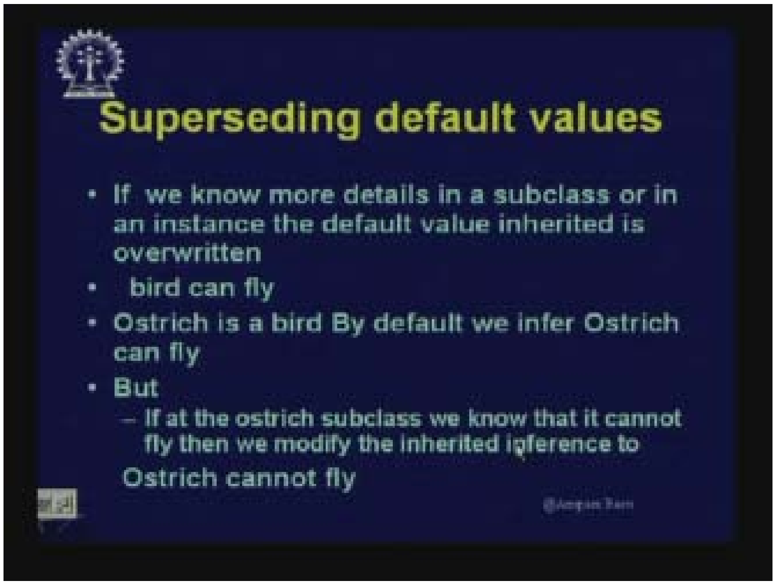
Default

- If nothing is known to the contrary inherit the value
- Fido is a dog
- dog has part tail
- If we do not have any information that Fido's tail has been chopped off
- Fido has part tail

Therefore what should we infer? We will have to infer that Fido has tail so there is no exception to this. If we have this that dog has tail and I have got Fido is a dog then I can say that Fido has tail. Now this is the property link, has part is the property link and tail is the value of that property. Now Fido is a dog and has part tail is fine. Therefore if I look at Fido then I will inherit this and I will say it also has part tail. But if it is said Fido is a specific dog, now this is is-a but this is-a is an instance. Now this Fido being an instance of dog if we know that there is a special case that Fido’s tail has been chopped off then this does not hold.

Therefore, I inherited this part by default because nothing was known to the contrary. If anything is known to the contrary that will override whereas in the case of generic systems or generic properties we do not have this scope of overriding because that is generic to that class and any subclass will have to inherit that. So, under normal circumstance you would have inferred that Fido has part tail. Now if we know more details of this that Fido’s tail has been chopped off then we would have inferred Fido does not have tail.

(Refer Slide Time 32:33)



Superseding default values

- If we know more details in a subclass or in an instance the default value inherited is overwritten
- bird can fly
- Ostrich is a bird By default we infer Ostrich can fly
- But
  - If at the ostrich subclass we know that it cannot fly then we modify the inherited inference to

Ostrich cannot fly

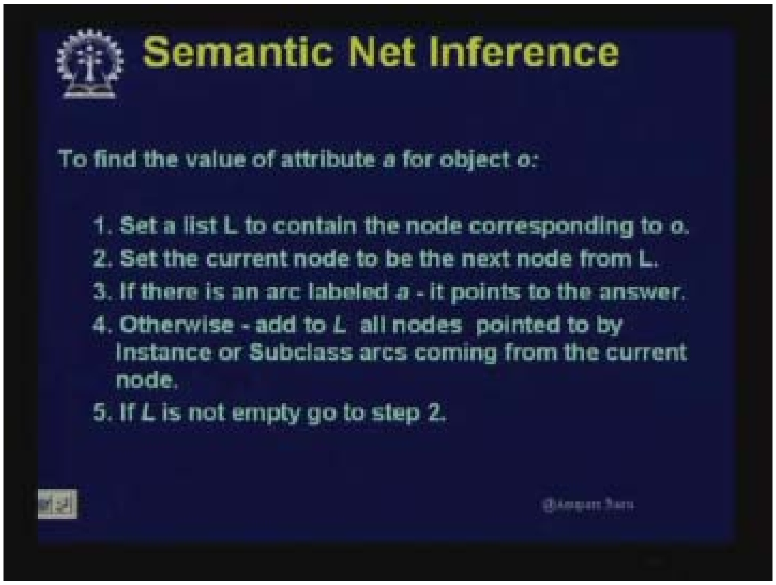
Now it is necessary that we can supersede default values if we know more details of a subclass or in an instance the default value inherited can be overwritten. Now here is a classical example; birds, we know birds can fly, we know birds have feather birds are covered with feather and we also know that penguin is a bird, we know ostrich is a bird, we know eagle is a bird, sparrow is a bird. Now if we think of sparrow, sparrow is a subclass.

Here is a question, is sparrow an instance or a subclass?
Obviously sparrow is a subclass of birds, it is not an instance because I can have ten sparrows in my cage and each of them is distinct. So sparrow is a subclass of birds will inherit the property has feather because all birds have feather. Now we also know birds can fly therefore we can also infer sparrow can fly because sparrow is a bird. Now, has feather is something common but can fly is not that generic because if we take the example of ostrich then we will see that ostrich cannot fly but ostrich has feathers. So ostrich being another subclass of birds will inherit the property of has feathers and will also automatically by mechanism it will inherit the property can fly but because we know something to the contrary specific to the subclass ostrich that will be overwritten and we will infer that ostrich cannot fly, so here is that example.

Ostrich is a bird and by default we infer that ostrich can fly. But if at that ostrich subclass we know that it cannot fly then we modify the inherited inference to ostrich cannot fly. Now this is a very important and powerful means of reasoning and this one can be discussed in a little more detail. Two types of reasoning we often talk about; one is monotonic reasoning. Monotonic reasoning means whatever we infer at a particular point of time given some facts will hold. So anything that is inferred at some point of time T will not be invalidated by the arrival of some more information at a later point of time. So, if we go along this timeline and at this point we are inferring some x is equal to 5 for example and now if I add this point of time which is coming then later I cannot invalidate that reasoning, it is not like a program. Instead of this we say we infer at this point that Tom is a man. The arrival of other facts is being inferred at this point of time during the reasoning process but never will revise this. We will add on new facts here but this will not be revised.

On the other hand, the other type of reasoning called non monotonic reasoning which means whatever I have inferred at this point of time with the arrival of new facts or information that I am generating this can be changed, it is possible to invalidate that and you will see that this default reasoning is nothing but an example of the same. Here we have done exactly the same thing that we had made an inference here but after that as we came to know more facts about ostrich we could override that. Now these default values can be overwritten but generic values cannot be overwritten. If we just think of a procedural form, suppose you want to write an inference machine if we want to use semantic nets as a valid knowledge representation scheme then we must have an inference mechanism for that. Therefore we will have to write a procedural code which will work on the semantic net and we will be able to infer new facts.

(Refer Slide Time 39:00)



Semantic Net Inference

To find the value of attribute *a* for object *o*:

1. Set a list L to contain the node corresponding to o.
2. Set the current node to be the next node from L.
3. If there is an arc labeled *a* - it points to the answer.
4. Otherwise - add to *L* all nodes pointed to by Instance or Subclass arcs coming from the current node.
5. If *L* is not empty go to step 2.

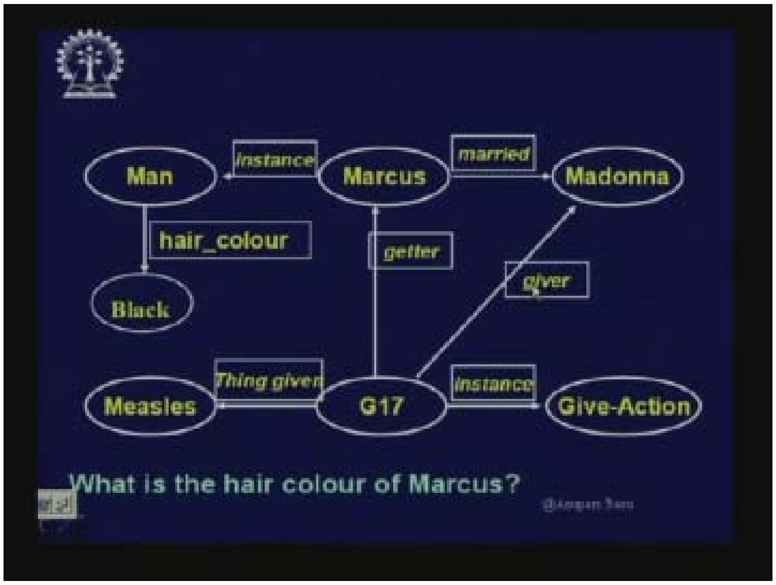
Now what we want is we want to find the value of an attribute for an object. What is that object? For example, Nellie is an object or Fido is an object or may be mammal is an object. Now you want to know whether mammal has part head or Nellie lives in Africa. That means if I have a query like does Nellie live in Africa then in that case what I am trying to do is I am going to that particular object Nellie and I am trying to see whether it has got any relationship lives and if such a relationship exists there then is the value of that link is leading to another value which is Africa, that is how we go about. So essentially in order to answer the queries whatever queries you form we can form that in the form of an attributes value that we are seeking.

For example, if we take an earlier example like has part, does elephant has head, so basically what we are trying to do is we are looking at the object elephant and we are trying to see whether it has got an attribute has part which is leading to head. Therefore in this case we are setting a list L to contain the node corresponding to O. We are setting a list linked list sort of which is containing the node corresponding to the object O and we are setting the current node to be the next node from L because we have to traverse that over a link. If there is an arc labeled a…. the attribute we are seeking then it points to the answer. We get the answer if we get it over there. Otherwise what I have to do is I will have to traverse that network. So in order to do that we will add all the nodes to L pointed to by the instance or subclass arcs coming from the current node.

For example, here I want to have some I have got a query where does Nellie live, now this question where does Nellie live can be translated in the form of a semantic net like I have got Nellie lives in some value which is not known, I could have put a query or could have put x so what is the value? I want to instantiate this particular value; I want to find the value of this value from the net. So this is my query semantic net you can think of as if it is a template with which I am going around in the semantic net and I am trying to find the best match.

Now what would have happened?
With this query I start my journey along this semantic net. Here unfortunately I do not find the match because here I am not finding any arc coming out of Nellie so I am making a list with Nellie and it is pointing to all the arcs which are connected by is-a or instance, subclass or instance type of relationships. So right now in that node we have got only elephant so now we come to this elephant and as we come to this elephant what are the next possible places to go? In that list I add mammal and if I do not find the answer here I have to go up and look into mammal whether that lives in link can be answered.

(Refer Slide Time 45:11)



What is the hair colour of Marcus?

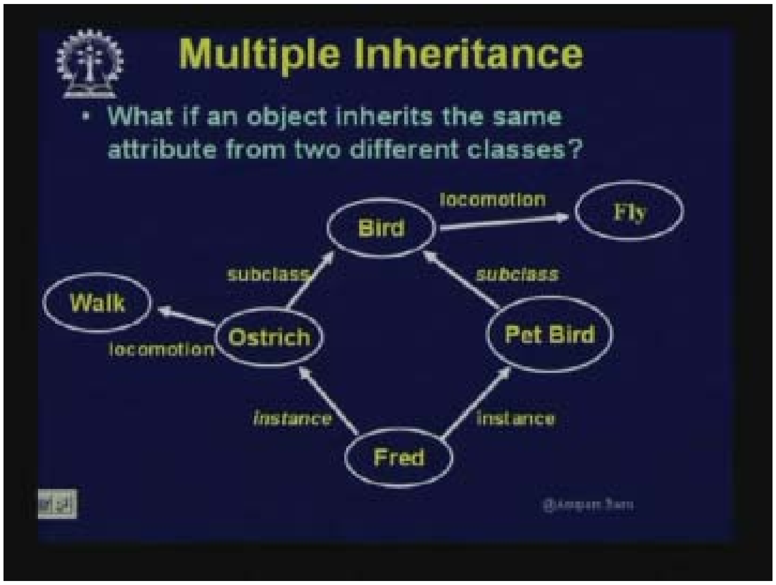
Here as I go in I find that the lives in link are available here and so my answer is Africa. if my query was does Nellie have head for example a funny query but then I would have started from here I would not have found the answer here but I would have gone up and again i would not have found the answer here so the link to the list would have included this node and would have gone up in this way till I find this answer.

Let us again go back to that algorithm. If we do not find an answer we add to L all nodes pointed to by instance or subclass arcs coming from the current node. and if L is not empty that means there is a possibility of exploration we will set the current node and we will look into that current node to br the next node and will see if there is an arc labeled a at that point. This is the search mechanism we use in semantic net.

Now here is another example; what is the hair color of Marcus? Now let us look at this semantic net what does it say?

We have seen a similar example in the last lecture, man is a class; hair color of man is black is it generic or is it default? It is default because we are assuming that there are people with hair color brown and other types. Marcus is an instance of man and Marcus is married to Madonna. This is a semantic net, Marcus gave Madonna measles was an example we already saw. Now if in this case we want to answer the question what is the hair color of Marcus? So obviously the net that will be is Marcus with an edge hair color x which we do not know. And I make Marcus my current node, now no arc is coming out of Marcus with the relation hair color. But there is instance or subclass, I am not traversing this at all the pointers I am putting into the next possible nodes are either the instance or the subclass type of relationship. So here we take man and we find that man has got a hair color therefore Marcus is inheriting the answer and the hair color of Marcus is black.

(Refer Slide Time 48:30)



Multiple Inheritance

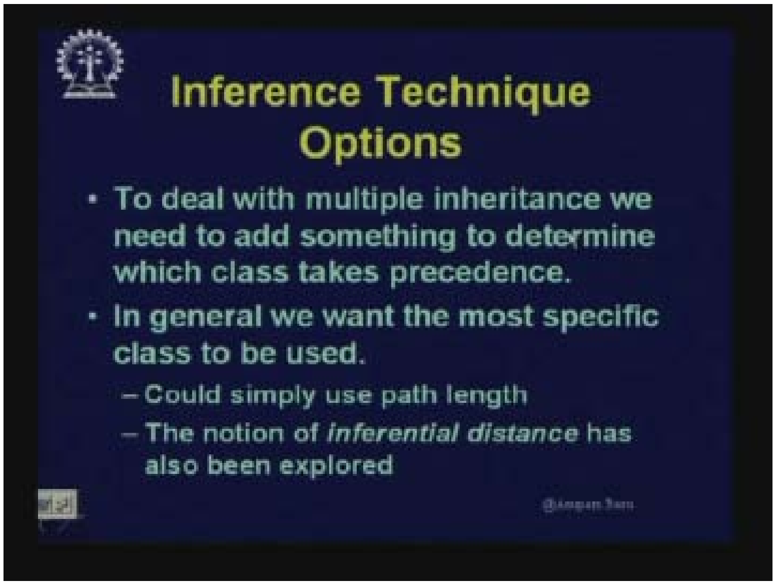
- What if an object inherits the same attribute from two different classes?

That is how we carry out inferencing in semantic nets. But there is a major problem over here and that is known as the case of multiple inheritance what is this multiple inheritance? The problem will occur if an object inherits the same attribute from two different classes. For example, here you see Fred is an instance of pet bird and is an instance of ostrich. So Fred is an ostrich that has been kept as a pet. Now ostrich has got the property locomotion is walking so ostrich walks. Pet bird is a subclass of bird and its locomotion is flying. Now Fred is inheriting from two subclasses ostrich and pet bird.

Now how do we infer if we say does Fred fly how do you answer? Or rather in terms of this net if my query is formed as what is the locomotion of Fred?
Now I can say I start with this I do not find any locomotion edge from here so I can go up this stream and come to ostrich and find that ostrich has got a locomotion edge so I infer walk. But what if I had chosen Fred to be an instance of pet bird and what would my algorithm lead to? I would have come here and here I do not find an edge therefore I have got no way of going down this level so I go up and there I find locomotion fly so I could have inferred that Fred flies so there can be different answers as to how to handle with this.

One thing is; suppose you know about different search mechanisms and if I had done depth first search then I would have either gone to this answer or to this answer. But if I had done breadth first search then I would have got both the answers such as fly through this path and walk through this path. Now my common sense would have told me that whatever is more specific is more correct. So what is more specific here is walk so that is one way of handling this situation. So, an inference technique option in such a scenario is something like this. To deal with multiple inheritance we need to add something to determine which class will take precedence.

(Refer Slide Time 50:06)



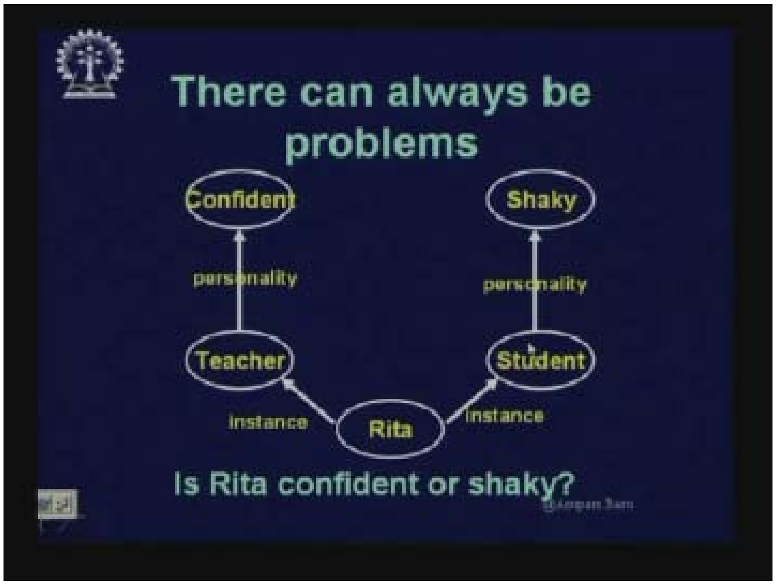
Inference Technique Options

- To deal with multiple inheritance we need to add something to determine which class takes precedence.
- In general we want the most specific class to be used.
  - Could simply use path length
  - The notion of *inferential distance* has also been explored

@Anupam Basu

For example, in this case we can say that well pet bird will have precedence over ostrich or ostrich will have precedence over pet bird depending on the particular application you can think. You can decide on some policy to do that. in general we want the most specific class to be used so we could simply use the path length because the notion of inferential distance can also be explored so it could be something like this that the path length of fly is 1 2 3 or 2 at least 1 2 and the path length of walk is 1 so which ever path length is less is more specific and that is more valid we could have taken that. Now here there is no hard and fast answer and depending on the application often we have to select a particular policy. But still there can be always problems like here is another example; Rita a girl is an instance of teacher, Rita is a teacher and Rita is also a student. Rita teaches in a school and also goes to a B. Ed class or some postgraduate class where she is a student. Now the personality of Rita is confident when she is as a teacher in the class but the personality of Rita is rather shaky when she is a student.

(Refer Slide Time 52:00)



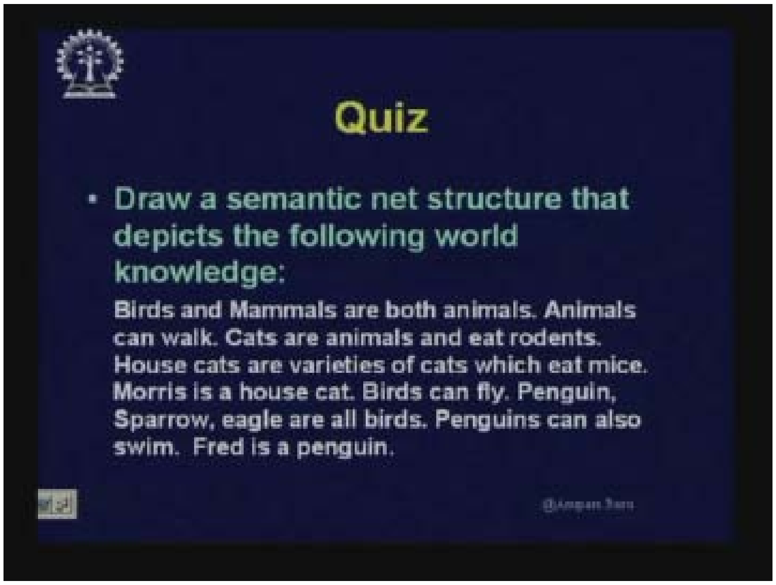
There can always be problems

Is Rita confident or shaky?

Now, if we have to answer the question is Rita confident or shaky it is really difficult because the path length will not lead us anywhere. And what is the personality of Rita? For this we are getting two different answers at the same distance so that problem will remain. So this is a problem of semantic nets and this is one of the problems, multiple inheritance has always been a problem and it is a very practical problem and there has been practical solutions to this. Now semantic nets as we have discussed is a very important basic tool.

From semantic nets we will go to frames which were proposed by Minsky and frames can be considered as the predecessor of the object oriented systems and we will see how the notion of semantic nets has been applied in frames. Here is a quiz, draw a semantic net structure that depicts the following world knowledge. Birds and mammals are both animals, animals can walk, cats are animals and eat rodents. House cats are varieties of cats which eat mice. Morris is a house cat. Birds can fly. Penguin, sparrow, eagle are all birds. Penguins can also swim. Fred is a penguin. In that way I have just put in a number of facts. Depict it in the form of a semantic net and you will also find that it is possible to add on more sentences to this and enrich the semantic net.

(Refer Slide Time 54:15)



Quiz

- Draw a semantic net structure that depicts the following world knowledge:

Birds and Mammals are both animals. Animals can walk. Cats are animals and eat rodents. House cats are varieties of cats which eat mice. Morris is a house cat. Birds can fly. Penguin, Sparrow, eagle are all birds. Penguins can also swim. Fred is a penguin.

©Anupam Basu

Before concluding this lecture let us quickly summarize what additional things we have seen in this lecture. We have first of all made a distinction; till now we were just talking of instance relationship but there can be subclass and instance relationship. And the major inferencing mechanism of semantic net is through inheritance mechanism. When we inherit the properties from the super class, inheritance mechanism means the properties of the super class is inherited by the subclass and the instances.

But there are two different types of properties; generic properties and default properties. In the case of default properties if nothing is known to the contrary then the default values will hold but if at a lower level a specialized level the subclass level there is some overriding fact that will dominate and the default value will not hold. And the algorithm of inferencing is to make a template of query semantic net and try to find the best match by searching through the links in the semantic net and while we do this search what is natural is either the instance or the subclass links are traversed. And we can answer so many questions, if we represent the world facts in the form of semantic nets it is still being used in different forms and the frames and object oriented systems are two different products of the semantic net links.