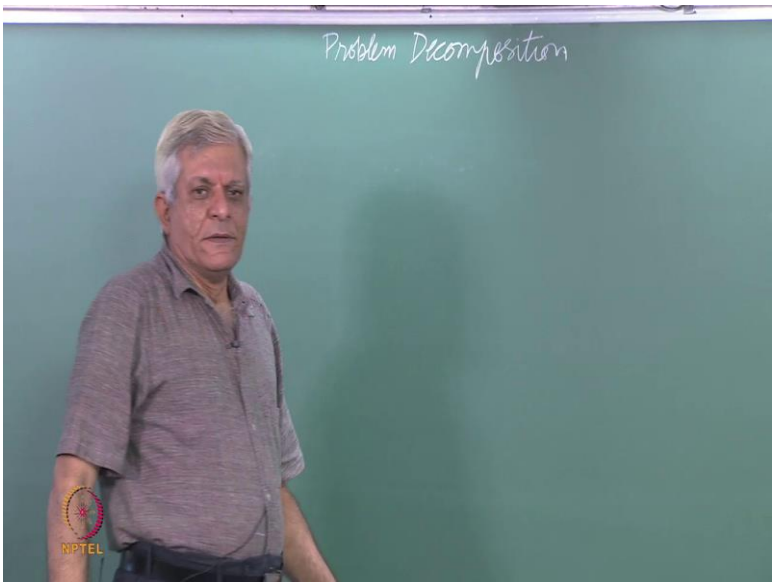
Artificial Intelligence
Prof.Deepak Khemani
Department Of Computer Science And Engineering
Indian Institute of Technology Madras

Lecture - 24
Problem Decomposition With Goal Trees

Today, we take a slightly different approach to problem solving. So far, our approach has been that we are in some given state, and we want to find a path to the goal state or something like that, or we are given a description of the goal state and we want to search for a state, which satisfies that description. So, today, we want look at a slightly different approach, which is kind of, has a flavor of backward reasoning, in the sense, you reason from the goal that you want to achieve, and try to break down the problem into smaller problems and so on, essentially.

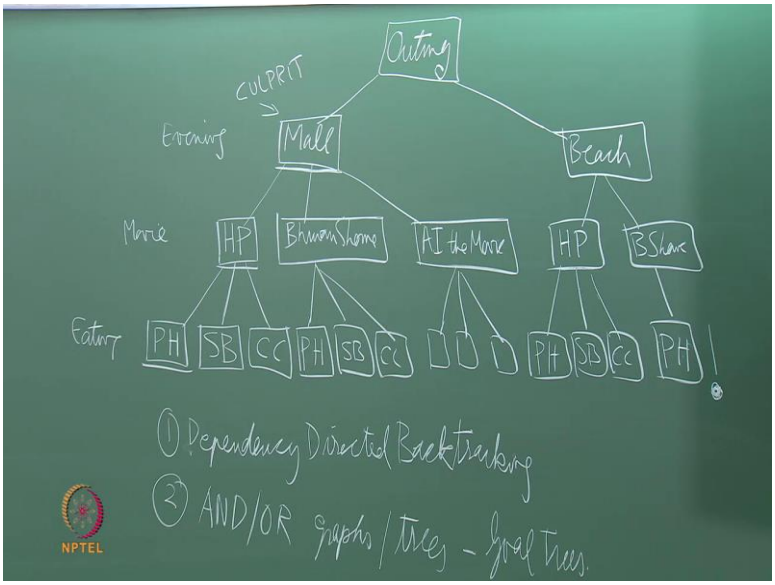
(Refer Slide Time: 01:06)



So, the idea that you want to explore is called problem decomposition. To motivate, why we should use this approach, let me try and take up a problem and solve it in the approaches that we have seen so far, and let us say that you have a friend, whose birthday it is. You have all decided to go out and let us say, give a treat to your friend. I know that is not the normal thing, but let us say that is what you are doing, essentially. Anyhow, decided that you will go in sometime, somewhere, in the evening. Then,

maybe, you will go to a movie or a music show or something, and then, you will have dinner somewhere, essentially. So, that is the idea of planning in evening, and let us say, it is friend, whose birthday it is, who is yet, to decide what is acceptable. So, you start querying, essentially.

(Refer Slide Time: 02:28)



So, let us try to visualize this as a search, in a simple search space where, you give different options to your friend. Let us call this problem as designing an outing, and the goal state is, when you have decided upon the three things, that you will do; what will you do first; what will you do next; and what will you do after that. So, let us say that you start up, by saying that you will go to a mall; very popular thing, nowadays. Then, maybe, you will go and see a movie. Let us say, I will use HP for Harry Potter, one of the Harry Potter films. Then, you will go and eat somewhere. Let us say, pizza hut. So, I will use PH for pizza hut. So, you present this option to your friend, and he or she says, no. That is not acceptable. So, let us say that you are working in a depth first fashion, essentially, or something similar to depth first where, this may be, heuristically guided or something, like that. So, you try something else, and let us say, you give another dinner option.

Let us say this well known restaurant in Chennai; Saravana Bhavan, and your friend

again, says no, essentially. Can you think of another restaurant name? Let us say Coffee Cafe Day; you can go and eat there, which I will use CC here. Your friend says no, again, essentially. Let us say you are run out of your budget and your restaurants and so on. So, you go back and say, let us try another option. So, just to clarify, this is the evening level, so, to speak. This is a movie or entertainment, and this is a eating out. So, there are three decisions, you have to make. Let us say, you offer to Bhuvan's Home, which is a very nice movie, as most of you might know, but and then, of course you cannot ask at this level, yes. So, in the approach that we have followed so far, we will consider every node and ask, whether this node is a goal node or not, essentially. This cannot be a goal node, because it has only two things; the third thing is not there, essentially, but there are other approaches. For example, when you do constrain satisfaction, then at each stage of the partial solution, you can inspect the partial solution and say, is this consistent with what I am looking for or not; but we will not get into that at this moment. So, again, you give these offers. So, pizza hut; answer is no, Saravanaa Bhavan; the answer is no, Cafe Coffee Day; the answer is no, essentially; then, you try one more movie. Let us say, A I, the movie. So, you must be knowing, there was a film called A I, the movie. Then, you try these three options, again. I will not write them, but these are the same three options; what you are trying, and your friend again, says, no, essentially.

Let us say at this stage, you have run out of movie options as well. So, you back track from here, and you say, shall we go to the beach. Then, you try all these same things again. This entire tree that is below mall; you are going to explore below beach, again, essentially. So, let us say, go to the beach and then, go and see harry porter; and then, go to pizza hut; then, go to Saravana Bhavan; then, go to Cafe Coffee Day; then, go to Bhuvan's Home; then go to let us say, pizza hut; and suddenly, your friend says, yes. You have found the solution; your friend is agreeable to the idea of going to the beach; then, going and watching Bhuvan's Home, and then, going to the pizza hut for dinner, essentially. Now, if you look at this search tree, I want you to look at this, and make some observations; is it a good way of searching, or what is happening in this tree?

.Many nodes are replicated, or let me ask a more specific question; what is wrong in the left side of the tree? What is it that is not working in the left side of the tree? Where is

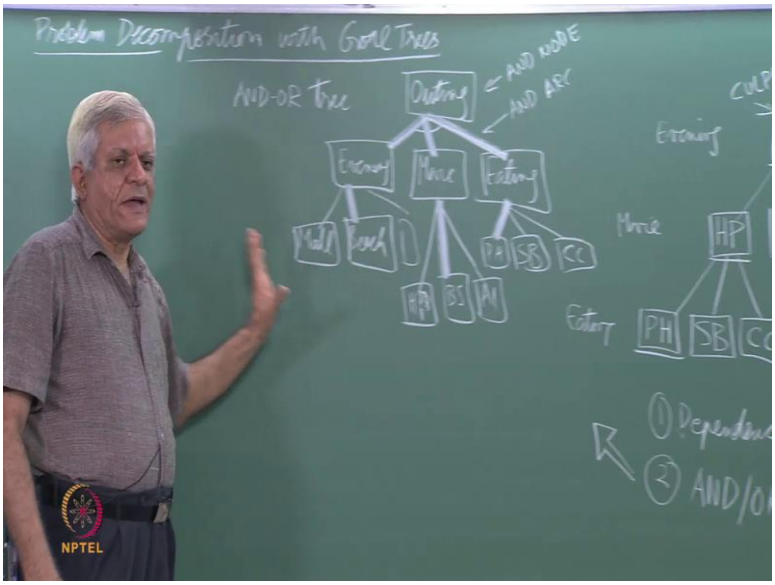
the fault in the solution that you constructing, or what is wrong with the solution that the left side of the tree is trying to construct? What is, or which part of the solution is not working? Can you look, inspect the tree?

Because you explored the entire tree below mall, and I have drawn three movies and three restaurants; they could have been many more, essentially. There could be a greater depth, because we have explored the entire tree below mall, and it did not work. If mall was the problem, then you can see that if in the first solution that you considered, which is mall, Harry Potter and Pizza hut; if somehow, you could figure out that mall is the problem, then you would not do all these wasteful work, here; trying to see if not pizza hut; then, Saravanaa Bhavan, if not Saravanaa Bhavan, of course, then, Cafe Coffee Day; and if not Harry Potter; then, Bhuvan's Home, if not Bhuvan's Home; then, A I, the movie; all these work is wasted work, essentially, and the reason is, because the culprit is here, and keeping that choice fixed, and search it below the tree, is not going to help, any longer. Anything you try below this is not going to work. If you figure out that mall is creating a problem, essentially. It turns out that you go this side, and then, you get a solution, essentially. So, how does one solve this problem of doing, what is the issue that we want to address is that; you tried one thing and then, you are doing this useless work, exploring this part of the tree, which will loss and not going to work. Only, when you change this, it is going to work, essentially. So, there are basically two approaches to addressing this difficulty. One is called Dependency Directed Back Tracking.

Now, the backtracking that we are doing here, in most of the algorithm that we have seen, is chronological in nature; that you undo the last choice that you make. So, you tried this solution. It did not work so, you undo this last choice, which is pizza hut and try the next one, at this level, which is Saravanaa Bhavan; undo this and try this. When all these three fail, then you undo this and try this. This form of backtracking is called chronological, which mean that you undoing the last choice that you are making. What we try to do is dependency directed backtracking, is to identify the culprit of what is wrong with the solution, or in other words, without changing that attribute, you are not going to get the solution. If you can identify, after coming here, or after coming somewhere that the culprit is the choice of evening twist, went in the mall. Then, the algorithm should actually, directly jump back to this, and try the next choice, essentially.

So, we will not really, explore this option, here, unless we get time later on, but it is something that we study, when you look at constrain satisfaction method, essentially. But we are not going to do too much of constraint processing, here. Constrain processing is just a different formulation of solving such problems in which, you formulate the entire problem in terms of variables, and values, that variable can take, essentially. The other approach that you want to consider, today, is to construct what I call AND OR graphs or trees. We could have converted this into a graph by, instead of generating these three, I could have connected them to these three, and then, it would have been a graph. So, there is not really, too much of difference, whether it is a graph or tree; just a way of looking at it. These are also known as goal trees.

(Refer Slide Time: 13:33)



We all want to look at problem decomposition with goal trees. The idea of problem decomposition is something, like using a mechanism to break up a problem into smaller parts. In this example, it is a little bit, like using a context free grammar, to say, that an outing is made up of an evening out, and the movie and dinner, and then, so on and so forth; but it does not have to be context free. In this case, it could be context dependent. So, how do we do this? This approach is to view the problem differently, which is to say that you have outing as a; this is the goal that you trying to solve. Now, you are going to break it up into three separate sub goals. So, one is evening out, or evening plan. Then,

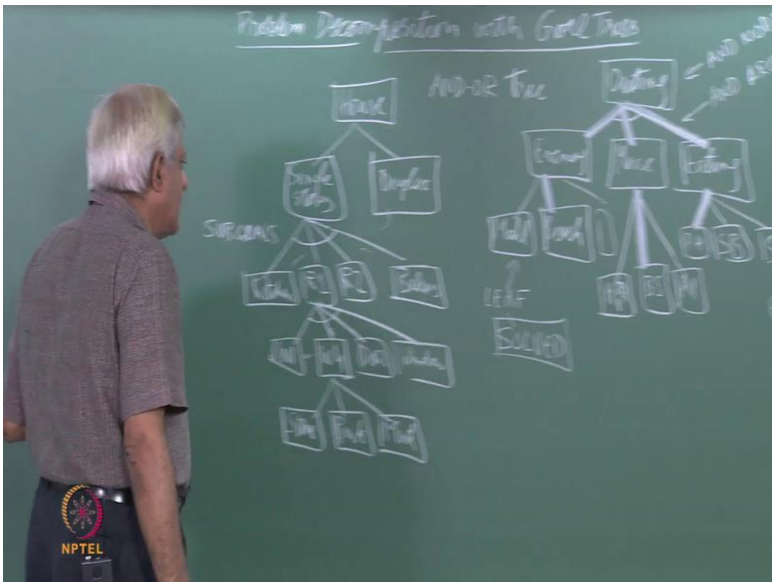
movie, and then, eating on dinner. So, this is the different representation. To distinguish between this sort of tree and that sort of a tree, we put an arc here, for our benefit.

Such an arc is called as AND arc, and such a node is called an AND node. So, an AND node is a node from which, the choices, all the nodes that it points to; it is a directive graph. Every one of them has to be solved, as opposed to such a node; this is called an OR arc. This view is called an OR arc, or this, would be called as an OR node. So, an OR node has all OR arcs below it. The meaning of an OR node and OR arc is that you have to do one of the things below that. For example, you have to either, go to the mall or you have to go to the beach, or if you had decided upon the mall, then you have to either, choose Harry Potter or Bhuvan's Home or A I, the movie, essentially. That is the node. So, we have those all choices here, as well. So, let us say, mall and beech, and may be, some other choices; does not matter. Then, we have the three choices for the movie, Harry Potter, Bhuvan's Home and A I, the movie. Then, we have the three choices for the restaurant. So, such a tree is called an AND OR tree, because it has AND nodes as well as OR nodes in that. The AND node is at top level. OR node is below that, and a solution in the OR graph like this, the kind that we have been studying so far, is the path. The solution in an AND OR tree is a sub tree. So, a solution will have, for example, this and this and this and this. So, the solution for this same problem is a sub tree here. The solution in that combination was a path. The important thing is that you have broken the problem down into three smaller problems, and solve them independently, of each other. This means that when you know that you have to choose between mall and beach, you do this independent of the other things. Whereas here, one should choose the mall, then you are saying, even the mall; what else should I search for?

Now, that does not really make sense, because mall is independent of what you are going to do after that, and this formulation of AND OR tree, allows you to break up the problems into smaller parts, and solve it, essentially. So, we want to study approaches to solving algorithms that we will work on such formulations, and solve problems, like that, essentially, but first I want to give some motivation about, where it has been used and so on. Now, obviously, there would be cost, if you going to do some kind of optimal solution, finding there, would be some kind of cost associated with it. In this example, cost could be, for example, mall may have some cost associated with it; may be, the

distant you have to travel, or sometimes, some malls now; they charge some fees to some people, but I do not know. Obviously, movies also will have charges; eating out will have some costs, and you may have some criteria; you want to those, find the solution within this cost, or within this budget, or of optimal budget and things like that. So, we will worry about cost as we see this later.

(Refer Slide Time: 19:30)



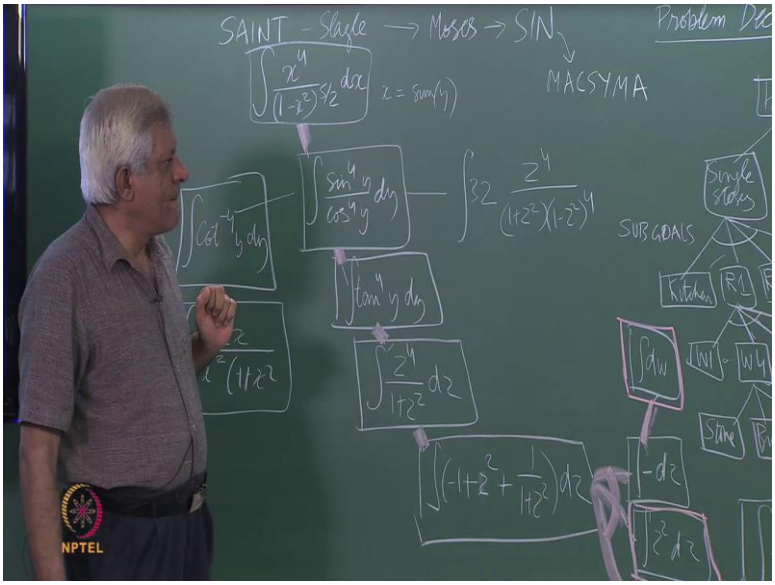
Let me take another example. Suppose, you want to construct a house, you can see this also, you can pose this problem also, as an AND OR problem. So, just to take some simple variations, you might say something, like single storey or a duplex; these might be choices. In a single storey house, may say that you have to make a kitchen, followed by, let us say, room one. So, I will just say, R1 followed by, let us say R2 and so on, and let us say, balcony. Now, this, of course, would be an AND node here, and I could have edges coming from here. For example, duplex house also will have a kitchen and so on, which I can always, draw it as a graph or I can draw it as a tree; it does not really matter. Then, may be, room 1, would say, wall 1, up to, let say, we have 4 rooms; we have rectangular rooms. So, from wall 1 to wall 4 and then, may be door; let say, we have only 1 door in this, and let say, we have a window.

This would be an AND node, and if you look at a wall, for example, you may have

choices, like stone or brick or mud. In this manner, the choices that available to you to construct a house, can be organized into an AND OR tree where, the top level, you have the high level goal, which is to construct a house. Then these, at lower levels; this is just as choices here. At lower levels, these are sub goals. Then, you keep breaking down. Till what extent you break down a problem? You break it down, till the problem is so simple that you have a solution, already available, essentially. So, I will illustrate that with the next example that we see, which is from one of the first systems that were to use AND OR graphs, essentially.

Now, keep in mind that they are two kinds of cost that are going to be involved. One is cost of transforming a problem. Let say, we assumed, there is a cost of doing that into sub problems. The second is the cost of solving the sub problems and so on. So, at the end of this, these nodes are called leaf nodes, I mean, the leaf nodes are actually, called solved nodes. So, the leaf nodes in the tree would be solved nodes. Solved nodes, of course, would have an associated cost with it. Depending on the domain, we will see a domain; we will start with looking at symbolic integration, for example, which was one of the first applications, which used AND OR graphs. The solved nodes may not have any cost associated with it. For example, you know that integral of XDX is something that you can just pick up the solution, or X square DX, you can just pick up the solution and things like that, but if you look at the problem, like building a house, then if you going to build 4 walls, then each of those four walls, when you build it; it will have an associated cost, and then, you will have to aggregate all that cost into the cost of building that house, essentially. So, solved nodes may or may not have cost, depending on what is the nature of the problem, but transformations will always, have some costs associated with it. That is the way we will look at the problem, essentially, because you want to control the amount of search that you do.

(Refer Slid Time: 23:55)



One of the first programs to look at that AND OR trees, was the program called SAINT, which was implemented by a guy called Slagle in 1961. When you are looking at the introductory lectures and we are saying, what are the achievements that A I has done over the years; we had mentioned symbolic integration, at that time, in 1961. I think in MIT, he wrote this. This was his PHD thesis on how to do symbolic integration. So, what do you mean by symbolic integration? That you are given some expression, and you want to integrate that expression, in definite integral and you want to convert, you want to produce a expression, which is the integral of this expression, essentially. So, my mathematics is a bit week. So, let me, yes, use this break up, which, as strong as thesis. So, let us say that you want to integrate X raise to 4 divided by 1 minus X square raise to 5 by 2 d x. I am sure that you people are more familiar with integration than I am.

You know how to solve such problems is that basically, you apply some transformations and may be, you break up the problem into smaller parts, something that would fit very nicely into the AND OR situations, because you have a choice of transformations to make. So, first let us look at what the solution, look like. So, this is the node, when you can transform it into sin raise to 4 Y Y; do not ask me how, but think for it, which we can. So, the transformation is X equal to sin Y. So, eventually, when you break it down into something, which is simple, which is accessible to you; you will have a sort of, invert this transformation as you construct the solution for this, essentially, I am in a risk of converting this into a Maths class; let me try one or two more steps.

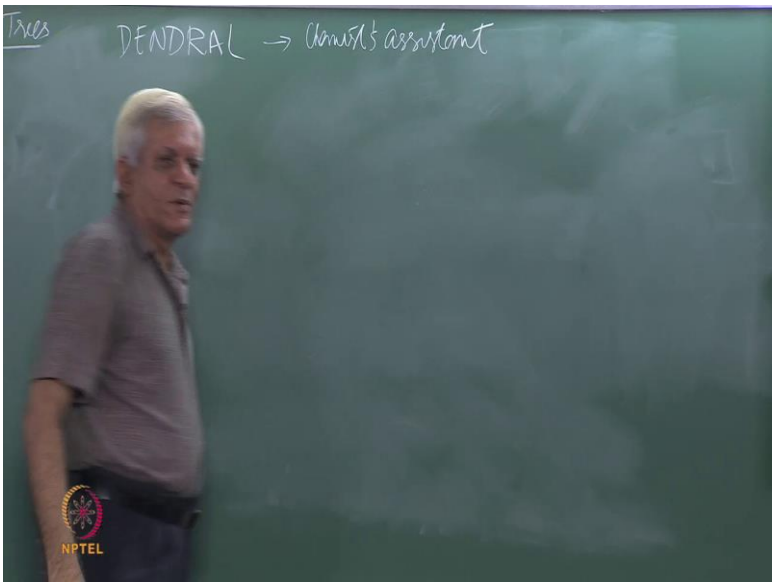
You do not have to really write this. Notice that here, we have replaced something like Z equal to tan W, let me write it here, and transformed into this, and likewise, this can get transformed into d w. So, now, we are assuming that this node is a solved node, in the sense, that you can do this without any difficulty. This node is a solved node, and this node is a solved node. So, you have the solution now. So, this is an AND node, of course, and this so. The solution to this integration problem is a series of transformations that you do, followed by, breaking up the problem into smaller parts, because we have this addition here; these smaller parts, and then, we can solve this integral DW integral Z square and integral DW, very trivially. So, we will treat those nodes as solve nodes. So, that is what I mean by saying that the cost of finding such nodes, the cost of this solution is; you can consider it to be small, but that cost of transformation can be counted essentially.

Of course, this is not the only way of doing this, essentially, and there could have been other choices. So, for example, you could have said that transform this to cost minus 4 YDY, and transform this into something else, DX, and may be, it does not look so promising or something of even, worst you could try something like this. So, to solve a problem like this, there are many options. Those of you, who remember doing your Maths or some entrance exam or the other, you would remember that you have to break your head; try to remember, what are those transformations and so on. That is why, this needs a lot of practice, but what this program SAINT; SAINT stands for Symbolic Automatic Integration.

Later, a guy called Moses developed a program, which he called as SIN, because this was called SAINT; which stands for Symbolic Integration. This SIN was eventually, transformed into a product called MACSYMA. Nowadays, of course, you must be familiar that we have these mathematical packages, which can do symbolic mathematics for you. These are all the descendants of this SIN and SAINT, essentially. Some MATLAB, for example, will also allow you to do symbolic mathematics. So, what you want to show here, is that symbolic integration can be seen, as solving a problem in this manner where, you reason from your large level, higher level problem to break it down into sub goals, till sub goals are simple enough, to be solved trivially, and then, you have a solution for solving this problem. Of course, actual answer to this, you have to undo all

the transformations that you have done, which I have not written here. So, let me give another example where, this idea was used effectively. We had mentioned that in the 70s, was the area of what we call as expert systems.

(Refer Slide Time: 33:06)



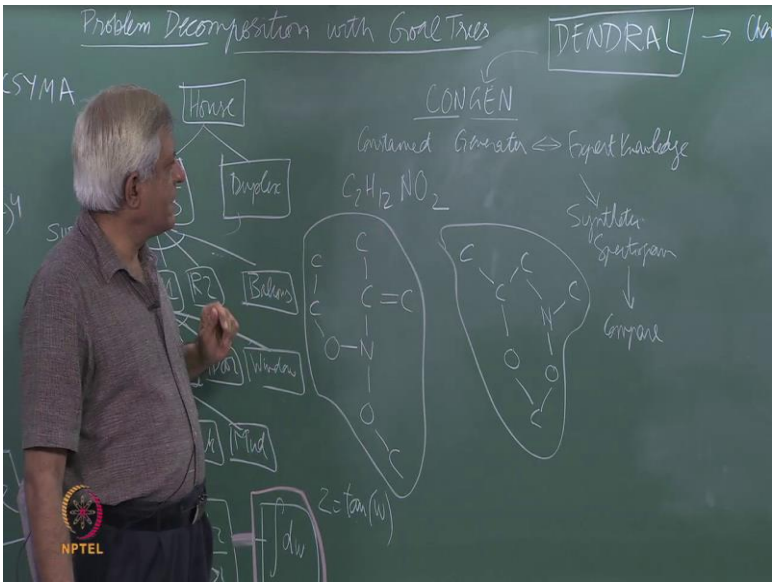
One of the first expert systems was this program called DENDRAL. This DENDRAL, basically, also can be, some people say you can extend it to a dentritec tree X algorithm or something like that, because it generated also, an AND OR graph, which look like that, but the idea of DENDRAL was that it was the assistant; it was originally developed or programmed it as an assistant to a chemist, and the task that the chemist was doing was to determining the structural formula of an compound, essentially. Now, you know that again, I presume your chemistry is better than mine. So, you know that if you are given a molecular formula for a compound, for example, C6 H6; how was it C6 H12; whatever, benzene; what is the formula for benzene? C 6 H 6, and you must have heard the story of Kekule, and you know how people were trying to figure out, what is the structure of benzene.

So, at a molecular level, of course, it has 6 carbon atoms and 6 hydrogen atoms, but the question to ask is; how are these atoms arranged, in a structure? That is a very important question, because the physical and chemical properties of the substance, depends on the

structural formula, that it is behind, essentially, and to find the structural formula, is not such a straight forward task, and apparently, Kekule was dreaming, or day dreaming or sleeping, or something, and he saw this snake, which was biting his own tail, and then, it is benzene ring and so on, essentially. So, the idea behind DENDRAL was to help a chemist, find structural formulas and apparently, people who have done PHD in chemistry, spend their considerable amount of time, trying to find the structural formulas of this thing. The problem is not so simple, because a given molecular formula may have millions of different structural formula, associated with it, essentially, for larger compounds, obviously.

At a must lower level, you know, the structure plays an important role in materials, because whether you are holding a diamond in your hand, or a lump of coal; is just a matter of structure. You, basically, holding carbon in your hand, but it is really the structure of it, which gives it the properties that you looking for.

(Refer Slide Time: 36:17)



So, the way that is DENDRAL was implemented, the idea was, it will explore the space of possible structures, and this can be done by breaking down a molecular formula into smaller parts, and exploring the structure of the smaller parts. We will see a small sample example, but this was aided by a process of generating a synthetic spectrogram. So, there

was an algorithm that given a structural formula, what will be the spectrogram of that material look like, essentially.

Once you can do that, you can compare it with a real spectrogram of the material, and if the spectrogram matches, then you can say that you have found the structural formula for that material. That is the approach with chemist for doing it laboriously. DENDRAL was a program; it was developed by Stanford in 1971 or something like that, came a bit after SAINT. It was touted as the first expert system, which was built, and 70s and 80s was the era of expert systems where, people said that we will build expert systems, and by this, they meant that they will elicit the knowledge of experts, put it into a program, and the program will then, perform at the level of an expert. So, apart from DENDRAL, in the introduction, we might have mentioned things like prospector, which was an expert system to find a prospect for oils or minerals, and so on.

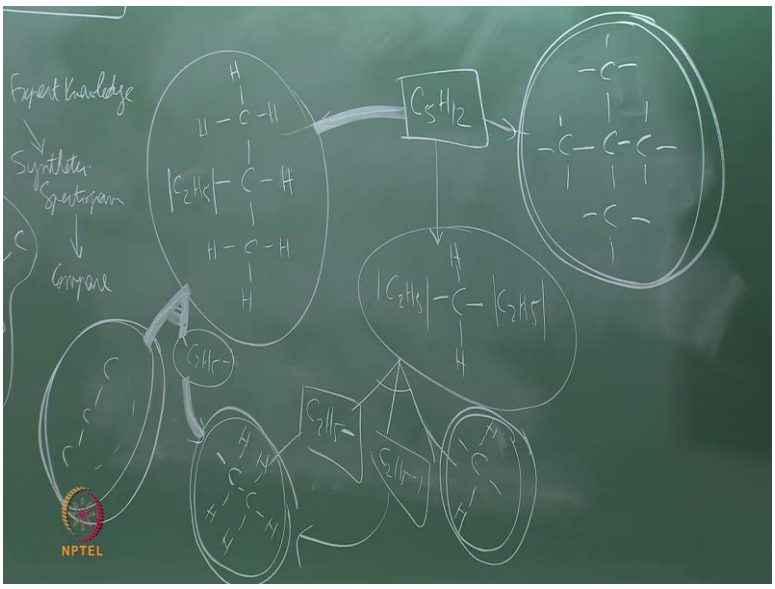
There was a program called R 1, which was also at Stanford I think, which was used to configure wax systems. In those years, of course, buying a machine, a computer was not such a straight forward task as saying that I want Apple, Mac book or something like that. You had to actually configure various components together, and R 1 was an expert system, which helped people, configure computer systems. We will look at this; how this R 1 and things like this; such programs have built, a little bit later in the course, but today, we want to focus on this A 0 star, kind of an algorithm, or AND OR tree algorithms, and which basically, search over AND OR trees. So, where did the expertise come in?

Expertise came in that there was first a program called CONGEN. So, DENDRAL was made up of a program called CONGEN, and this stands for Constraint Generator. This had expert knowledge inside it. So, of the many possible ways that you could conjure a structure for a given molecular formula, CONGEN was guided by knowledge gleamed from chemist, as to what structures are feasible and what are not feasible, essentially. For example, given a simple thing like C 2 H 12 NO 2, this CONGEN would produce a structure like this, or it would produce another structure. So, this is one, and this is another one, and there are more examples, which I am not drawing here.

Essentially, again since, you are better at chemistry than I am. You know that this is double bond and this is single bond, and so on; we have not drawn the hydrogen atoms and you can fill in the hydrogen atoms, based on the valance here, remaining. So, 3 here, for example, and 2 here, in that, but these are different structures that CONGEN generates and what was interesting about CONGEN was, it generated only feasible structures. That is why, expert knowledge came into that.

So, after CONGEN, you produce a synthetic spectrogram, compared with real and then, you can decide, whether you have found the structure of the material or not. What DENDRAL would do is that it would expose this huge space of all possible structures, generates this synthetic spectrograms, and compare it with the one of the original material, and eventually, do the job for the chemist, and it turns out that this was working, of course, much faster than real chemist, as you can imagine; and doing equally, well or almost, equally well and when you say real chemist means people, who have PHDs essentially. So, let me just end with a small example of the kind of space that DENDRAL generated, and in the next class, we will see the algorithms, which I used to explore this space.

(Refer Slide Time: 42:37)



Let us say that we are looking at C5 H12. This is a molecular formula, and any box,

which has a molecular formula, is an unsolved node or which, may be partially solved, as we will see. A solved node is where; this structure is entirely given to you, essentially. So, for example, it could be like this; C, C, C, C, C; and then, the hydrogen items. So, this would be a solved node. A simple way of organizing this 5 carbon atoms and 12 hydrogen atoms, but there could be other options, essentially. For example, these are directed graphs. You could say that it is C. So, a partially elicited structure where, part of the structure is known, and part is not known. So, 3 carbon atoms are here; 2, I buy it inside here. This is another partially elicited structure, and this itself, would have now, three children, which this part, this part and this part. So, you solved for C2 H5 with a bond. Then, you have another C2 H5; it is a bond and then, you have C, H, H, H, H, H, with two bonds, and this is solved; and this can be solved by, and this also, can be solved by this, and this will have, for example, two components. One of them will be the C2 H5 with a bond, which will be solved by this; and the remaining will be that; whatever remains here; C, C, C, and so on.

So, in this very quick chemistry lesson, you can see that three possible structures for C2 H5; one is this structure; one is a structure where, you have the middle part where, you have C and 2 hydrogen atoms, and the two edges contain this structure like this, and likewise, this is another structure where, the middle part has the C2 H5 component. Here, the two edges have the C2 H5 components; obviously, that matters. So, you have either, one solution is this one. This is an AND node. Another solution is here, and the third solution is that side, essentially. Now, what DENDRAL did is that it navigates this space intelligently, in the sense, that this CONGEN, component of DENDRAL, only generates those nodes, which are feasible in practice. How does it do that? It has got a lot of expert knowledge built into it, and that is why, this system is called the first expert system that was built, essentially.

Then, of course, it generates the synthetic spectrogram of the, for example, you take this generate the spectrogram and compare it with your real material spectrogram, and if this matches, then you know that it is the structural formula and so on, essentially. Today, we have just introduced the idea of AND OR trees or AND OR graphs. So, we will not distinguish too much between them. We have just very briefly, talked about the cost associated with solved nodes and which, have to be aggregated into the root node. So, in

the next class we will look at an algorithm, which explores an AND OR tree or a goal tree, to find an optimal solution, essentially.

So, we will stop here, now.