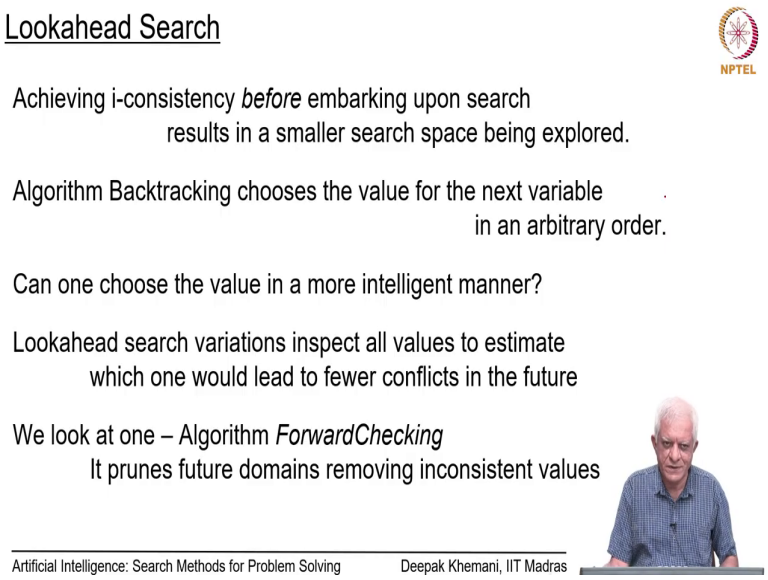
Artificial Intelligence: Search Methods for Problem Solving
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Chapter – 09
A First Course in Artificial Intelligence
Lecture – 95
Constraint Processing Lookahead Search

(Refer Slide Time: 00:14)



So, welcome back. This is the lap last lap of this week and also the last lap of this courses essentially. So, the last thing we want to look at is this very interesting idea of doing search, but while doing search also doing reasoning and by reasoning now we mean propagation and we said that we will talk about lookahead search methods. So, let us do that now.

So far, we said that achieving i-consistency before embarking upon search results in a smaller search space being explored that is fine, but what we did not say is that the amount of work you end up doing in enforcing this consistency can be quite large essentially and moreover because you are processing the network in a particular order, you do not have to worry about consistency in both directions.

Because once you have chosen a value for x 1, you should be only worried about future variables whether there is a consistent value of x 2, x 3, x 4 which whatever the relations are and whatever the level of consistency that you are doing.

You should not be worried about whether for every value of x 2, you will choose a value of x 1. I am talking about arc consistency here. So, you have only worry about one direction thing.

So, you do not want to do extra work which you do if you pre-process the network to do that, but even there we have kind of skipped over sections where we do what is called as directional consistency, but essentially, you are doing all this in advance so, it could end up doing more work essentially.

Now, the algorithm backtracking that we discussed chooses a value for the next variable in some arbitrary order. We said that this is the order in which you will select values. Though there are algorithms which we are not talked about called dynamic value ordering, where you try to choose those values which are likely to be more consistent, but what we are going to do now is a little bit different.

We will not choose the value in a particular we will not change the order in which we choose the value but when considering a value, we will see whether it is going to cause a conflict later on essentially which is a different thing which is where these algorithms are called lookahead search that I am considering this value, is it going to cause the problem later on?

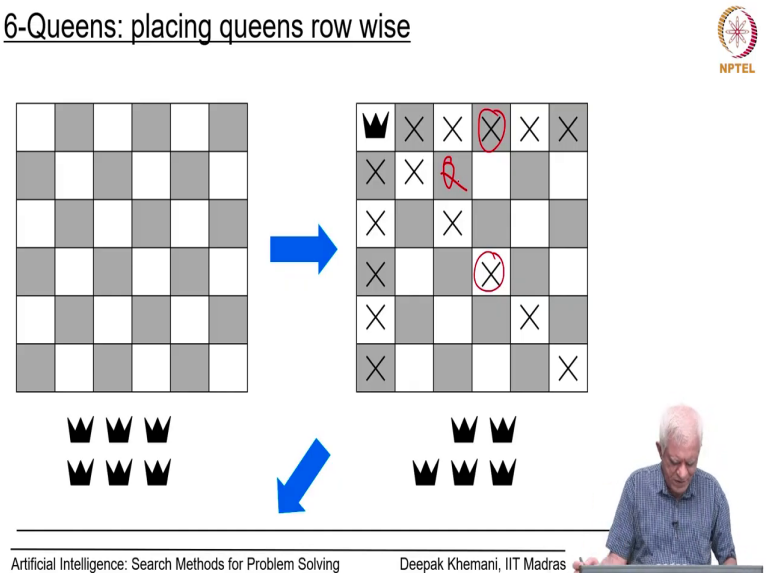
Dynamic value ordering which I just mentioned which we will not discuss says that look at all the values and choose one which is best likely to succeed so, that is a different problem altogether or different approach altogether.

So, can we choose the value in a more intelligent manner and by this in lookahead search we mean we are considering the value and whether we accept it or not can we do that in a more intelligent fashion.

What backtracking did was that for that value, you checked whether it was consistent with the past variables or the earlier variables. What for lookahead search does is it also looks at future variables and sees whether it is a good value for that. So, either you select it or you do not and that is how it works.

So, lookahead search there are variations to this. Inspect all values to estimate which one would lead to fewer conflicts in the future essentially. We look at one such algorithm which is called forward checking. It prunes future w domains removing inconsistent values as it goes along.

(Refer Slide Time: 03:58)



So, we will illustrate this idea first with this n queens problem. If you are working with people in pencil, this is what you might end up doing if you are solving it let me see in an intelligent fashion. So, the way we will do this is we will keep placing one queen at a time and after we place every queen, we will do a certain amount of reasoning which is as follows.

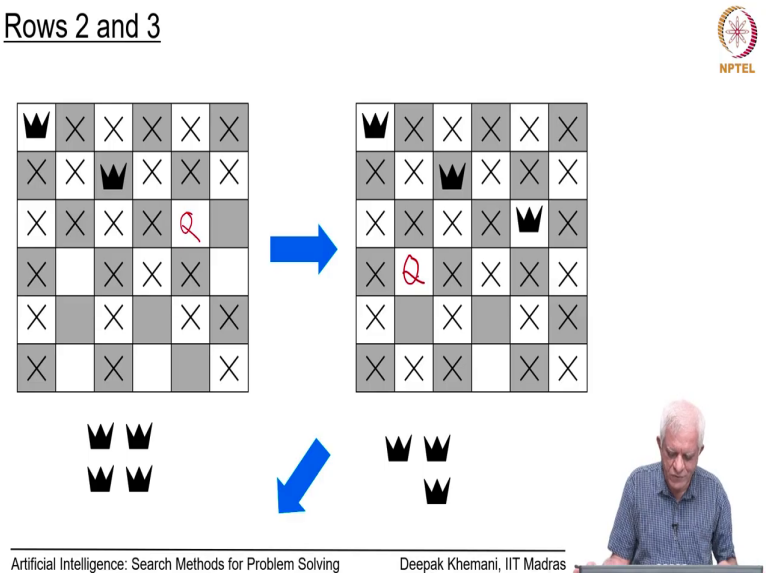
So, we will place the queens row wise so, first in the first row, then in the second row and so on. So, after you place the queen in the first possible position which is the corner most square as shown here, you can immediately say that the chessboard squares which have a cross in them you can never place a queen on that essentially so, but these are.

So, I have not restricted myself to what representation of the constraint satisfaction problem we are doing. We had looked at two representations; this is in some sense a generalization of

rule. If you have placed a queen as shown here, you cannot place another queen on all the squares which are marked X essentially.

So, the second queen can only be placed here and that is what the algorithm is doing it is still backtracking like algorithm, it takes a next value and tries that essentially.

(Refer Slide Time: 05:37)

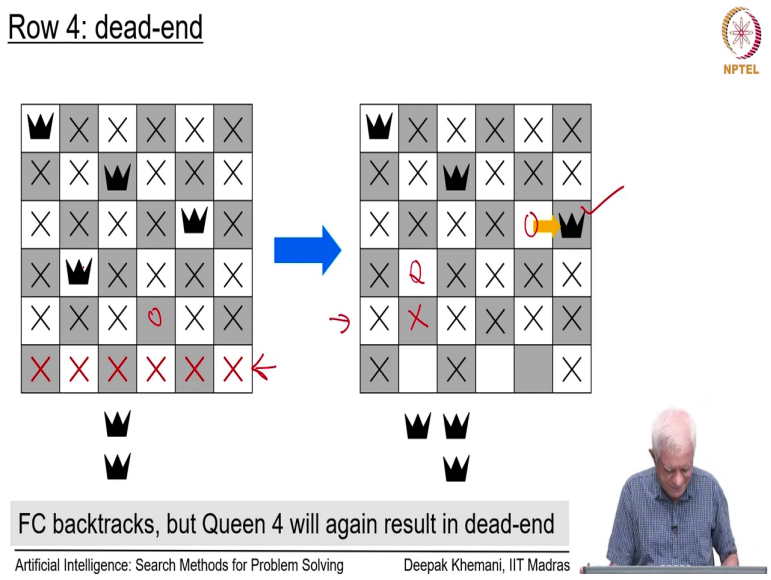


When it does that, you can see that the number of crosses on the chess board has increased quite a bit and the next value that you can place for the queen is here and that is what forward checking does.

It places a queen in there so, there are only three queens left and three queens on the board and you can see that the chessboard is the number of habitable squares are decreasing quite

rapidly, but nevertheless we still have a hope we can place the fourth queen in the fourth row here.

(Refer Slide Time: 06:15)



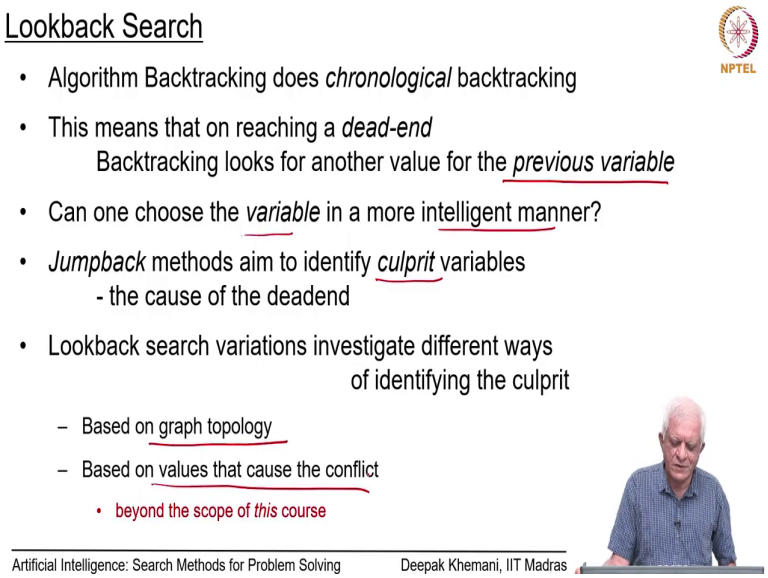
And when we do that, we find that in the last row, we cannot place a queen. So, even though we have a possibility of placing a queen on the fifth row here. It does not make sense because forward checking has seen that already there is no place for the six queen so, it will backtrack from their itself and that is a basic idea of the forward checking algorithm. It will not; it will not place the fifth queen at all.

Backtracking would have place the fifth queen and at when it reaches the six queen, it would have said that I cannot find a value, it is a dead end. So, backtrack from there. Forward checking backtracks from this place itself essentially.

So, it will backtrack, it will try different value for the third queen. So, instead of placing it here, it will now place it here, but as you can see, if you were to place the fourth queen here, then this square would not be available so, this row would not have O position essentially.

Anyway, so, that is the idea of forward checking. You lookahead, prune future domains and if any future domain becomes empty at any point, you backtrack from their place itself essentially. So, like we did here, we backtrack from after placing the forth queen, we realize we cannot places the fifth queen so, we backtrack to the third one essentially.

(Refer Slide Time: 08:00)



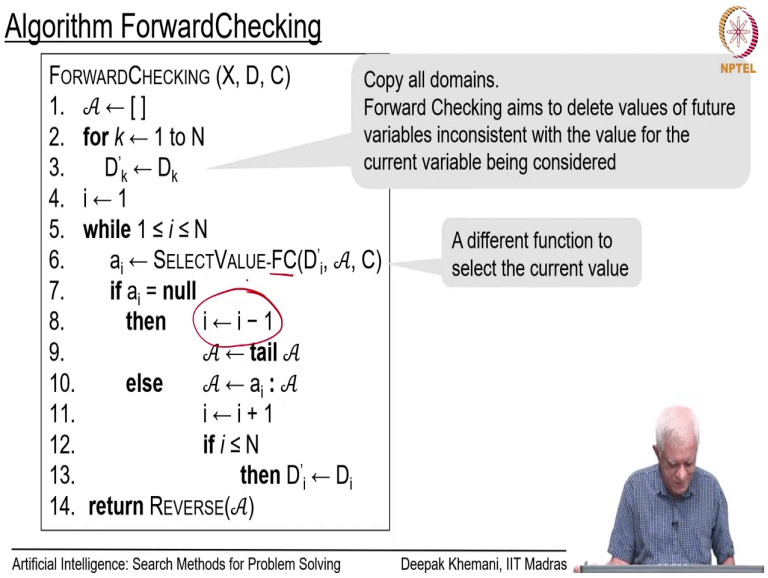
I had mention lookback search. So, let us talk a little bit about that. Backtracking does chronological backtracking. This means that when you reach a dead-end, you go back to the previous variable to look for a new value.

Can one choose a variable to which you want to give a new values in some intelligent manner and there are these jump-back methods that aim to identify what is called as a culprit variable. As so, the variable which led to the inconsistency or which led to the dead-end and you directly jump back to that variable essentially.

So, there are algorithms for example, there is an algorithm called conflict directed back jumping. Its figures out has to which variable I should go in change. Instead of plodding through in a chronological passion, if you are looking at variable i, go back to i minus 1, try all values to i minus 1, then go back to i minus 2, try all values and so on intelligent backtracking would jump back earlier essentially.

And there are two kinds of methods one which are based on graph topology because the graph tells you which variables are related to which variables, but also there are methods which are looking at values as to what values have we used up and so on, but we will not have time to go into this course.

(Refer Slide Time: 09:38)



Let us look at our algorithm on lookahead and the algorithm that we are looking at is called forward checking. It works as follows: it is very similar it is a with this is a variation of the backtracking algorithm.

So, it starts off in a very similar fashion that it gives an empty assignment, but instead of making a copy for the first variable, we make a copy for all the variables because you are going to be looking at future variables and you know we may need do a careful amount of bookkeeping to see that we do not miss out on actual solutions.

So, we start off by making copies of all the variables and then, we proceed almost like the backtracking algorithm that we look at the variables one by one and we call a different select

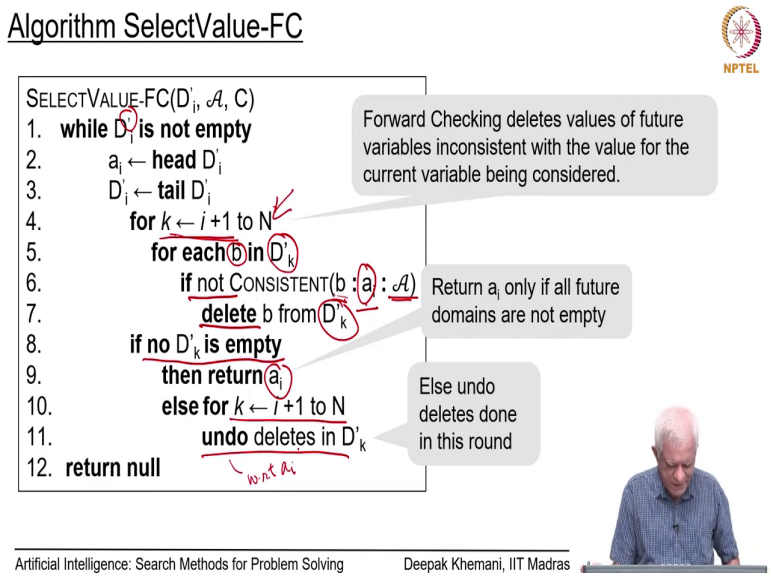
value function which is designed for forward checking and the idea is going to be that when we and we will see this in a moment is that when we are considering a value.

We will prune future domains and if any future domain becomes empty, we will not selected values essentially. So, this is called select value forward checking. As opposed to select value which was there in backtracking which simply looked at the past variable and said if this value is consistent with the past variables, then.

So, after we have got a value, the rest of the algorithm is like backtracking itself. In fact, it is a same piece of code that you can reuse. If the value is empty, then backtrack and if the value is not empty, then augment the assignment and go on to the next variable essentially.

So, what I said in a short while ago is that if you were doing intelligent backtracking instead of saying i is equal to i minus 1, you would use do something more interesting with which we will not look at here.

(Refer Slide Time: 11:40)



So, let us look at the select value forward checking algorithm here and it works like this. It is a little bit more involved than the select value function we saw earlier. Starts off in a same fashion that while the domain, the copy of the domain remember that this prime means that you are working with copies of domains.

So, while the domain is not empty, look at all the future domains, all the values of k which go from i plus 1 up to N essentially and we will go there and delete values which are not consistent.

So, for the future variable which is k look at the domain of that variable, look at a value b in that if you take that value and if you consider the value that you are considering for the select value a i and if you look at the past assignment so, you are looking at the past as well as

looking at this particular row in the future, the kth row. There are algorithms which look at all rows, we will just see a glimpse of that before we finish.

So, what is forward checking doing? It is looking at the kth it is looking at the it is looking for a value for the ith variable, then its looking at all future variables which are i plus 1 onwards and for each row in the future variables, it is considering whether the value b is consistent.

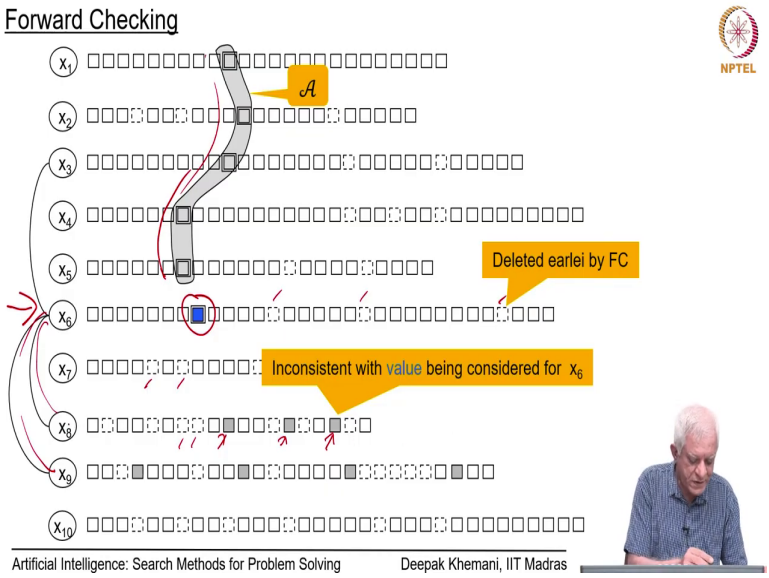
If I choose b for the k th row and if I choose a i for this row and I already have a past assignment, then will this whole thing be consistent or not. If it is not, then you delete it from the domain of the kth variable.

After you have done this, if the domain of some variable has become empty at some point and you can do this efficiently not in the manner that I have written, you can push this inside the loop a little bit, but what I am doing here is that after you are finished for all the n variables here, if no domain future domain is empty, then you can just go ahead and return this value a i.

If some domain is not empty which means that the first condition is not met, then you go and look at all those variables and undo the deletes that you did for those variables, reset the values to the ones that were before you choose the value a i so, this undo is only with respect to a i.

You were considering the value a i, then you ran into some future domain became empty, then look at all the future domains, reset them back to what they were before you considered a i because by the time you come to the case wherever, you might have deleted some on the way essentially. So, that is a algorithm for select value forward checking.

(Refer Slide Time: 14:54)



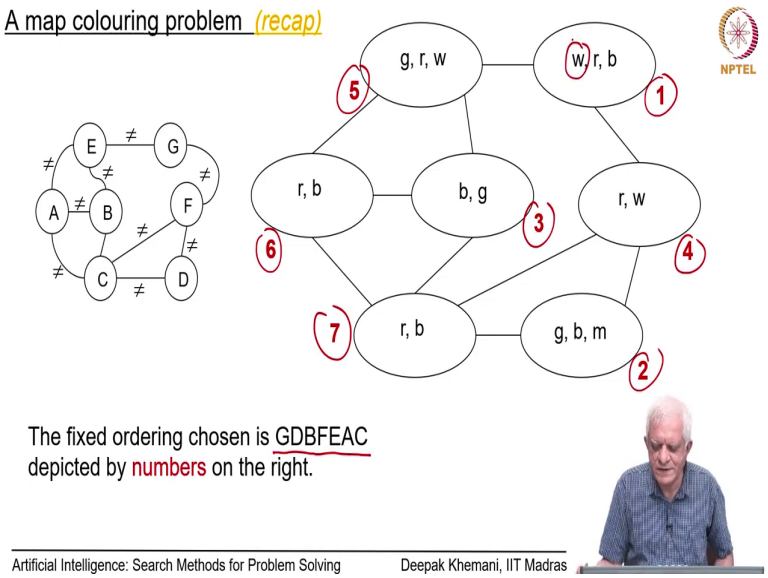
We can try and see this in a diagram probably helps us a little bit. What you see in this shaded part is the assignment that you have already done. You are looking at variable 6 in our case and this edges tell you it is the constraint graph which tells you what are they related to essentially.

You are considering this value for the 6 variable. There were already some values which were deleted by forward checking before even you looked at this value not only in this row, but in other rows also as you can see here. But when you are looking at this value, the value which is shaded in blue, then certain values which are shown in grey here, you might want to delete them.

So, notice that these are values which are related to X 6, X 8 and X 9 in this diagram so, you may want to delete those values before you progress further essentially. So, in that manner,

the select value algorithm will delete future values and those future values are the shaded ones shown in for X 8 and X 9 essentially.

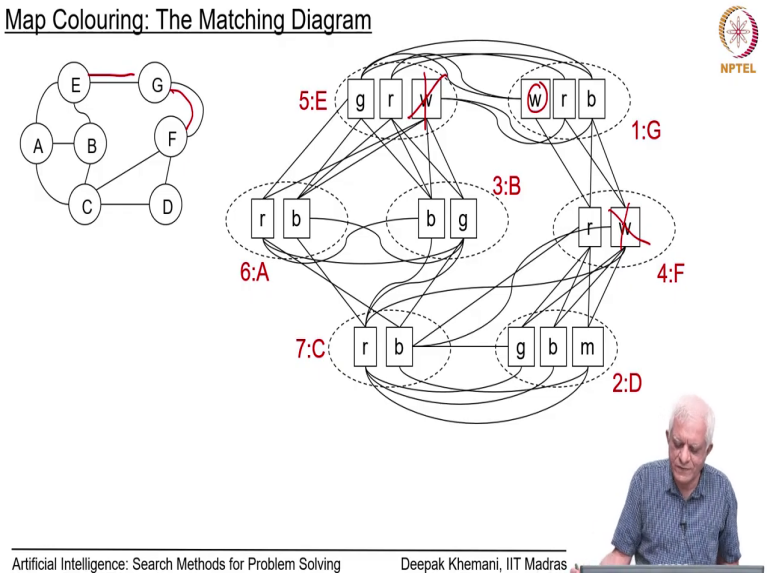
(Refer Slide Time: 16:04)



So, let us see this in action. We have this map colouring problem that we talked about earlier, we have a chosen order which is G first, D then B, F, E, A, C shown in red colors here, 1st, 2nd, 3rd, 4th, 5th, 6th, 7th.

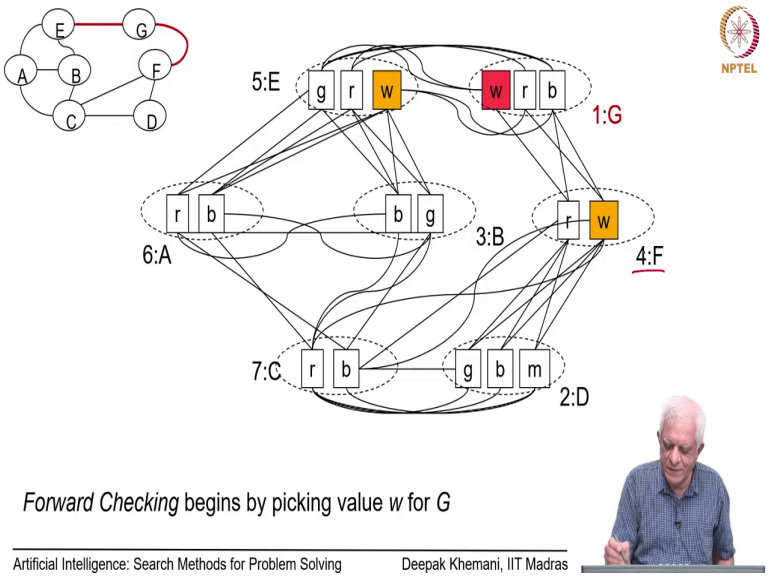
So, we are going to consider those variables in this order and we want to see how forward checking will work essentially. So, we will obviously, start with the 1st variable and we will choose the first value in the variable which is w and see what happens.

(Refer Slide Time: 16:44)



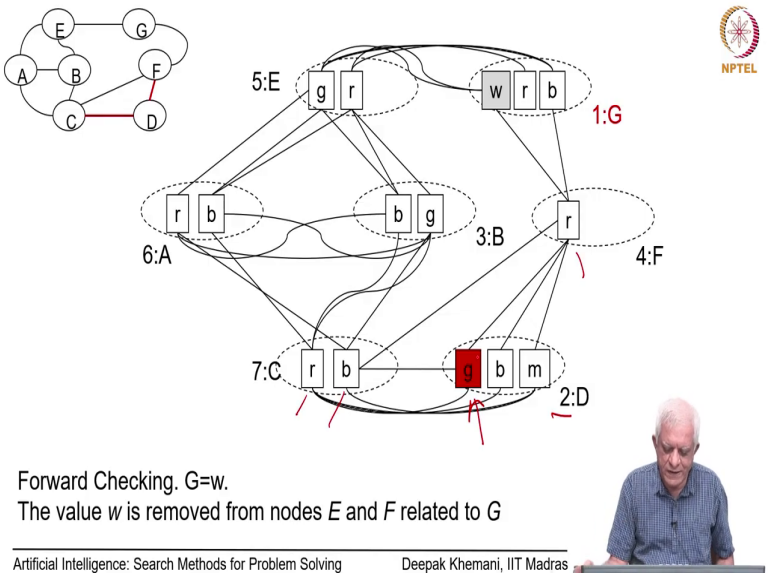
So, we will easier to work with the matching diagram because we can see as to what is connected and what is not connected and we are going to choose this w and as you can see, once we choose this w, forward checking will say remove this w from F and remove this w from E why? Because G is connected to F and G is connected to E and these values are not consistent with the value of w essentially. So, let us see, we will follow the progress for a few rounds.

(Refer Slide Time: 17:22)



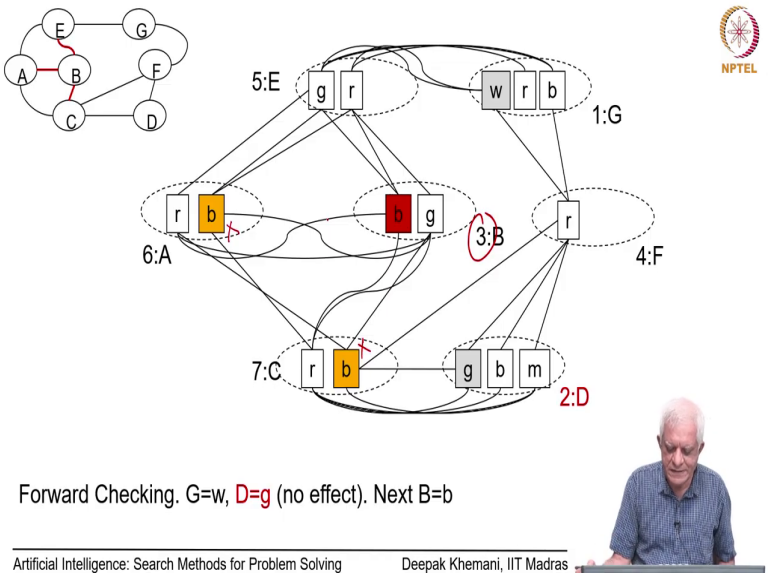
So, we pick the value [noise[w and as you can see, those two values w for F and E will be deleted.

(Refer Slide Time: 17:35)



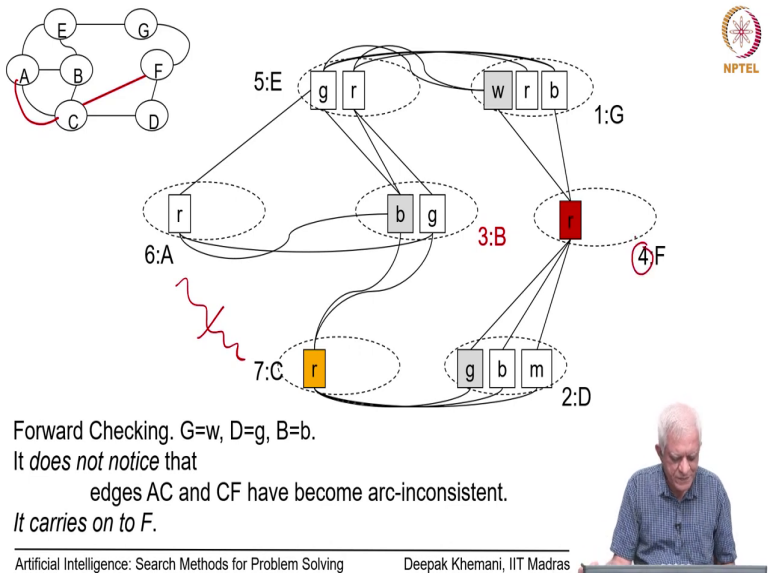
Then, we will pick for the 2nd variable a value the first value which is g, but it is not conflicting with anything so, we will move ahead. g, the variable D is connected to F and C and you can see that they have values if you choose g for D, you can choose r or b for C and you can choose r for F so, g is fine, we do not need to proven anything in the future.

(Refer Slide Time: 18:05)



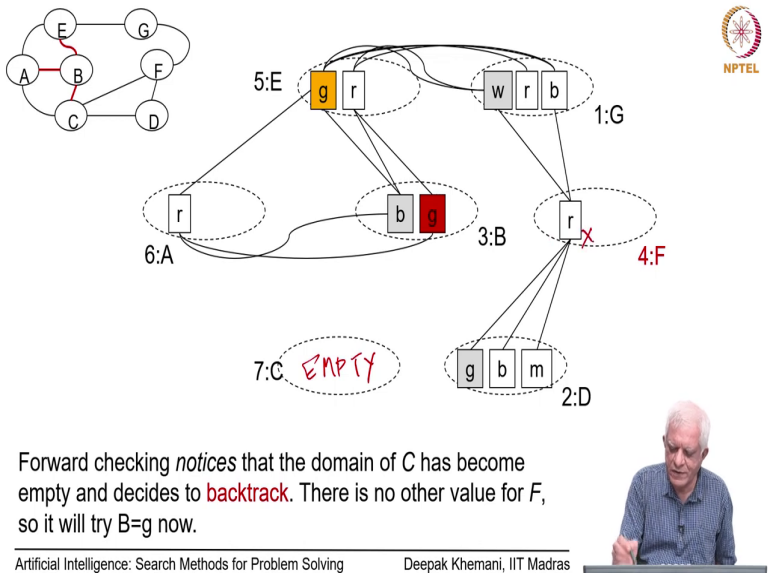
So, now, we move to the 3rd variable and we choose the first value there and as you can see, we will end up deleting these two values because they are not consistent with the value of b that we are selecting for this essentially.

(Refer Slide Time: 18:24)



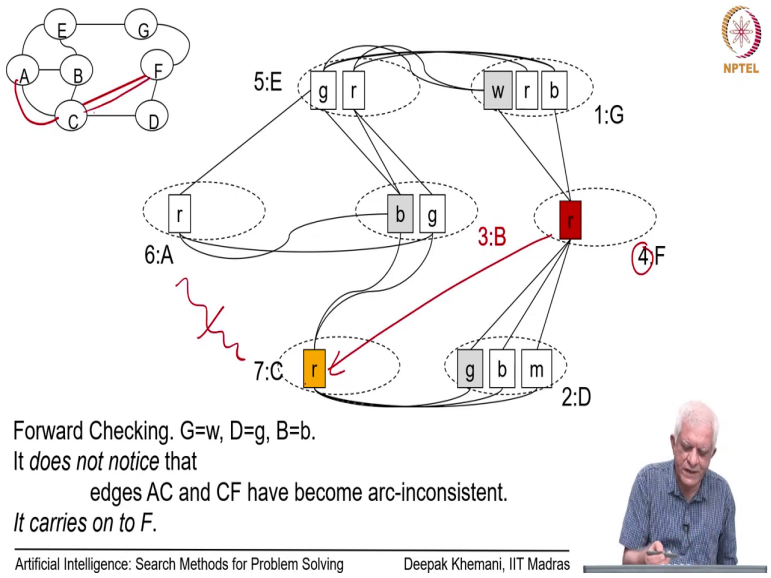
Then, we will move to the 4th variable, but at this point because forward checking is doing only very limited kind of lookahead, it does not notice that these two variables have become arc-inconsistent, they have only one value r and they are connected to each other in the constraint graph. So, you cannot give r to both of them, it does not see that and it goes ahead and gives the value of r to the 4th variable this thing so, it carries on to F essentially.

(Refer Slide Time: 19:01)



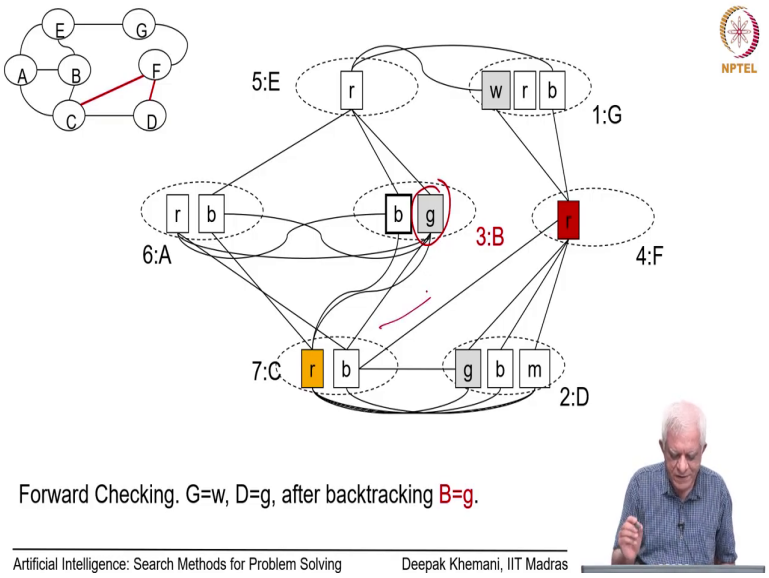
At this point, it sees that the domain of C has become empty.

(Refer Slide Time: 19:15)



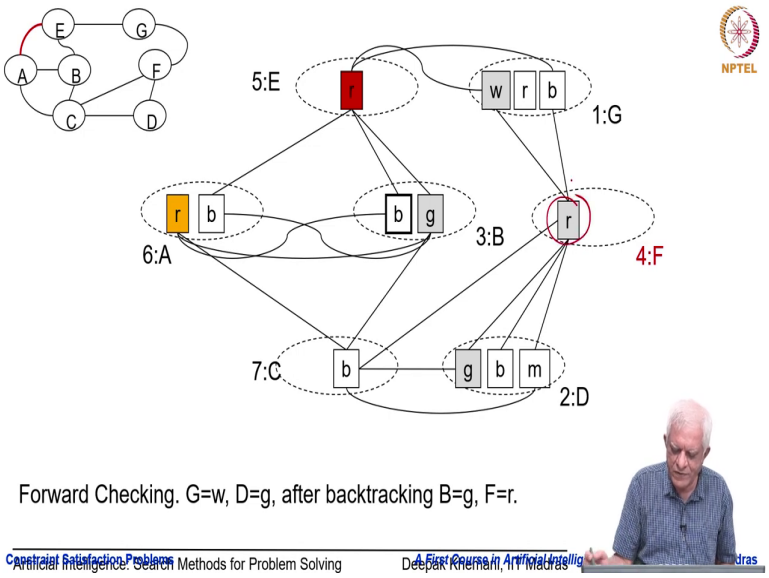
Why did that happen? That happened because of this connection that we see here. If we are going to choose r for F, then because F is connected to C, we cannot keep r in the domain of C even though C is the 7th variable, it deletes that and it notices at this point that this domain has become empty and it backtracks, it says that we cannot give this value of r to F.

(Refer Slide Time: 19:47)



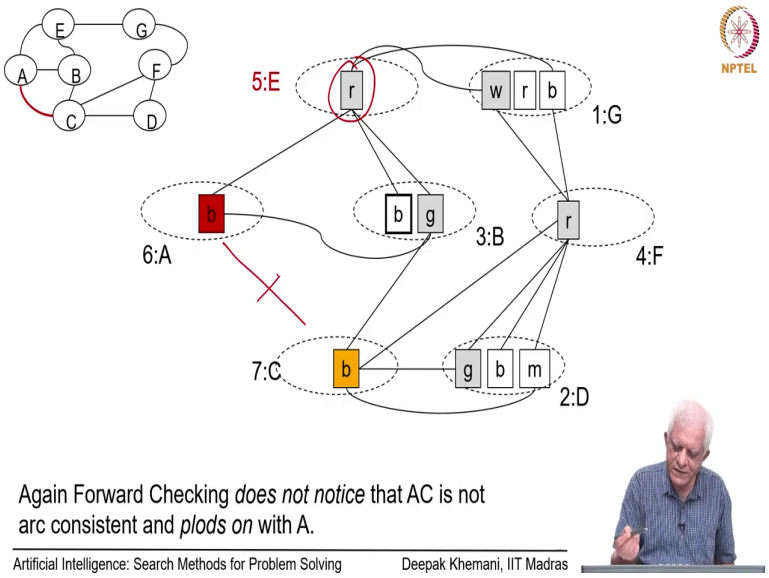
So, it goes back to the 3rd variable. So, its backtracked from the 4th variable and now, it is gone back to the 3rd variable and it says, I am going to look at g for B which is a 3rd variable and you can observe that this r, r inconsistency is still there.

(Refer Slide Time: 20:14)



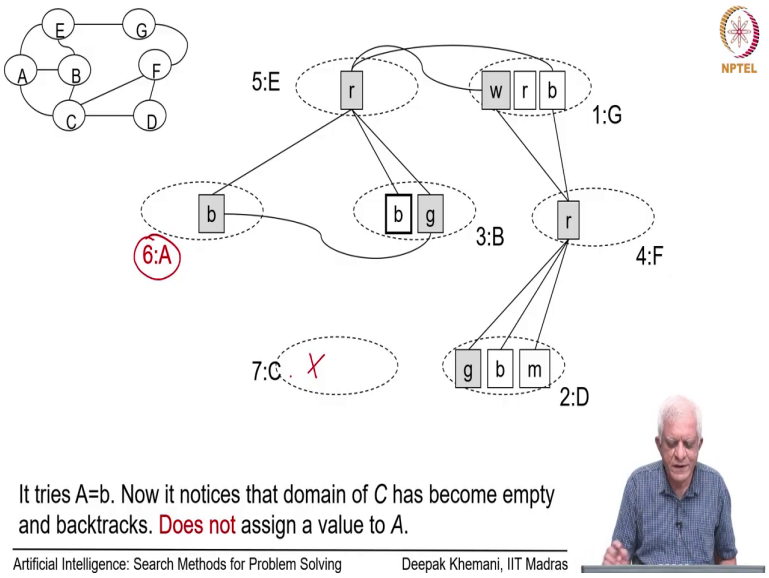
But it is not noticed that as yet essentially. So, it; so, it is again going to consider a value r for F.

(Refer Slide Time: 20:35)



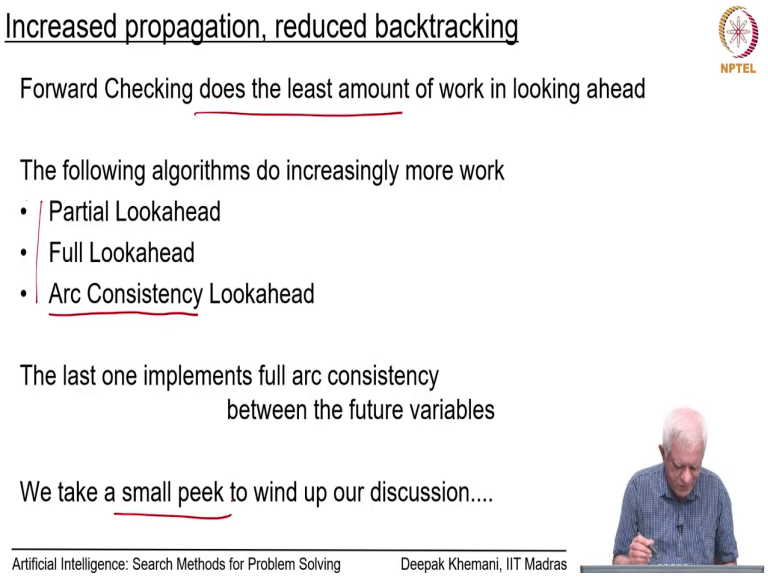
And then, it is going to consider a value of after F, it is E, r for this it is still does not see that these two are inconsistent so, it selects the value.

(Refer Slide Time: 20:51)



And only when it comes to variable 6 which is A, then it deletes the value for C, it notices that it is empty and then, it backtracks.

(Refer Slide Time: 21:13)



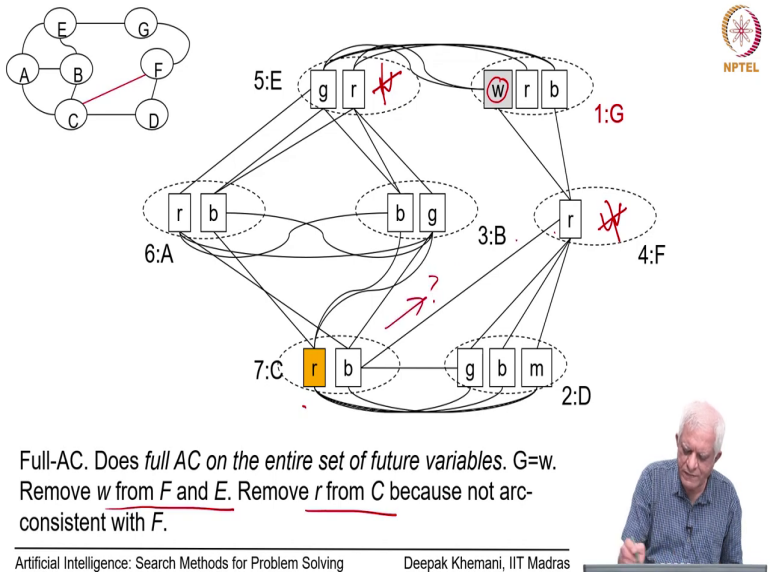
So, this gives a little bit of flavor of how forward checking works essentially. Let us without going into the details look at another example to illustrate the idea that if you do more propagation, then you have less backtracking essentially. So, our algorithm wents up to variable 4 and then, backtrack essentially.

Eventually the value of the 1st variable was going to be changed. Forward checking does the least amount of work that you can do and there are variations to that which are these four three; three more algorithms partial lookahead, full lookahead, arc consistency lookahead.

So, the difference between them is that forward checking only looks at the current variable and prunes the future variable. These algorithms do some amount of relation between different future variables and they are partial or full or full arc consistency lookahead. So, the

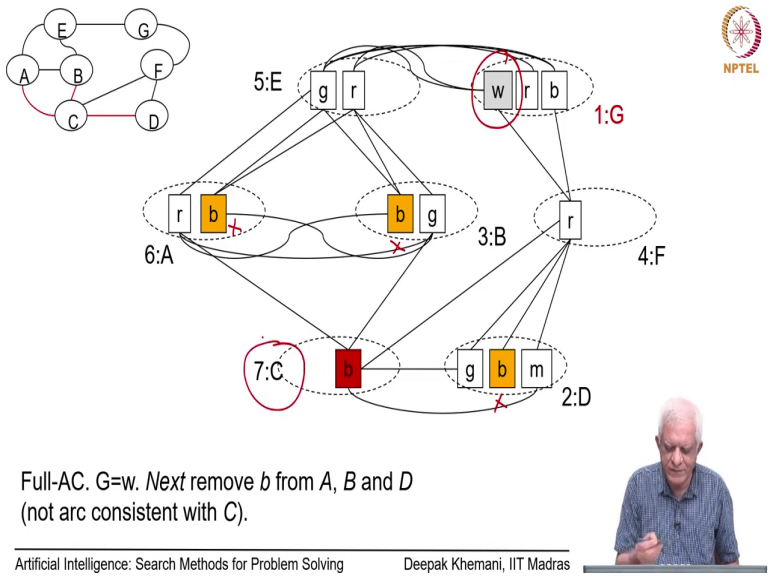
last one this full arc consistency and we will just see a glimpse of what happens when we were to use this form of lookahead which does more work than forward checking.

(Refer Slide Time: 22:21)



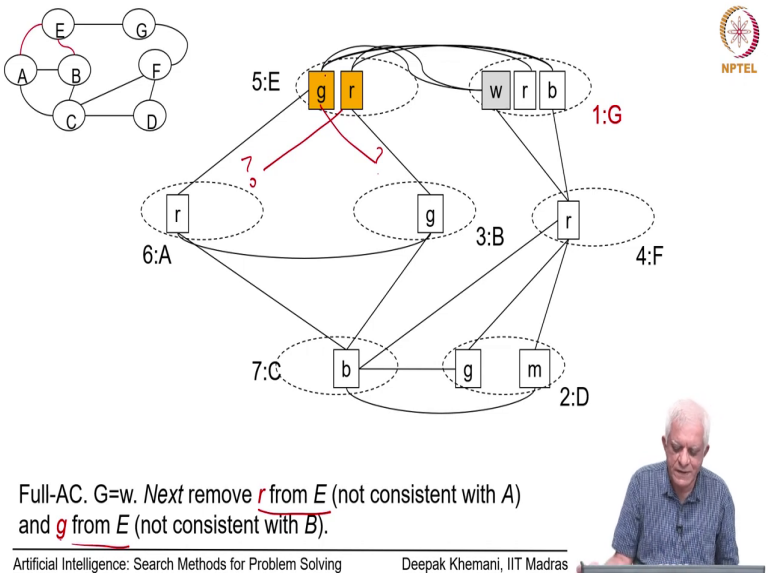
So, we take a small peak.We will not go the whole length. We have selected this value w, the first value for the 1st variable and we have remove w from here and then, it sees that because we have r here, so, we have removed w from F and E which we just did. Then, you remove r from C why? Because we are doing full arc consistency for future variables and this value does not have a corresponding value n four so, we remove it from there.

(Refer Slide Time: 23:07)



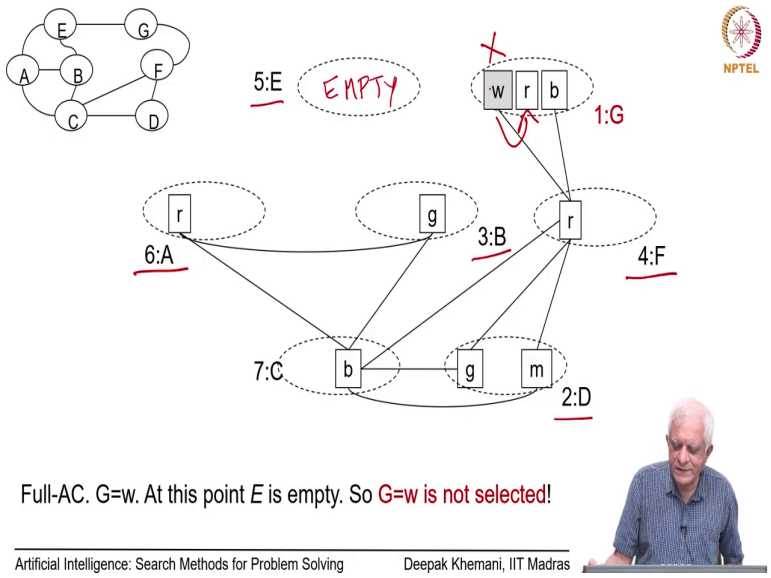
Then, we see that because these values do not have a corresponding value for C, we remove these three values from the domain of C. Remember that we are still doing AC, we have only assigned or picked up one value which is w for the 1st variable and all this reasoning is happening inside select value which would be called AC function. So, we delete these three values of B because for those values there is no value in the domain of C which has only B essentially so, the that vanishes.

(Refer Slide Time: 23:52)



And at this point after that we have these this thing we are still doing AC. Next, we will remove r from E and g from E. Why will we remove r from E? Because it has no value in variable A and g has no value in variable B. So, we will end up doing that when we make E, when we do revise of e with respect to A and with respect to B.

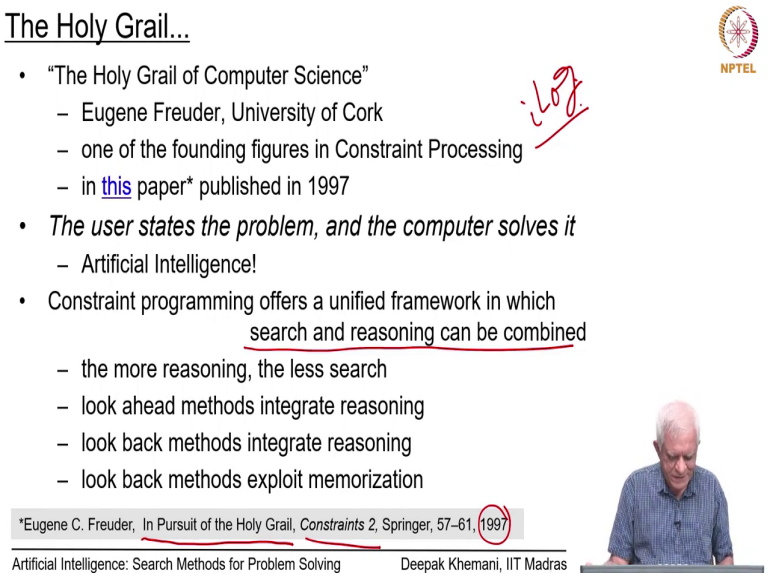
(Refer Slide Time: 24:37)



And we will notice that there is an empty domain. So, full AC will not even venture into assigning values for the 2nd variable, 3rd variable, 4th variable, 5th variable and sometimes 6th variable with forward checking ended up doing. Simply by a process of reasoning, it is decided that this will not work so, it goes and picks the next value.

So, I hope this gives you a flavor of the fact that lookahead arc consistency will do more work while selecting this value w and it will backtrack earlier. Forward checking did less work, it only looked at related variables and prune future domains, but it ended up doing more search. So, this tradeoff between search and reasoning, I hope is clear with this example.

(Refer Slide Time: 25:49)



So, we will end our study of constraints here. I want to end with some quotation from Freuder who is still active at the University of Cork. He is one of the founding figures of constraint processing. The AC 3 algorithm is credited to him and he published a paper in 1997 which is titled as you can see In Pursuit of the Holy Grail. It came in the Constraints journal in 1997 essentially.

And he says that this is the holy grail of computer science that people who are working in computer science want to do the following and it is expressed as this that the users states the problem and the computer solves it that is according to Freuder the Holy Grail and according to him, it is the means to go about doing that is using constraint processing essentially.

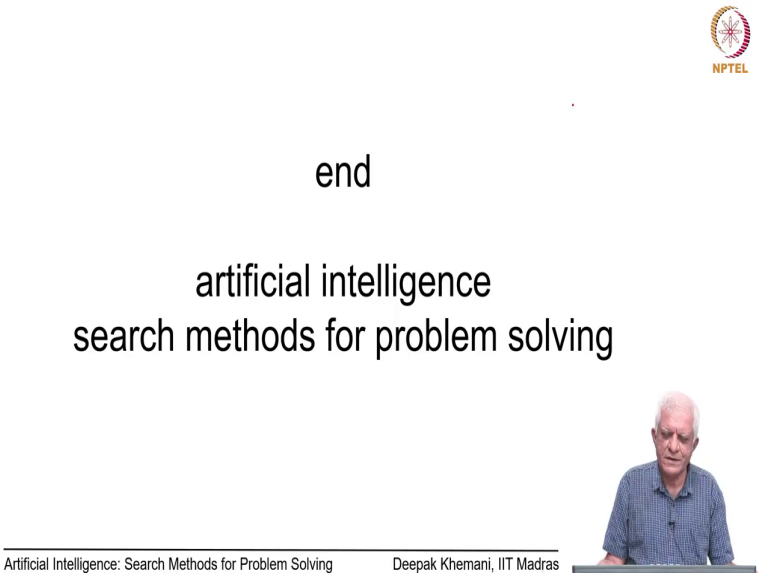
But you can see that in some sense, what he is saying is that that the holy grail of computer science is to build AI and by AI, you can say that we mean something like this that the user simply states a problem and the computer gives this user a solution.

So, that will definitely be AI and Freuder says that this is the holy grail of computer science and they have been having conferences every 2-year which is moving towards the holy grail and so on. So, just look up the holy grail of computer science and you will get some links to forth coming conferences or even past conferences.

And what we have seen in this last week is a constraint programming offers a framework in which search and reasoning can be combined and this is what is the most appealing feature of constraint satisfaction constrain processing. Apart from the fact that it gives us a unified framework for posing problems and solving them.

There are companies which will give you solver. So, for example, there is a company called iLog in France which you can get a solver from them, but there are also open source solver available elsewhere. But constrain satisfaction is a nice blending of search and reasoning and that is what I would like to end this course with.

(Refer Slide Time: 28:19)



So, here we are at the end of 12 weeks. We have looked at search methods, various aspects of it. I hope you enjoy the course and benefitted from it essentially. A goodbye.