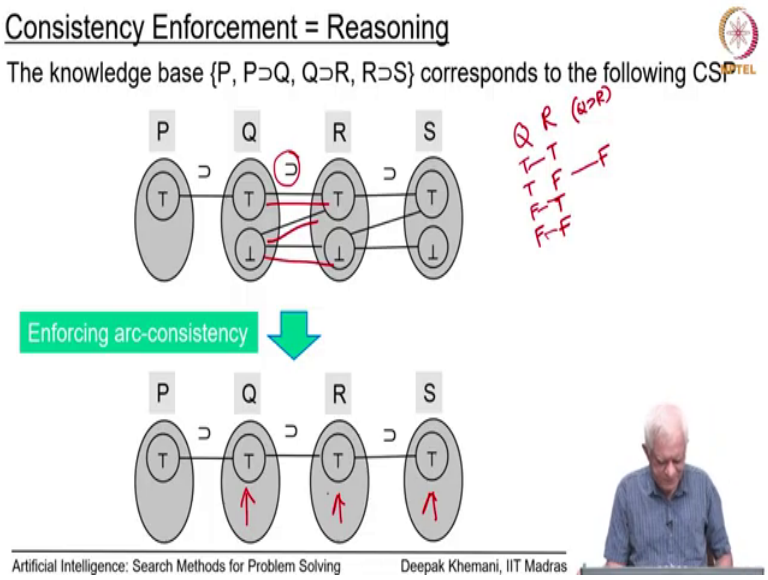
**Artificial Intelligence: Search Methods for Problem Solving**
**Prof. Deepak Khemani**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Madras**

**Chapter – 09**
**A First Course in Artificial Intelligence**
**Lecture – 94**
**Constraint Processing**
**Propagation = Reasoning**

(Refer Slide Time: 00:15)




So, I kept saying that constraint satisfaction allows us the possibility of combining search with reasoning essentially, and reasoning we have typically associated with logic essentially. So, let us try to see what is happening here when we treat Boolean variables and the relations between them which are logical relations as the constraint network.

So, here is a very simple problem. The knowledge base given to you is that P is true; P implies Q is true Q implies R is true and R implies S is true. Now, we have already seen that the implication relation is captured by the truth table. So, as seen here there are 3 edges going from Q to R and you will remember that there are 3 rows in the truth table of implication.

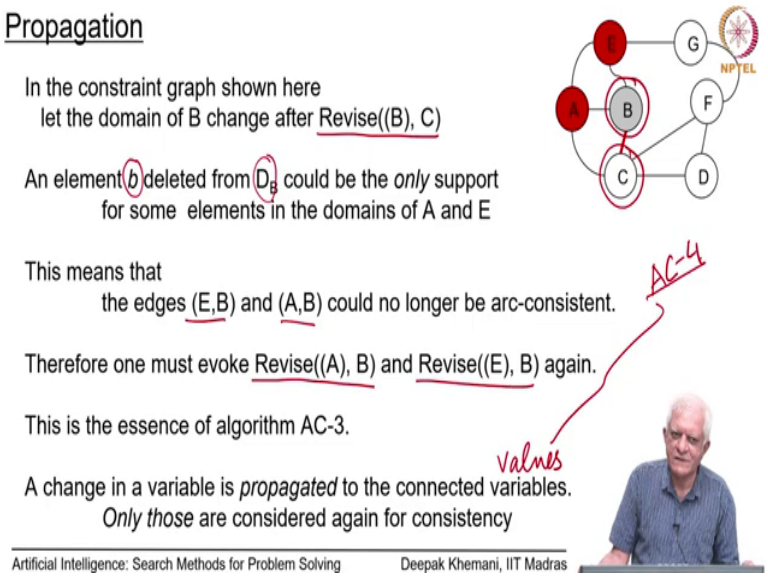
So, if you write Q, so this is true, true, false, false and if you like R true, false, true, false. So, I have written false instead of bottom, but I hope you are familiar with the terminology by now. So, the only case where Q implies R is false is this one and in the other 3 cases its true and those other 3 cases correspond to the constraint graph here. So, there is an edge between T and T there is an edge between F and T, there is an edge between F and F.

So, there is an edge between F and T here, F and F and between T and T. So, you can see that a logical relation can be seen as a constraint graph and this problem that we have written can be seen as a CSP essentially.

And now if you do consistency enforcement in this case R consistency, what we get is an R consistent network. And you can see that in effect we have drawn the conclusions that are derivable from the knowledge base that if P is true and because P implies Q is true Q must be true.

So, you can see that here. And if Q is true then and Q implies R is true then T must be true, likewise S must be true. And these are the only values in the domain, so in some sense we have done logical reasoning, we have made inferences. So, consistency enforcement is equivalent to doing reasoning essentially, ok. So, we said that AC-1 is correct, but it is not efficient. So, let us try and make it efficient now essentially.

(Refer Slide Time: 03:20)




We will look at this idea of propagation. And look at this map coloring diagram on the right. And what we have is that let us say we have done this operation of revised B with respect to C, and let us say that the domain of B has changed because you are pruning going to be domain pruning the domain of B. And let us say that some element b has been deleted from D of B.

And if that is the only support for some elements in A and E as a consequence of deleting this be from the domain of B those edges will become will not be arc consistent anymore. So, what do we need to do? That edges E and B, and E, B and A, B could no longer be arc consistent, so better go and make them arc consistent again.

So, what we can do is we can make calls to revise A, B and revise E, B again. Instead of what we did in AC-1 is that if any domain changes do all combinations of calls, for all the edges, all over again go through the entire cycle again.

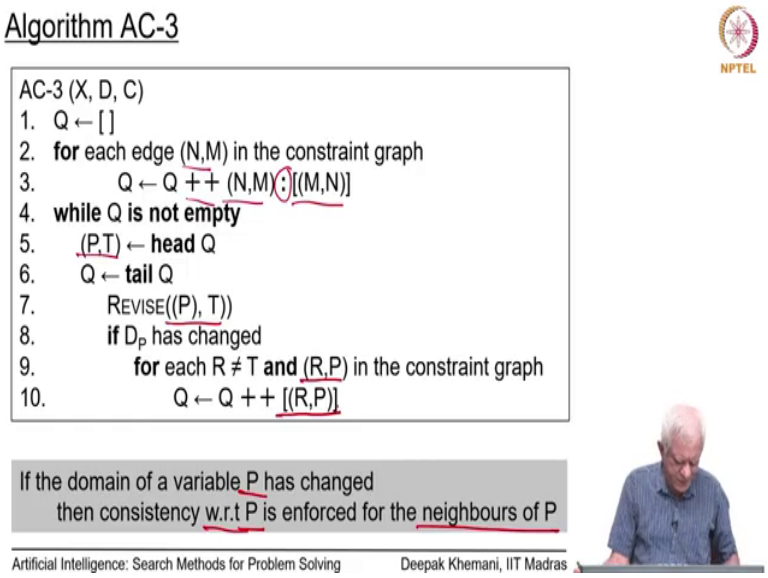
Now, we can be more selective and we can say that under certain circumstances some revised calls need to be made and this figure basically illustrates those circumstances that if the domain of B has changed with respect to C, then anything which was connected to B those must be inspected again. In this case, that is A and E. And that is in essence that is the essence of the algorithm AC-3, it is one of the most popular algorithms in constraint satisfaction.

A change is propagated to the connected variables and only those are considered again for consistency. In our examples we did consistency with respect to with B with respect to C, and because of that because B changed we said that we must cross check A, B again and E, B again. So, it does only selective calls to revise.

There is another algorithm, which again we will not go into that which is called AC-4 which talks about values here, that you just you do not do the whole variable itself. So, you do not look at all values in A and in E to see if it is consistent. You know that we have deleted B from the domain of B.

Just look at those values in A and E which was supported by B and if that was the only support for those values, then you have to do something to make them arc consistent. So, that is the even more fine grained level of looking at the problem and that is called an algorithm AC-4, which we will not go into that because it requires a bit of an initial setup that we do not have time to look at this.

(Refer Slide Time: 06:44)




So, we will be satisfied with AC-3. So, that is the algorithm that we have. If the domain of a variable P has changed, then consistency with respect to P is enforced for the neighbors of P that is the basic idea of AC-3 and we do that by maintaining a Q of edges that we want to look at and not just the Q of edges, but Q of one direction in each edge.

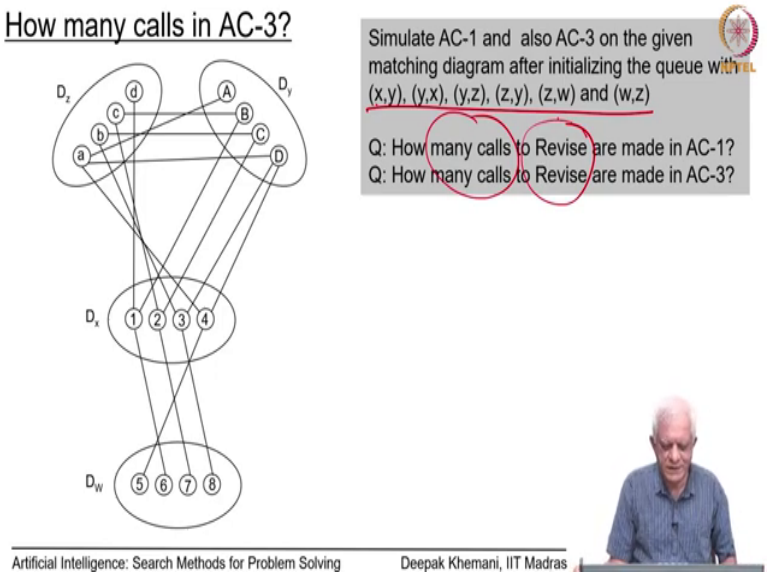
Initially, we have that empty Q as we can see here and then for each edge N, M in the constraint graph we add them to the Q essentially. So, this notation is the list notation that we have been looking at. I hope that is fine. This is the const operator and this is the append operator. So, this, so far it is like AC-1 that do all the edges in both direction once, but after that be selective and that is what happens here.

While this Q is not empty, so while it is in the first round of course, all the edges will be done once. We remove the first element from the head of the Q. We call revise, so we have

removed P T. Remember that P T and T P both will be in the Q we have looked at P T now, so we will call revised P T.

And the domain of P may change, and if the domain of P has changed then for each variable R which is not T such that R, P is an edge in the constraint graph you add this R, P called. So, this is like saying I want to do revise R with P again. Why? Because, P has changed and R was connected to P. So, let me just check that R is consistent, R consistent with respect to P.

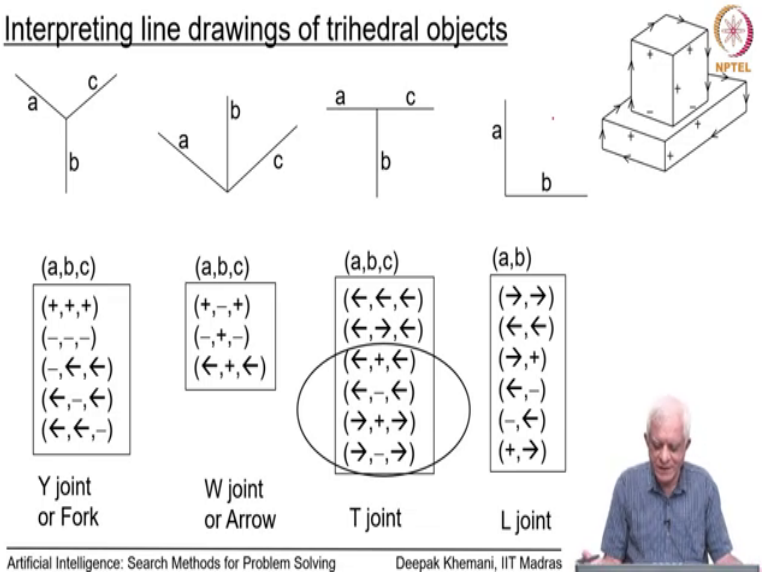
(Refer Slide Time: 08:37)




So, that is the algorithm AC-3. It is a very popular algorithm. I am used a lot in this thing. I will give you a small example to work on. And there are 4 variables here, and the connections are as shown. And I have given you an order in which you will start the revise operators.

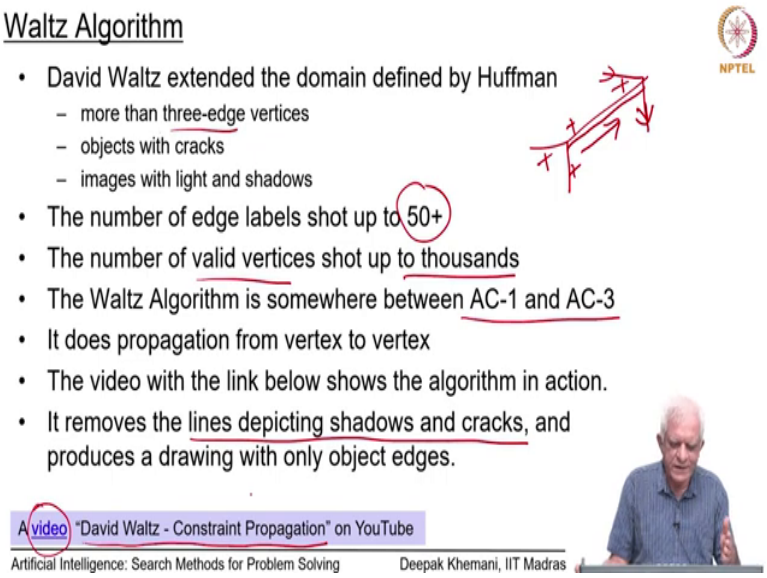
First, x with y, then y with x, then y with z and so on some order I have given you. What you must do is try both AC-1 and AC-3, and count how many calls you make to revise essentially in this. So, this is a small exercise. And hopefully you will see that AC-3 is more efficient then AC-1.

(Refer Slide Time: 09:24)




Now, remember this line drawings we had talked. We had this Huffman had given us talked about trihedral objects, edge objects which have 3 phases meeting at a vertex. And he had said that there are 4 kinds of vertices in such drawings, a vertex is somewhere where two lines meet, or two edges meet. And they were 18 kinds of labels that you could use for vertices.

(Refer Slide Time: 09:52)




This work was further extended by David Waltz. And what he said, then let us consider edges vertices which have more than 3 edges. So, they do not have to be trihedral, you can have 4 edges meeting or 5 edges meeting, it does not matter.

He said let us also consider cracks. So, a crack will be seen as a line in the line drawing. And he said also let us consider light and shadows because they will also be seen as line drawing. So, we are assuming that some kind of image processing activity has happened, edge extraction has happened, and what you are given is the line drawing and the task is to label the edges of those lines essentially, which means label the vertices also.

But now the number of edge label shot up to 50 plus, in Huffman's problem they were 4, you know 2 arrows, N plus, and minus, but now they have 50, more than 50. And then number of

valid vertices short into 1000s essentially. So, it is a huge problem that David Waltz worked on, it was part of his PhD thesis at MIT.

And he wrote an algorithm which is now called as the Waltz algorithm. And it turns out that this Waltz algorithm is somewhere between AC-1 and C 3. So, you might have been wondering why AC-3 came after AC-1 and where is AC-2. So, some people say that you know Waltz algorithm was kind of AC-2.

But it is so close to AC-3 that people did not want to give it a different name it does propagation from vertex to vertex and we have seen that right, that if you have a vertex like this and then there is another vertex like this you can imagine that this is a block that we are talking about.

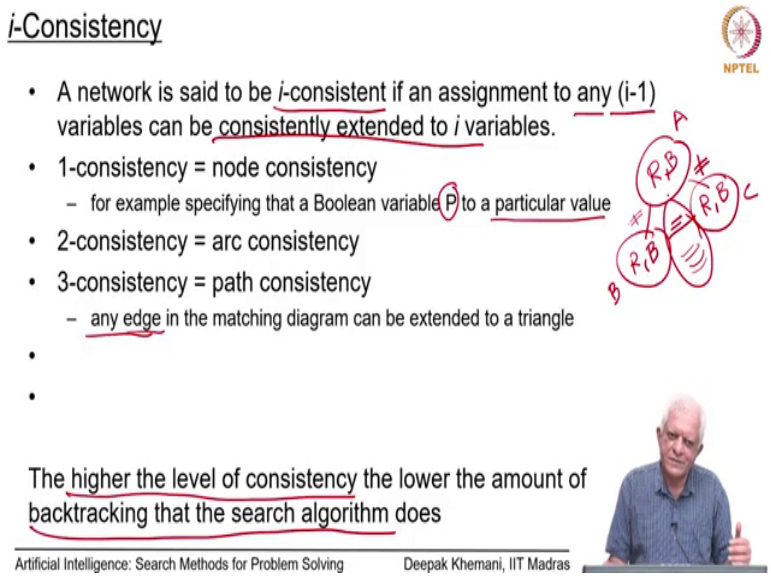
And if you have figured out that this has to be plus, then this particular edge cannot change its nature at the other end, so this must be plus. And once that is plus you are forced to choose something like this and this. It could be minus also as if you remember the set, but typically it would be like that essentially.

So, propagation is happening from here to here and likewise essentially, which is what we are doing in AC-3 like algorithm. And this is what Waltz algorithm did for line drawings.

We will not go into the details, but I will give you a pointer to a video which is available on the net which kind of shows an animation of Waltz algorithm. There is a clickable link here, but otherwise you can go to YouTube and search for this Waltz. This thing David Waltz constraint propagation in this video.

And what this video does it looks like a jumble of lines to start with it removes the lines depicting shadows and cracks and produces a drawing with only objects edges. So, the original task which Huffman had posed as to, is there some real physical object in the line drawing, he solved, but he solved it not just for trihedral objects, but for objects which were more complex as well.

(Refer Slide Time: 12:57)




So, let us end this section with a discussion on different levels of consistency. While we were talking we mentioned path consistency a little while ago, and we can formalize that notion of higher levels of consistency by saying that a network is set to be i-consistent which is here if an assignment to any i minus 2 variables, any i minus 1 variables can be consistently extended to the i-th variable essentially. So, you choose any i minus 1 variable and you will get a value for the i-th variable.

The simplest kind of consistency is 1-consistency. We have not really considered that, but if you remember the example of logical reasoning you could say that we have this 4, 4 variables P, Q, R, and S.

And the domain of the 4 variables are two values, true or false. And then on top of it you impose a constraint that P is true. So, if you say that P is a particular value, then that is like

enforcing node consistency and saying that you can only keep those values in P which are consistent with the unary relation over P essentially, and the unary relation may say that P must be true or P must be false. And in our example that we saw we had say that P is true essentially. So, that is 1-consistency.

2-consistency is called arc consistency. We have already seen the meaning of 2-consistency is that you take any variable, choose a value for that variable, and take any connected variable and you will have a value available in the other variable essentially.

We mention in passing 3-consistency, which is also called path consistency. In the matching diagram, you take any edge and you should be able to extend it to a triangle. If that is the case; that means, we have found a value for the third variable which is consistent with the first two variables essentially.

And then there are higher order of consistency and it turns out that it depends upon the topology of the graph. As I said earlier if the network is a tree network then arc consistency will be enough. But if network has more connections, so it is a more complex graph then you need higher order of consistency.

And, when we study this relation between consistency and how much how much search it does, it turns out that the higher the level of consistency the lower the amount of back tracking the search algorithm does. That there is a certain amount of consistency that if you do then the search will be back track free.

What is the advantage of backtrack free? That you will just end up taking those variables in one by one and you will find the solution, you will never have to backtrack essentially. So, you have done enough pruning and so that this property happens. It is not just pruning of this thing, it is also pruning of universal relations.

We will not get into that. We had mentioned sometime that if there is no edge between two variables, it is equivalent to having a universal relation between those two variables. So, those get pruned sometimes essentially.

So, for example, if you have a simple map coloring problem here, again 3 regions, but only two of them are connected and the relation as before is not equal. Then, if you impose path consistency on this network, it means that there were 4 edges between these two variables. So, let us call them variables B and C. So, there were 4 edges between B and C to start with because we say there is a universal relation.

And, we will once you make this path consistent only two of them will remain. And the two will be such that this relation will become equal to. So, if you choose R red for A, you will have to choose D for both B and C, if we choose B for A you will have to choose red for both B and C. Alternatively, if you start with B and if you choose red then you will be forced to choose red for C also and so on.

So, this is something we added to that essentially. So, we prune the universal relation to give us a new relation, but we will not get into that. What we will do is to look at this idea of enforcing this consistency as we are doing search.

So, so far what we said was you can make the network consistent to some level i essentially which spoke only about arc consistency. And then you do search and you will find that there is a reduced back tracking. But can we do this kind of reasoning?

Remember that consistency enforcement is a kind of reasoning on the fly, and that is what we will do next and that will be the last topic that we will study in constraint satisfaction problems. And we will do that after a short break in the next video.