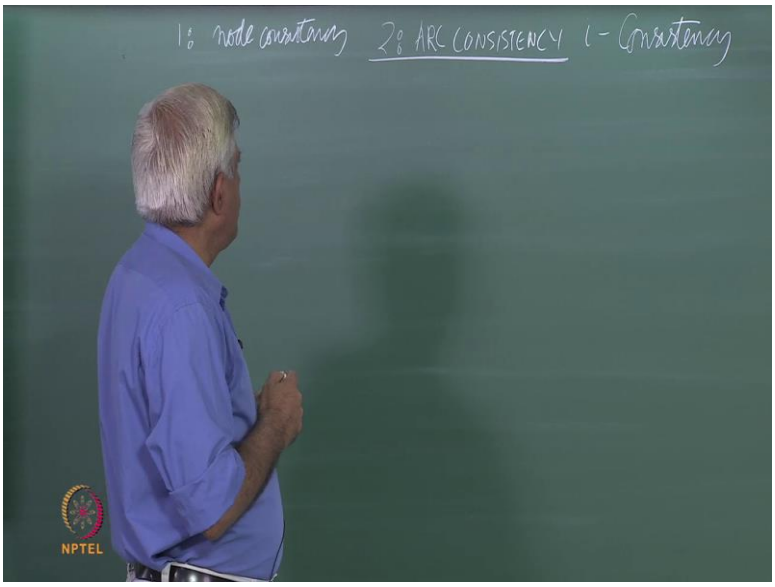
Artificial Intelligence
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 40
CSP Continued

So, in the last class, we saw the definition of constrain satisfaction problems and we made this observation that solving the C S P basically allows co-operation between two kinds of algorithm, one, which the search over possible assignments and the other which does some kind of reasoning which we will call as propagation. So, we have this notion of consistency.

(Refer Slide Time: 00:36)



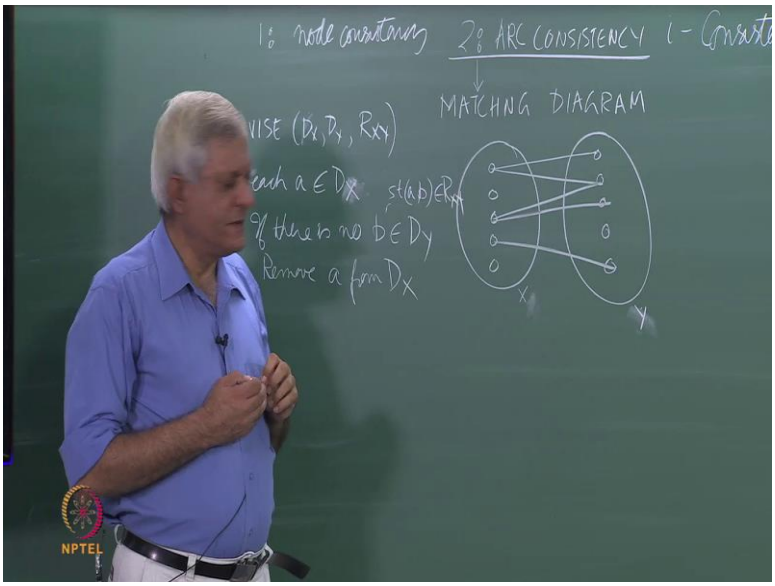
We are, this notion of I consistency, so we say that a C S P or a network is I consistence, if every consistence assignment to I minus 1 variables can be extended to I variable which means that, if we have found values for I minus for 1 variables any I minus variables. Then, you can always take any I variable, any next variable and extend find the consistent value for that. So, of course not all networks will be I consistence, but the general effort in reasoning is to enforce consistency in some manner essentially.

So, we can start with the very simplest notion of consistency, which is called node

consistency or one consistency, when you say one consistency then we use the term note. So, it basically means that you take any variable and you will find one value which is consistent. What do we mean by consistence with one variable? That if there happens to be a constraints over only that variable, a unary constraint then all the variables satisfy that. So, we can enforce node consistency by simply saying that if there is some value which does not satisfy constraint, remove it from the domain.

So, in general node consistency and there is one consistency, two consistency will domain which essentially. So, two consistency, what we will look at it cal A R C consistency and what two consistency says that you take a assign a value 2, any variable which is node consistence which means that values satisfy any unary constraint that there might be, you take any other variable and you will be able to find a value for that variable which is, which. So, that there is a 2 values, for the first variable and second variable are together consistence essentially.

(Refer Slide Time: 03:19)



So, to talk about A R C consistence is often useful to draw, what we call is a matching diagram, and the matching diagram basically does something like this, that it creates a domain for each variable. So, this is X 1, this is X 2 and then you have variables inside is domains, so these are the values that this variable can take and edge represents the fact

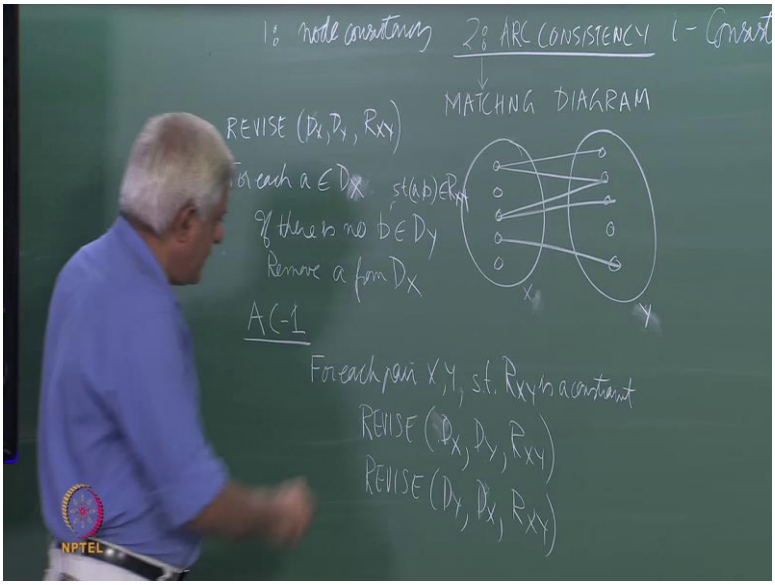
that this value, let us call it A 1 and this value let us call it B 1 belongs to the relation essentially. So, basically it is a depiction of relation essentially, so I whatever the relation is something like this, so such a diagram column is matching diagram.

So, what we want to do in our consistency over these two variables is to keep only those values in the domain, for which there is a corresponding value in the other domain essentially. So, for example we do not want to keep this value because there is no matching value in the other domain, we do not want to keep this value, we do not want to keep this value. So, to implement this, we have a simple procedure called revise and this is kind a standard name for it D X D Y or X Y it takes. So, we say that we are going to prune the values of D X to only those values which have a value corresponding value for this R X Y.

So, if this is X and that is Y we can prune in this essentially, so for each A belonging to d X, if there is no B belonging to D Y, remove A from D X very simple procedure which looks at each value. So, this is A, this is a procedure which is only pruning the domain D X, so it looks for each value this is A X and that is Y, and it looks each value of X. If there is no value in Y, if there is no B belong with D Y, I should have add it first, that A B belongs to R X Y. In other words, if we have a binary constrain over X and Y, then every value of X should have a matching value in Y and this procedure revise is pruning this domain X domain of X.

So, it looks at all these values and for A, every value where there is no matching value, on the other side this one, this one has two values. So, it, this one has no values, so it will remove this from the domain, it will keep this, it will keep this and will remove this. So, it will remove 2 values and keep one essentially, so the process of doing our consistency, basically doing this repeatedly so that all pairs of variables are our consistency essentially.

(Refer Slide Time: 07:29)



So, the simplest algorithm for doing that is call A C 1, so and, so our consistency 1, and it does basically the following for each X Y, such that R X Y is the domain, R X Y is the real, is a constrain, call revise D X D Y, R X Y and call revise in both directions D Y D X R X Y. So, we will assume that R X Y contains the relation between them, so I instead of writing R Y X, I am just writing R X Y, for each pair X Y of variable we want do this which means if I have 3 variables.

Let us see this one, and let us say I have something like this, so this is X Y and Z, so I have 3 variables, 2 relations between X and Y, Y and Z and I want to make this network our consistence which means that every assignment one of these variables can be extended to an assignment of 2 variables? Notice that, in the constrate network diagram that we had drawn earlier, we would have drawn this like this, this is Y, this is Z and this is X. So, there is A between these two, this is called a network from same network, it basically a constrate graph which says which variable is constrain, by which variable and we are talking about binary constrain.

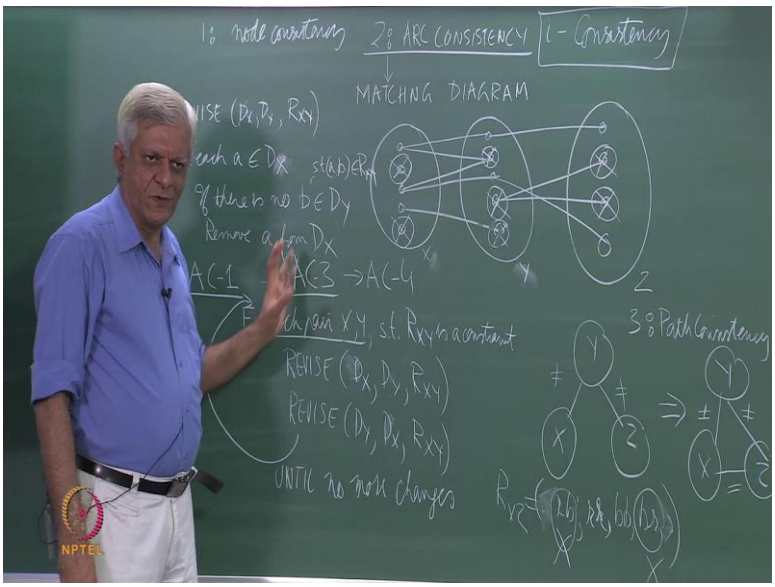
So, we have only edges, otherwise you would have hyper edges, the constrain of, basically identifies which variables are related to each other. The matching diagram tells you which values of variables, our participating in those constrains, when we say

something like, this that, this is a constrain graph. So, if you remember this map coloring example that we saw, we said that if this is, if the domains of all 3 are red, blue and this is not equal to this, and this is not equal to this. So, that is a constrain given to first, implicitly we have a constrain between this and this which says that anything is allowed.

So, R B is allowed R, R is allowed, D B is allowed, and B R is allowed, if we have not specified, if we have not specified a constrain, that means implicitly it is a universal relation. It means anything allowed to anything which means, what that our back trekking algorithm said that if you first want to give a value for X, and if you want to give a value for Z, it is not putting constraint which means you can choose. If we choose R for X, you can choose B for Z, it does not care because it does not know any constrain essentially.

So, there is the universal relations, but of course this not expressed, it simply sit, so it does not participate in the matching diagram here and it does not participate in this. So, when you say R X Y is the constrain, remain it is an expressive constrain mention in the C S P essentially.

(Refer Slide Time: 12:02)



So, before I come back to this, we have a notion of 3 consistency which is known as path

consistency, remember the notion of I consistency which is the generic notion. So, 3 consistencies say that any pair of values which are constraint can be a extended to a third value. So, if I choose 2 values for this, I should be able to extend to a third value, so if I choose Z for this and blue for this, I can extend it to blue for this.

If I X, if I choose blue for this and red for this, I can choose blue for that, but if I choose these two values first, if I choose red for this and red for this then I can extend this two blue for this variable Y. But, if I go to choose red for X and blue for Z, then I cannot extend to Y, so this network is given to us is node 3 consistence or is node path consistence. What path consistency would do, would to prune this relation in this example?

So, what we do? It do that it will say that you choose red for this and blue for this and it cannot extend it to the value. So, this red blue must be removed from this relation you choose blue for this and red for this and you cannot extended to a value, so you must prune this from the relation. So, enforcing path consistency prunes relations the moment you have pruning a universal relation, you have making a express it, so in fact it adds a new relation. So, when you do path consistency you get a new network, it looks like this, it adds the relation in some sense, in the sense that earlier it was a universal relation, but now it is not a universal relation.

So, you have to express it, represented which means X is not related Z, so these two are not equal to, but this one is equal to, so we have new network. So, the general idea of enforcing consistency is to limit the choice is a available to a search algorithm, to only those which are like you to participating and solutions. But, since an N variable problem will have N variable, so we, to really achieve full consistency you would have to achieve N consistency, which off course is you can take it from me is a hard task.

So, which is why we do not often do it, so very often algorithms will do ache, some degree of consistency, either node consistency or path consistency, consistency or path consistency, or some higher order consistency depending on how much, how complex the problem is. Then leave the rest to search and leave the rest to those other kinds of things that we just briefly mentioned in the passing like dependency direct is back

tracking or loop a heading search.

So, there are various tools in the, of a C S P solver consistency enforcement is one of those which is what we are trying to look at today. So, coming back to arch consistency, I want to make this network arch consistence, which means if I choose any value in any variable, I should get the allowed to by the consistency, given to me choose value for the next variable. So, obviously as you saw, when we call, what will we do? We will call revise with X and Y with Y and X with X and Z with Z and X with Y and Z and Z and Y. So, at least 6 calls to revise, we will have to make remember, revise is the directional call it prunes only the domain of the first one.

That if for each X and D X, there is no corresponding value in Y, remove this A from D X, that is what revise does for this. So, we have to make at least 6 calls to revise the other 6 call, enough is a question I want to ask you. If I say revise X Y, revise Y X, revise X Z, revise Z X, revise Y Z, revise Z Y, am I done, and do I get a network which is our consistence? Let us try it on this, so let us call revise X Y first, which means we are going to throw this away and we are going to throw this away from this essentially. Then that is called revise Y X, which means we are going to throw this away from the domain of Y, so let me actually circle it.

So, the N cross it, so we know that those things are not there essentially, then next called X Z, X Z we do not have a constraint at all, the constraint given to us is this. This constraint diagram which turns anything about, which means it is a universal relation which means there is no constraint, on choosing any value from there. So, this revise call will not do anything, but we have a constraint between Y and Z, so let us try that. For this, we have this value, for this we do not have this value, so we have to delete this for this, we have this value for this, we have this value, anyway this is not there, for this we do not have this value, so we have to delete this. So, this is gone, now let us see what happen the moment I have deleted this, this value

Well, this has one left here, but this related this value this one does not have a matching value, so I read something revise X Y. But, now when I did revise wise it, I deleted the value of Y and this value of X does not have anything left, so at least one more to call to

revise, I will have to make essentially. So, if you do this, then this one has this, this one does not have this, so this goes away, this one has does not have this of this goes away this one has this.

So, it is almost our consistence except for this value, here this value does not have corresponding value, here this value has, this value here and then this value has that value there, this value has this value here, and that is that, but this one does not have value here. So, I must make another call to X Y, so which means I must put this into a loop. What is the safest thing to do until no more changes, until no domain changes? I will keep doing this, revise all these, revise at the very outside, we can see calling revise X Z does not really make a sense.

But, of course we are not calling revise X Z, because the condition says for each were X Y. So, set R X Y is constraint, so R X Z is not a constraint. So, it is we will make calls revise X Y, Y X, Y Z and Z Y repeatedly, till we come to a condition where no domains as change because then we are show that it is our consistence essentially. So, is this algorithm, is it nice, is it a good algorithm? The answer is no, because it can be argued that the number of calls number of calls revise that you make is actually very large that in the worst case in every cycle you will remove only one value from one domain.

In the worst case, you can construct a network like there, so that in one cycle which means a complete set up revise calls, you will remove only one value from one domain, and in the next cycle, you will remove one value from one domain. So, if there are N domains, and let us say all of them are connected and each of them has K value, then you will make N into K cycles which is obliviously not in efficient things to do. The reason for that is that, why should we do this route force call to all combinations of revise.

Now, if we look at what is happening here, what we should have done is the following that when we made this revise at call, and then we deleted this value from the domain of Y, we should we have said Y has change. So, therefore any constraint participating in relation in which Y is there, look at that relation again because Y has change any X Y participating in this thing, so look at that relation again. So, which means when we delete a value from a domain then we should for future, revise call only those relations which

are participating with that variable essentially.

So, that actually leads just in algorithm which I will not describing any more details here which is called A C 3, which was describe a who is consider by many, to be in some sense, the big boss of constrain satisfaction. If you want to meet him, we can go to of Ireland, where there is the big constrain center and it probable the place which the largest population of the constrain researches essentially. So, there are algorithms for some AC 3, there is no, there is A C 2 is missing lost essentially, but this A C D 3 algorithm ,what I will just refreeze describe. What we does is that initially it maintain the cube of all possible revise called, but then it removes elements from the and it makes a call it add to the queue only if it beans it necessary that I need to check this relation again.

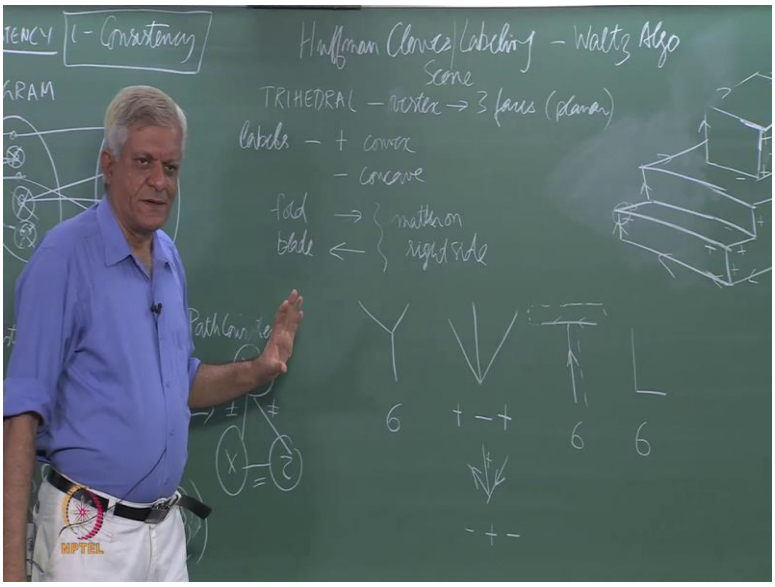
Essentially, which means if I deleted something from Y then let I should check this revise X Y call, again make this at least time as loop again if I did it something from in this case it is only Y. So, if I did something from y in a call from y to x then I must make a revise call to Z, Z Y again essentially. So, only if there is a danger of having lost some matching value I will make a call to revise again. So, I will put that particular revise call in the queue which means only where changes are matter are happening this constraint will it propagated in that sense that is why we call this constraint propagation.

We have to another algorithm which of course we will not discuss call A C 4 does not even make generic calls to revise, again it says this value has change in this thing. So, if this value was a matching value for some this thing some other, so let us say this value was a matching value. For this value in X, let me go and check if X still has a matching value or not, otherwise I will remove X from that. So, I does not even make this revise X Y call, again it is only looks at this particular value to see if there is a corresponding matching value left because it could have another value.

But, in which case, of course we do not have to delete it simply, because it does not have any value remaining after this has been deleted, this should be deleted. So, that is the still final level detail algorithm A C 4 and, obviously the complexity decrease as you go from here to here representation increases, we have to represent more things essentially. So, I want to talk about one particular constrain satisfaction problem where an algorithm,

which his claim to be some were between A C 1 and A C 2 or close to. What would have been A C 2? If it has been describe is use.

(Refer Slide Time: 25:20)



This is a famous problem know as Huffman close labeling and there is a well know algorithm call Waltz algorithm which is some were like is A C 2, one more less, so it is not officially called like that. So, let me first describe the problem, the problem is for scene labeling and when you say seen labeling we mean that we have a line drawing to available to us and you must label it is essentially. So, for example you have a figure like this, so we will look at a variation which talks of simple figures though Waltz algorithm actually applies to more complex figures the simple figure that we will talk about a trihedral which have 3 faces.

So, each vertices has 3 faces and faces of planer, so any object which is made up of planers, whether surface of planer and where every edge every vertices is made up of exactly 3 edges. So, all this vertices qualify, so these are 3 edges three edges and 3 faces coming to meet there, these vertices 1, 2 and 3. Then every vertices made up 3 faces, such object are called trihedral object and certain object, for example if I am able to draw something like this, if you can imagine this over this let me object like this. You would not be trihedral object even if equal to be object, so anyway this is not a, cana of object

we are dealing with and interested in these cana of object.

Now, it is a labeling problem which means you have to label every edge and labels are as follows, class labels for stands for convex edge. Remember, we are talking about an edge by convex, we mean that the matter or material is inside, in some sense I would label this edges plus because it is a convex edge. As when seen from outside, minus edge is a concave and I would label this and this edge, for example as minus because it made up of a concave, so corner of a room between the walls and roof.

For example, these are out concave edges essentially and various books used different some people use notion fold and blade. But, we will use the notion of arrow, either pointing this way or that way we will distinguish between them, but the general idea is matter on right side. What you mean by right side? That if we are following the direction of the arrow the matter is on the right side and the other side is blank arrows of whatever. So, if I went to look at this figure matter is this side, so I must label it this side and like this and so on. So, this Huffman close labeling task is to label a line drawing with these 4 kinds of labels, in a more general case there are other kinds of labels that we use.

For example, we have shadows, then if we have cracks in objects or if we have more than 4, 3 edges, 3 phases meeting at vertex. So, all kinds of edges in be there and they could be other kinds of labels, but for this very simple class of object which is trihedral objects which are made up of planer where each vertex made up of 3 faces, there are the 4 kind of objects we can use to label. So, object in the task is to find this labels essentially, now this space for this problem you can see is that each edge can be label in 4 ways essentially and we in vertices.

So, when we look at a line, the line, the drawing like this, we can distinguish between 4 kinds of verities, one which we call is the Y vertices which look like this, which is made up of 3 vertices which are looking something like this. One, we sometime call as a W vertex which look like this, one we call as a T vertex which look like this, so this is 3 edges, so this is the vertex, but the view point is like this. So, if go to seat from here, exactly it should look like a tea essentially and the 4 is an L vertex where you cannot see the third edge, which is coming to that.

So, for A, these are the only 4 kinds of vertex that we can see and we are assuming here that these are generic views and these vertex. You know it is not as if we have 2 objects and somehow by placement of 2 object it is looks like it is a straight line, we avoid such views essentially where the little bit of align change of camera, if you want to call it will change the view. So, we assume these are cana of generic views essentially and the task is to label drawing like this, so if you go back to this original drawing you want to find this set of label for this.

Now, any vertex with 3 edges coming to it in principle can be label in 4 into 4 into 4, 64 ways, so, 64 plus 64 plus 64 plus 16. But, because these objects are trihedral, we know that they can be ladled in less than 4 ways, so let me take this example this W vertex, it can be label as. So, I will label it from left to right, it can label, this can be minus, this can be plus and this can be plus, as we can see here minus plus and plus that is one possibilities or it can be matter on. So, let me just label it, it can be like this as you can see here this is plus this is like this or it can be plus minus.

We have an example, we have one here this is plus this is minus and this is minus, now it turns out that for this W kind of vertex, these are the only 3 physically possible set of labels. So, from 64, we have broaden it down to 3, can we, this information is the question we ask in this labeling problem and I will leave this is an exercise. This is broaden down 6, this is broaden down 6 and this is also broaden down 6, there are only 6 base of labeling base T joints and so on. So, this T can be, for example part of a table, so you have a table and this is a leg or something like that.

You could see a T joint here in which case you can imagine this will be like this, this will be like this and this will be like this. This is one of the 6 base and there are 6 different ways you can label it T joint and it should be a interesting exercise for you to try and find this out. So, how can be exploit this information? The one simple piece, the one simple pack we should exploit is at one edge can be labeled only in one way at both the end. So, every edge has 2 ends it participates in every edge participates in 2 vertices, so it participate it must be label does a same in both the ways.

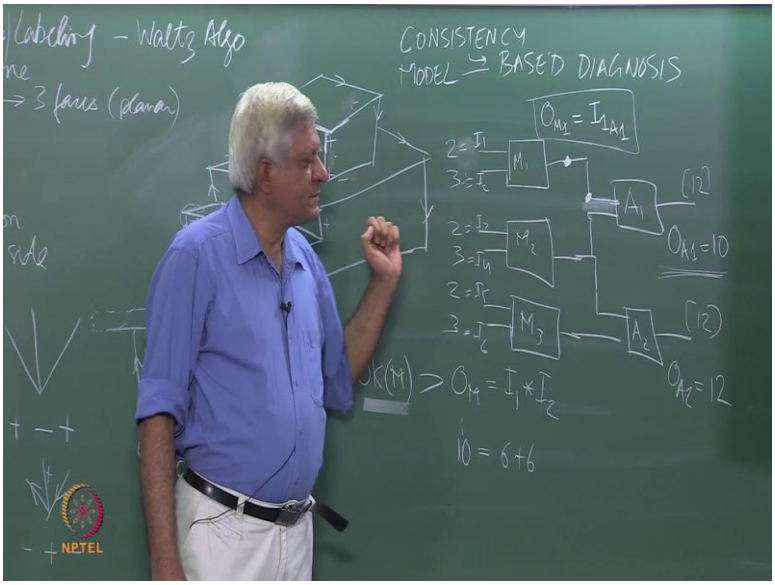
So, for example, once I have label this edges plus it means, it plus at this end it is also

plus at this end, obviously it is same edge essentially it is a simple trihedral object, we are talking about. Essentially, once I know that this particular edge is plus here, this part is plus I have constrains on this part, so I can look up a table for this out of the 6 possible relations which of them has a plus in the middle. What are the 2 other things on the side? So, I can prune the domain of that particular those 2 edges exactly like we prune the domains for arithmetic puzzles when we got some valuable prune values for the domain essentially.

So, what Waltz algorithm essentially does is that it does a, it first does a scan of this entire set of 9, drawing it makes an assumption that this is a solid objects which means that was edge is are the boundaries which means matter lies inside them. So, it can start up by labeling this then labeling this then labeling this then labeling this and so on and then it does propagation. Essentially, that once we have this outer label they will constrain the inner label, for example if we look at this place here it allow only a positive label to be given here because in the W joint we have only these three are possible.

So, we are, fix this variable we can propagate to plus here and this is plus you will see that this plus, plus, plus is the one of the queue combination that is allowed essentially. So, this is the idea of propagation, essentially once you know a value at some place you propagated to the next one and prune the domain for the next one. Exactly, like what we talked about here revise the domain of the particular variable and these up to waltz algorithm does essentially. So, let me give another example where whole same satisfaction has been use as a mechanism for doing something.

(Refer Slide Time: 38:34)



This example comes from, what we call is consistency base diagnosis, so let me use one standard example which we use in this thing, let say we have a small device made up of 3 multiplier. This is M 1, this is M 2, this is M 3, it takes to inputs, let us say the inputs are fix, so let say I 1 I 2, I 3 I 3 I 4, I 5 I 6, it produces, each produces one output. Let us say this output is to an adder, and this output is produced, spread to another adder and we have 2 values, here is a small device which does something from add an addition.

So, multiplication combination consistency base diagnosis is kind of also called as model base diagnosis. So, model base consistency base diagnosis got it says is that you construct a model of the system and a model that we build is a constrain model. How do we do that? We say that multiplier is defined as follows that if it is, then I will use this for implications sine output of M equal to input 1 into 2. So, I am defining a multiplier which works I, saying that if the multiplies is then this relation must old which is the constrain between through variables, 3 variables of that M is equal 2 I 1 into I 2, I N, so on.

For the others, we can define then we can talk about the connectivity, we can say that the output of M 1 is equal to input one of a, one in some notation we can say this. So, we are saying essentially that this output is connected to these input and that is the express by

this con this also constraint between 2 variables that this value must be the same as this value that is all we are saying. So, by we, so we describe these 3 multipliers plus 2 addressing constraint like this, this is the logic constraint is the logical statement, this implies this essentially we can converted in to a constraint which this being a statement some sort.

So, we describe the whole device by saying this, this part tells you what the connectivity is, this part tells you what the behavior of individual component is, and between these 2, we have describe the whole device essentially. So, now what supposing we get this input 2 input, 3, here what do we expect? We can predict that what this device will produced is 2 into 3 is 6, here 2 into 3 is 6, here 2 into 3 is 6, here and then 6 plus 6, it should give us 12 and should give a 12, that is the expected output essentially.

But, what happens if O A 2 is equal to 12, as expected this is the real value real constrain or observation some people would call it. But, anyway in the C S P notation, it is a constrain and O A 1 equal to 10, now supposing I give you this problem I H, what is it? It is a constrain satisfaction problem where these is the variable, in variable it is a either true or it is falls and this whole statement is also in variable it is either true or it is falls which can be from the mathematics. So, let us not get in to the details here, but there is basically a set of constrains which as description of the aiders predicts, what the output should be?

But, the system given to us is at the inputs are these 2, 3, 2, 3, 2, 3, this is a multiplier, this is a multiplier, this is a multiplier, this is an adder, this is an adder, the outputs that you seeing is output here is 10, the output here is 12. So, what is happening? So, this approach to diagnosis the task have diagnosis here is to identify of faulty component and, so we assume that connections are never faulty. So, this is a simplified, we have looking at things that one of these either the multiplier or the adder has become faulty. So, how do we do this?

So, that is has a description level, we say that this is a C S P, each of these, the 3 inputs are variables, this is constrain between those variables 3 inputs and a this multiplier. So, M 1, I 1, I 2 and O 1, 4 variables, this is a constrain between 4 variable, so if M 1, if the

multiplier is the then the output of multiplier should be equal to the product of the 2 inputs for the adder, then the multiplier of the multiplier. So, I have these constrains I have the constrains that I 1 equal to 2, I 2 equal to and so on and I simply say give me a solution for this C S P.

What are the variables which are missing the variables, which are missing are that? So, I know, I know the input variables, I know the output variables, I do not know the intermediate variables and I do not know whether all the devices are working or not. So, without going into to the details of how this C S P is search and it is quiet a domain in itself and there is a whole community which does model base diagnosis, the idea is to find the solution to this C S P.

What will be the solution look like? The solution will basically say some of these things of fault essentially for the C S P, P to be satisfiable some device. Some component must be faulty, some component, some statement which says this component is must be faults and that will be discovered in the solution which the solution remember must contain this at this output is 10. So, which means I making a statement like 10 is equal to 6 plus 6, that is what the adder is saying, so obliviously this is the faults statement.

Now, whether the fault is with the adder or whether the fault which is the 2 inputs, because nobody, Y is told me that the input is 6, the input is hidden from me, it could be something else essentially. So, is this adder faulty? Is this multiplier for faulty? May be these two multiplier is a faulty, which possible that these two a faulty, in such a way or not that these two are faulty in such a way that this is doing something and this is undoing that and producing 10. What this is doing? Wrong is being reflected here which possible, so diagnosis of course there is no clear cut answer to this question, has to why are we seeing this 10 here.

But, the algorithms are given towards finding minimal diagnosis which kind of is a laser that you must heard out the simplest solution are the best which says that all though diagnosis in which only component as faulty is other preferred diagnosis. Then we can see it must be A 1 or M 1 and the algorithm actually find that essentially that either A 1 is not or M 1 is not and that is why we are seeing 10 there. So, what I wanted to show

here was that this is the another example where the problem can be as a C S P, N solved has a C S P essentially, and this is problem of diagnosis essentially.

So, I will stop with constrain satisfaction here because we do not have too much time left in the course and in the remaining part of course I want to look at knowledge representations which is really a core of A I. We should not finish and course without acknowledge in presentation. So, next will lecture will focus on logic as a language for representation and listening essentially.

Thank you.