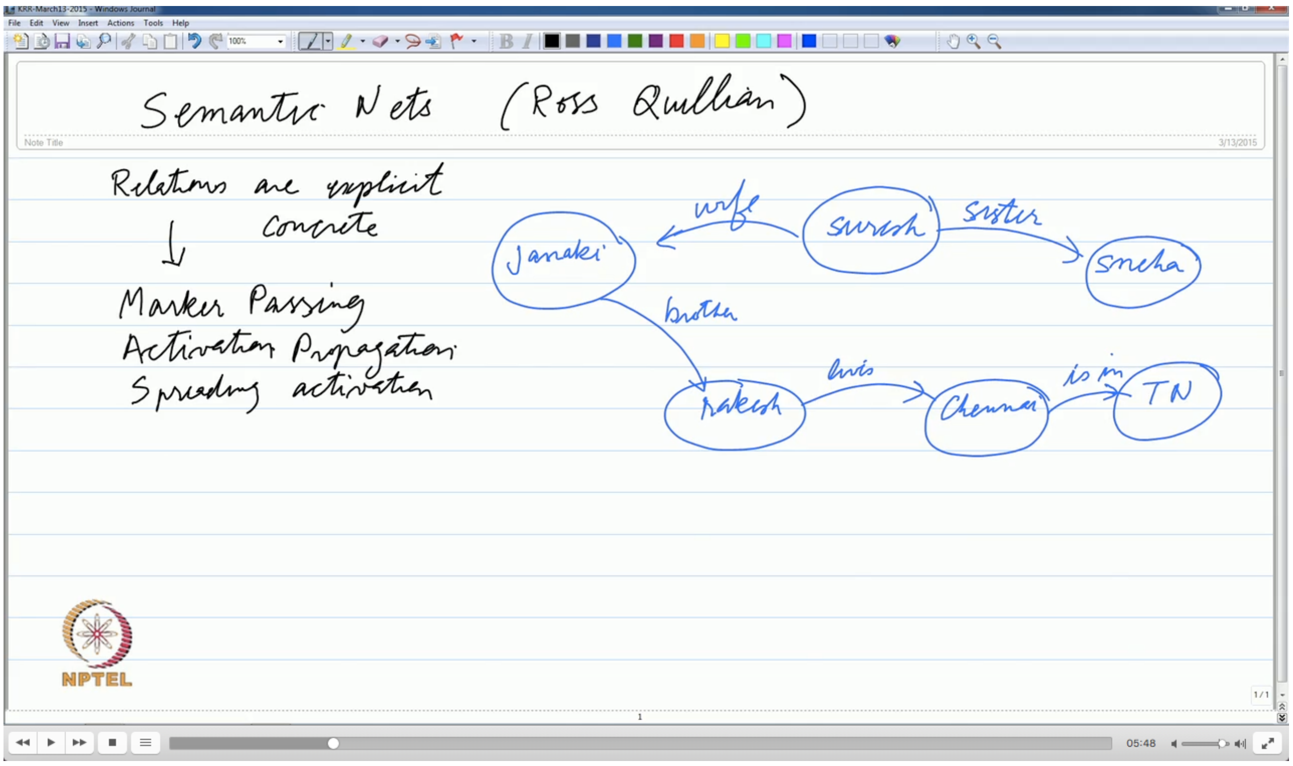
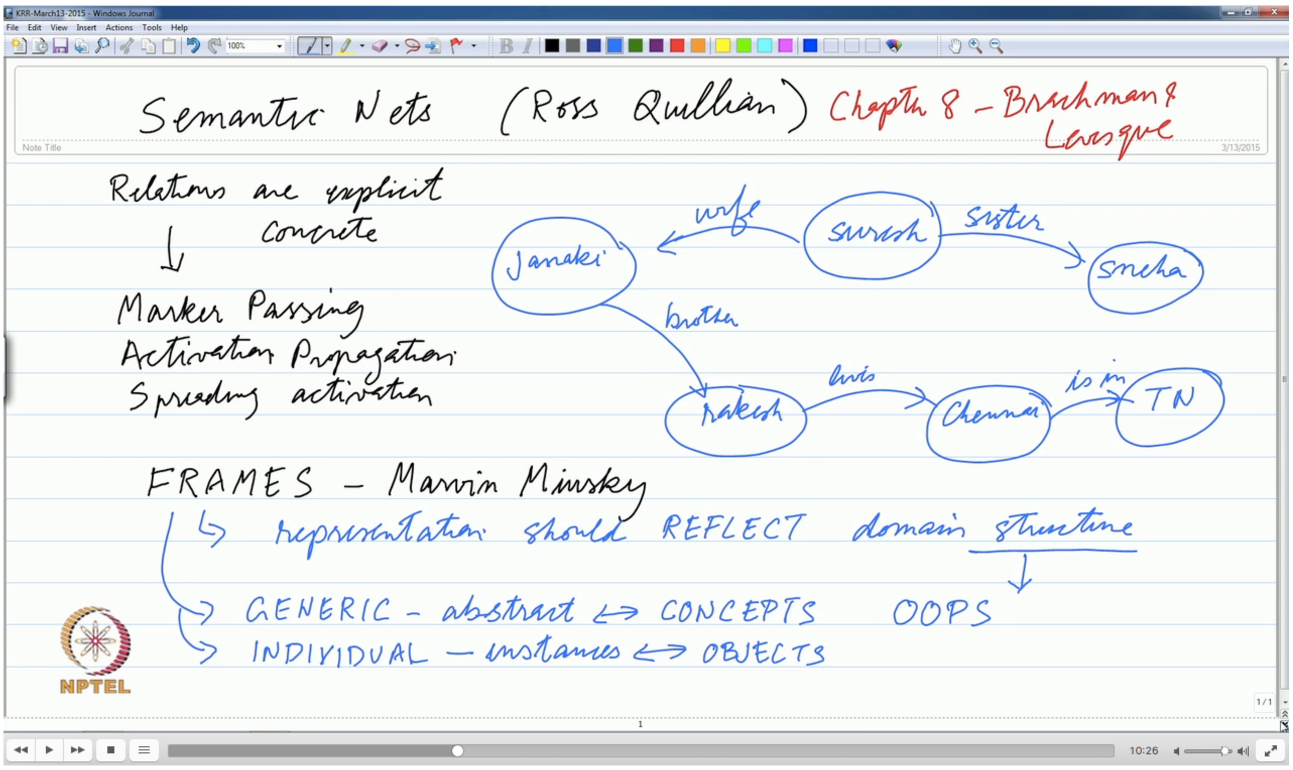
Semantic Nets and frames

We have spent considerable amount of time looking at logic, reasoning and theorem proving. One thing common in all of that is you have to write programs which will search for matching formulas. So for example if you are doing backward chaining or forward chaining where you have to search for matching formulas which will allow you to do the chaining process. Now obviously search require computational efforts. There has been fair amount of work focussed towards avoiding this search essentially. We are moving towards representation now and trying to see ways of representation where this kind of reasoning will be much much simpler essentially. One of the first ideal came forward towards this is notion of semantic nets and this idea is due to Ross Quillion. So essentially a Semantic Net is a graphical form of representing logical statement. So you might have something like this [Time 01:39]. You may talk of Suresh and you might say that Suresh has a sister who is called Sneha. And Suresh has a wife. Let us say she is called Janaki. May ke Janaki has a brother who is called, lets say, Rakesh and so on and so forth. I have shown relations so far, human relationships. Any kind of relations you could add. For example, you can say Rakesh lives in Chennai. Chennai is in Tamilnadu and so on so forth. You can edges where every edge is a relation and every node is a term. You could construct this way. Now what is the objective of this The objective of this is Relations are explicit. They are explicit even when you talk about logic. Charniak and McDermott use a term called concrete. In the sense that they are implemented. You can use a



term physical. For example, if you want to ask a question who is Suresh's sister? Then all you have to do is to follow sister arc from Suresh and you get the answer. You do not need to search through file looking for sister relationship and then seeing one of the argument matches Suresh and returning the other argument in that kind of a thing. Quillion also gave some algorithms to work on these. We will not go into those but I will just mention them here. They are Marker Parsing or also known as Activation Propagation. The idea here is that if you want to find some relation between any two entities. So for example, if i want to ask is Suresh anywhere related to Chennai, then by spreading activation, that is also a term we use 'Spreading activation'. We activate the node Suresh and Chennai and then we spread the activation. We activate their connected nodes. This is also called as marker passing sometimes as we mark those nodes and pass the marking token to connected nodes till the point that markers collide or when they intersect. Then you know that you have found a path from Suresh to Chennai. In this case you might give the answer that Suresh wife Janaki has a brother Rakesh who lives in Chennai. So this was one idea, that was, motivated by trying to avoid searching through unordered collection of formulaes. Of course, as we see that Prolog is not so unordered as Prolog have fixed search strategy, but basic idea is that somehow you find the matching formula and then you do matching and unification.

A related idea which we will discuss in next couple of classes is the idea of Frames. This is due to Marvin Minsky, who you must have heard is one of the founding people in AI. He founded AI lab in MIT for example. You can read up what you are going to discuss in this chapter eight of Brachman and Levisque. I am basically going to teach from there for this class . So idea of frame was that the representation should reflect domain structure or the structure in domain. Just as all the domain are structured, knowledge representation should also reflect this structure. Again goal is to avoid search for related elements. What is structure in a domain. This somehow corresponds to the structure that you use in the programming language. The whole idea of object oriented program-



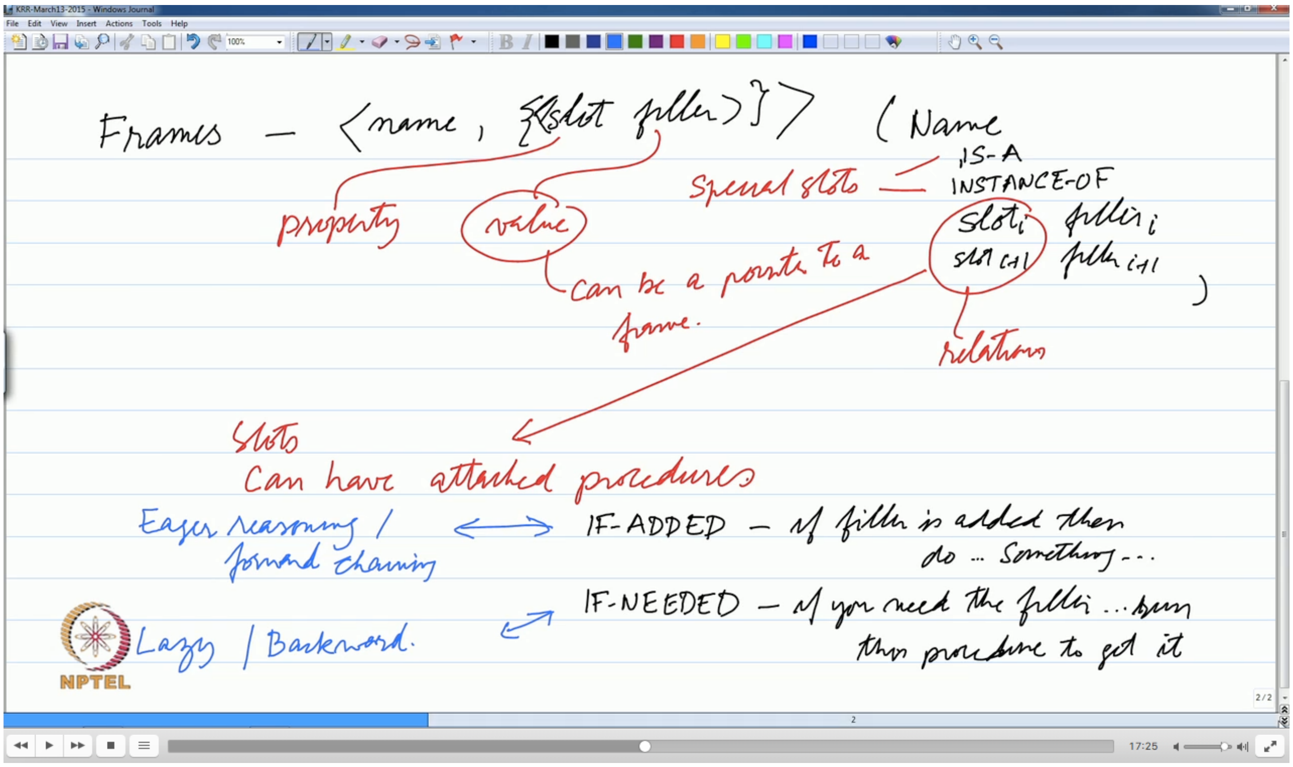
Semantic Nets (Ross Quillian) Chapter 8 – Brachman & Levesque

Relations are explicit
concrete
↓
Marker Passing
Activation Propagation
Spreading activation

FRAMES – Marvin Minsky
↳ representation should REFLECT domain structure
↓
OOPS
↳ GENERIC – abstract ⟷ CONCEPTS
↳ INDIVIDUAL – instances ⟷ OBJECTS

ming actually came from this idea of frame. What you are representing must be a reflection of domain you are talking about. So when you are talking about institute, representation must say that institute has department and department has people. People could be staff members and students and all this kind of things. This whole thing has been put in the structure. Brachman and Levisque talk about two kind of frames.
One they call Generic. We can say Abstract. And in some sense they corresponds to what we call concepts. They are abstract. They do not represent any concrete thing. It is a concept *e.g.* what is a chair or a house. These concept are represented in a generic frame. And then we have instances of generic frame which are called individual frames which talk about specific entities or ou can say they corresponds to objects. One of the ideas you will see that frames allows you to do, apart from the fact that you put everything together in one place other thing is that it allow economy of representation. You represent things at suitable abstract level and then you inherit it into something which is more specific. Lets look at the frames in little bit more details.
So Frames are made up of a name and a collection of slot and filler, structure essentially. So idea is so common now a days, one need not elaborate further too much. We can represent it as follows. For example, Let me just write it [Time 11:41]. We have a name, we have a slot and a filler.

Lets say, slot i and filler i, and slot i plus one and filler i plus one. And there are two special slots. This is in the terminology of Brachmann and Levesque. One is called is-a-slot and other is called



Frames — < name , {<slot filler>} > (Name

property

value

Can be a pointer to a frame.

Special slots — IS-A / INSTANCE-OF

slot$_i$ filler$_i$

slot$_{i+1}$ filler$_{i+1}$)

relations

Slots

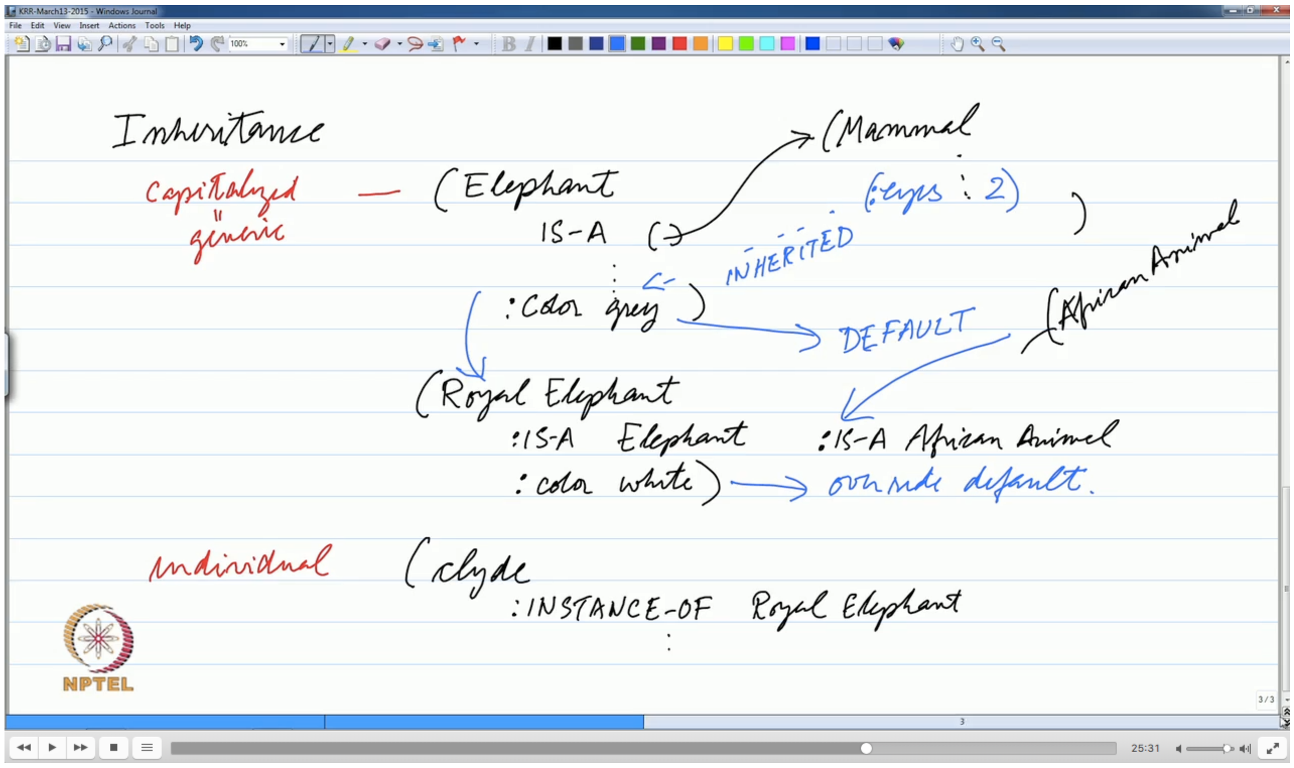
Can have attached procedures

Eager reasoning / forward chaining ⟷ IF-ADDED — if filler is added then do ... Something...

Lazy / Backward. ⟷ IF-NEEDED — if you need the filler ... run this procedure to get it

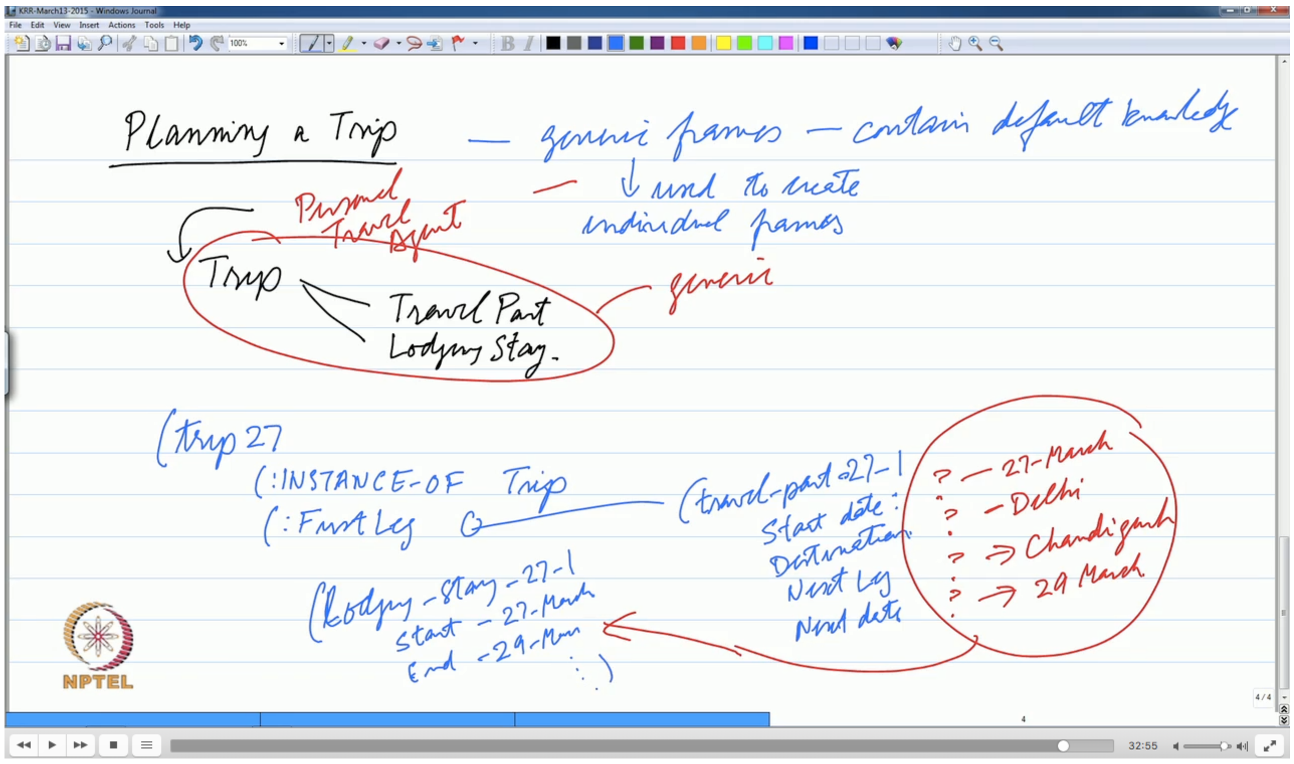
instance-of. So these are two special slots. [Time 12:00]. You can think of slot as a property and filler as a value. So remember the triples notation when we talked about RDF. So subject predicate object, subject-property value. Subject here is a name, whatever concept you are talking about. And there is set of property value which you put in a physical location in some sense and keep it together. This is the whole idea of avoiding search essentially. This value can be another frame. So you can see, this whole idea which we talked about in Semantic network is also being incorporated here. You have nodes which are frames, which are linked to other frames essentially. These slots capture the relations, edges in the semantic network, in case you put a pointer there [Time 14:18]. Slots can have attached procedures. and typically these procedures are as follows. For example, a procedure may be of kind IF-ADDED and it specifies that if filler is added, then do something. Some code has to be run. As soon as you fill in some value, a procedure is to run essentially. For example, you might say create another frame with this data. Other thing is IF-NEEDED. This says that if you need the filler, then run this procedure [Time 16:00]. Again we will see an example. May be you have constructed some frames. At the end of it, you want to do some computations. You can associate IF_ADDED procedure with Eager reasoning. And we have discussed this concept earlier. We also called this forward chaining. IF-NEEDED can be like wise be lazy or backward [Time 17:06].

By and large, most important idea from frames we get is that of inheritance. Let us illustrate this with an example. Let us say, we define a frame called Elephant. Bachman and Levesque say that capitalised names stands for generic frames. This is a convention we follow. Just as Prolog say that capital is used for variable and lower case is used for constants. So elephant is a generic frame. So it is a concept we are talking about . So it might have special slot IS-A for example. This could be a pointer to another frame whose name is mammal which we will not fill. We can say an elephant is a mammal. We can specify some fields for elephant. Whenever we say that an elephant is a mammal, we essentially are saying that a type, or sub-class or a sub-concept of or a sub-category [Time 19:05] essentially. In inheritance, you can say that how many some-property, lets say, eyes and lets say two. Essentially trying to say that mammals have two eyes. So whole idea of this is that once you created a frame elephant which is type of mammal, so this is inherited here [Time 20:00]. You do not have to represent that elephant has two eyes or leopard has two eyes or cat has two eyes and so on as they are mammals. Whatever is true of mammal will apply



to them essentially. Then you could have another frames which say royal elephant. Here, i could say IS-A [Time 20:47] . I am writing it here, you can think of this as kind of pointer. It is a pointer to elephant frame essentially. You must be able to distinguish between literals and frames and there should be a mechanism essentially. So I might have said in elephant. One of the property is color is grey. One would expect that this property will be inherited by the royal elephant. But frame system allow you to overwrite such properties. So you could say, color white. So you could think of this as default [Time 21 43] and overridden by this value [Time 22:01]. And then I could have another frame, so this is a individual frame because i have use lower case letter. I could have this as instance of [Time 22:45] and so on and so forth. So what do i know about clyde. We know that clyde is a royal elephant. We know that clyde is a elephant. We know that clyde is white as in the frame system we are talking about you can override the default and we can say that clyde have tow eyes and all that stuff you can get from inheritance.One can inherit from multiple sources. For exmaple, i could have said, royal elephant is an african animal, and african animal is a concept of all those animals found in africa. African animals could have certain properties which are different, which would be inherited by royal elephant inherit properties from here and as well as from african animal essentially [Time 24:29]. So one might ask what if we are trying to inherit two different properties from two different ancestors. This is tricky question, we will defer for the moment and we will try to see, if we have have decide based on the topology of the graph that we are constructing what is right thing to inherit. This is the thing which need a thought, we will defer it to later but as of now frames system do not restrict, they do not check for consistency so for example, if I have said that royal elephant is a fish, nothing in this whole system will crop up to say that no no no fishes are not elephant and all that stuff. We will look at something called description logics may be next week sometime in which we will be able to spot these inconsistencies.

I am rushing through this and I would expect you to read chapter eight from Brachman and Levesque where example I am going to talk about is discussed in great details. I am going to illustrate what is the idea behind this kind of frame structure essentially. The basic idea is that generic frame contain knowledge and they are used to create individual frames. You can think of short description i have given you, something like a personal travel agent that you have implemented a system where you want to plan a trip to a couple of cities and come back, then you start invoking



this system and system kind of help you along essentially. System is like a structured spread sheet or see normal excel you use is flat structure where you relate one cell to another cell. But here you are talking about a more structure knowledge representation, where you have type of things, taxonomy and all that. and you can store sort of store all this knowledge in a compact fashion essentially. When you talk of trip, we have something called a Trip, one generic frame. There are other generic frame like TravelPart and Lodging and Stay and so on. So these are generic. and this generic frames will be used to instantiate individual frames essentially. I am planning a trip, I want to go to a couple of cities. I will invoke the frame system. I will say something like trip27. My trip number is 27. 27th trip essentially. So moment i say I want to create a trip 27 so, i will start of by saying that it is a or sorry it is a instance of Trip. and moment I say this , the system should create FirstLeg and point it to another individual frame which say travel, something like first leg [Time 30:45]. It should prompt me for date, start date, destination, and lets say I am giving a few things, Next step essentially or next leg. So the system should prompt me, you want to go on this trip, what is this value. What is destination? What is going to be the next leg? [Time 31:00] Let us say I fill in some date 27th March, and lets say Delhi and lets say next leg is Chandigarh. The moment I fill in these three things, may be it will ask me next date also. Lets say 29th. The moment I have provided this information. The system should created lodging step, lower case, of trip 27. Lets say it figured out I want to travel by Air, so lodging stay should being on 27th march and it should end on 29th march and so on so forth. So this is the basic idea. In the generic frame, you know you can store information like how you pay for travel. Whether you pay by credit card or you pay by travel agent who give credit. Which airline you normally prefer or which hotel you normally prefer. If I go to delhi, then system might know that I stay in particular hotel and it should try to fill all that stuff here and essentially help me plan my whole trip. It should know that there must be return leg. So it must remind me that I can not get stranded. So It must automatically create a return leg and at that it should stop creating lodging stay and things like that. It should look at stay, for example, I am arriving Chandigarh in the morning and leaving in the evening, then it may not create lodging step.

All this stuff can be created by IF-ADDED and IF-NEEDED procedures, that we have just spoken about essentially. I will not go into details as it is very descriptive. I will expect you to read this chapter eight where this is described in considerable details. In next class, I want to move over to another application of frame which is used in natural language processing, which is also tied up to our study of conceptual dependency. So this is a idea which came from different school, which is this idea of Scripts and we will look into in next class.