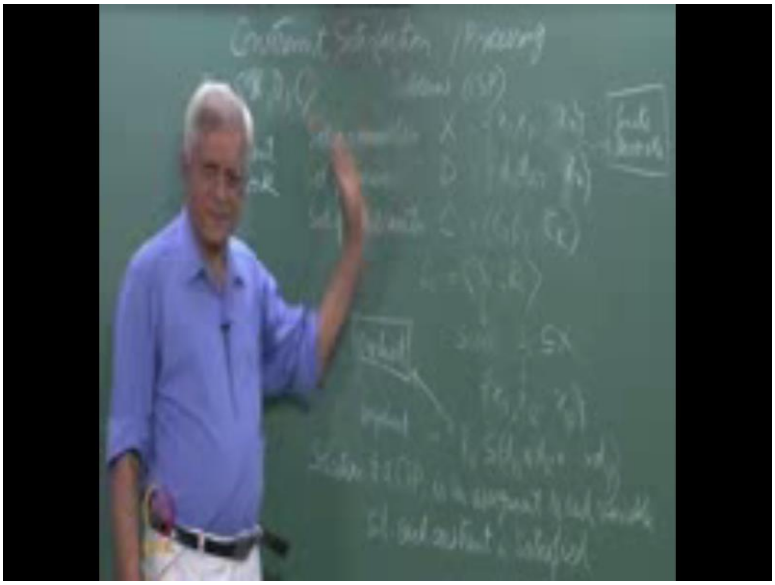
Artificial Intelligence
Prof. Deepak Khemani
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture - 39
Constraint Satisfaction Problems

(Refer Slide Time: 00:21)



So, today we want look at different approach to problem solving and this approach is called constraint satisfactions or some people called it constraint processing. And we talk of constraint satisfaction problems and we call them CSP essentially. So, this constraint satisfaction approach to solving problems is unified way of representing problems as constraint satisfaction problems as we will see today. And then solving the CSP essentially so essentially what this says. So, for example, in state space search we talked about states end, moves end and going to new states and so on. And then we looked at solution space search and these method in constraint satisfaction problems everything is expressed in the following way. There is set of variables x, x one, x two, a finite set, a set of domains d which you will denote by d 1 d 2 and so on. And the meaning of this domain says that variable x 1 can take values from domain d 1 variable x 2 can take value from domain d 2 and so on. So, each variable has it is own domain it does not have to be the same set of values.

So, one can be numbers another can be colors another can be day of the week; another can be name of the students; another can be the grade that the student has got anything in the variable. And each variable has a domain from the third thing is a set of constraints c which you will say c 1 c 2 let us say up to c k I will come to constraints in a moment first let us talk about variables. So, we will assume that there is final set of variables which is always the case so x n which means d n. And we will also assume far as for as we are concern that this domains are discrete domains essentially and not only discrete. But we will assume that they are finainic essentially which means that the set of values that available can take we will assume it is finite essentially. Because of the particular kind of sub problems that you want to look at and the solutions to those problems essentially now obviously many problems can be posed as variations of finite discrete domain. So, let us first discuss the constraint.

So, each constraint c i is a pair S i and R i where S i is called scope of the constraint. And it is basically a subset of x which means that a constraint is defined over a subset of the variables and that subset that particular sub set was for the i t h constraint is called the scope of that constraint c i and R i. So, this subset let us say this sub set is made up of variables x i 1 x i 2 x i p. Let us say that there are p variables each of them has a corresponding domain and the relation R i is basically defined over those variables. So, it is a sub set of d i 1 cross d i 2 cross d i p that is the most generate way of defining a constraint satisfaction problems a set of variables a set of domains for each variable. And a set of constraint defined over subsets of variable. And we have just using the generic definition of a relation here simply saying that is the subset of the gross product of all the domains in practice this could be explicit or implicit.

So, we will assume that the relations are explicit as for as we are concern for this discussion. So, for example, if you say that there is a number between 1 and 5 or the days of the week can be numbered from let say 1 to 7 when you will simply list this set as 1 comma 2 comma 3 up to 7. We could have said it in some implicit way like greater than 0 and less than 8. But we will not go into those things here and in many case it does not really matter. So, we will assume that the domain the relations are available to us as explicit pairs of tuples which is the subset of this cross product of the domains. Now, you will you must have for example, solved set of linear equation they can also be seen as a

constraint satisfaction problems except that the domains may be continues in linear programming. Whereas, domains are discrete in integer programming and they have their own so specialized kind of constraint satisfaction problems have their own methods for solving them essentially

So, for example, you know how to solve set of linear e Q inequalities or a set of linear equations. We will not going to specialize methods they eve those specialized methods obviously more efficient than the general methods that we want to look at that. We want to explode is that for general kind of constraint satisfaction problems. And we will be interested in those problems where the relations are explicit and the domains are finite and discrete. We will called this a finite constraint satisfaction problems and we want to look at ways of solving those problems what is the solution a solution to a CSP. So, we often say that a CSP is denoted often by R as this triple x d and c. An very often we use the term constraint network it is the standard term which the community uses. So, we will also stick to that but when you say as a CSP or constraint network basically mean the same thing this is triple of set of variables set of domains for the variables. And constraints was this variables solution to a CSP is an assignment to each variable obviously from their domains such that each constraint is satisfied.

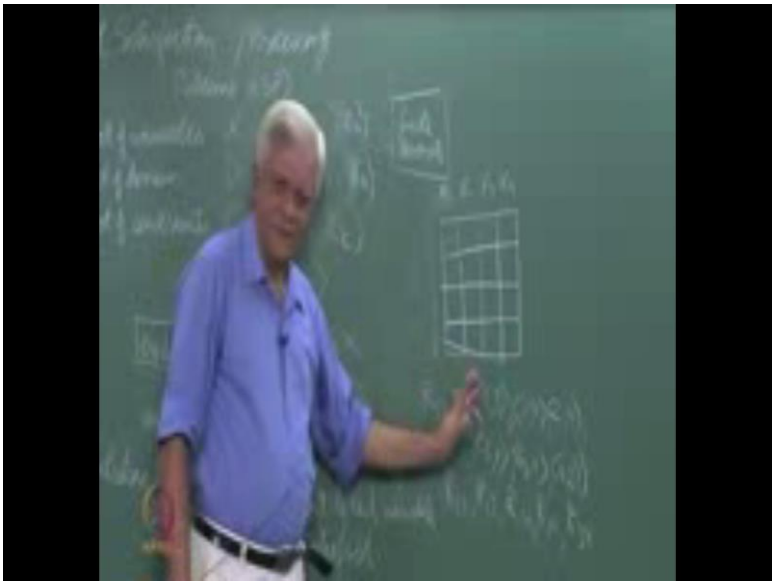
So, they will define what we mean by constraint in a moment essentially. But that is a general idea of a constraint satisfaction problem again to repeat we have a set of variables we have set of domains for 1 domain for each variable from which can take values. And we have a set of constraints defined over a subset of those variables without loss of generality will assume that for each possible scope there is only 1 constraint define. So, for example, if I take variable x 1 and x 2 i define only 1 constraint on that I will not define 2 separate constraints, because you can always combine the 2 constraints into 1. So, we are assume that there is 1 and the solution to CSP is an assignment to each variable such that each constraint is satisfied essentially. Now, this is very generate way of looking a things but it is very useful, because it turns out that a lot of problems can be as constraint satisfaction problems.

And we can now subscribe to this strategy of saying that if you solving a new problem just as a CSP then takes an of the self CSP solver and use it to solve the CSP essentially.

So, we can capitalize upon the expertise of people who have worked on constraint satisfaction problems and use their solutions directly. So, only thing when you need to do is so to pose it has a CSP and he turns out many, many problems can be pose as finite discrete C S P's for which we can use the methods that have discussed in the committee. We will not have time to discuss those methods again as I said this is just giving you an exposure to this particular field and for those of you who are interested you should come to the planning and constraint satisfaction course like semester in which you been look at all the defined methods which we will not have time to discuss here.

So, as in a side you should also observe that is the problem is a special case of a CSP and what is what kind of CSP it is? It is the CSP in which the domains each domain has 2 values 0 or 1 or true or false or whatever. And the constraints are defined in terms of the logical operators that we talk about and or and not and so on. And if you have such a problem where you have this then you have a sat problem essentially. So, the sat problem is special kind of a CSP and again of course sat has its own specialize approaches to solving them. So, there are approaches to sat and all kinds of things. So, again we will not going to the special ways of solving things essentially but we just observe that that is also a CSP.

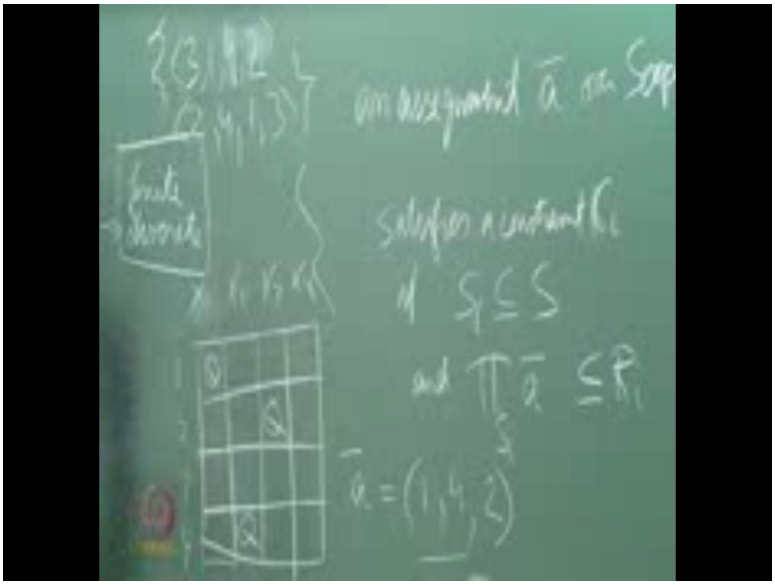
(Refer Slide Time: 11:51)



Now, our favorite problem one of our favorite problem small problems is this n queen problems. So, if you look at this problem let say we are looking 4 queen and how can we represent this? We can say that there are 4 variables let say x 1 one for this column x 2 for this column x 3 for this column x 4 for this column. And let us say the values the that can take a ne 2 3 4 4 numbers then we can express this as a CSP there equine problems by specifying the constraint So, I will use the short form R 1 2 to stand for constraint number 1 which has the scope of variables x 1 and x 2. So, will uses directly in a relation here so it sort of clear between us. Then we are talking about a constraint which is over the first 2 variables what us those constraints we can simply specify it as saying that these pairs are allowed. So, for example, you have placed the queen on first row the first queen on the first row second queen I can only place on the third row.

So, 1 comma 3 is allowed or 1 comma 4 is allowed or 2 comma 4 is allowed or 3 comma 1 is allowed or 4 comma 1 and 4 comma 2. So, this is what I meant by saying that we have it available to us explicitly it does not have to be implemented like this. As far as it is only for our discussion that we are assuming that this constraint between the first queen and the second queen i e expressed available to us as a pair of values that is 2 variables can take. So, each variable can take 4 on of this 4 values the domains are all the same and this constraint is expressed like this likewise you will have to express R 2 3 R 1 3 R 1 4 R 2 4 and R 3 4. So, there are 6 constraints, because you can choose 2 variables in 6 place and all those 6 which I am leave as a small exercise for you is to be can be expressed as a solution essentially. And then of course the problem is to find the set of values for this variables such that the values satisfy all this constrain which means if I taken a value for variable 1 and some value for variable 2 let us call it i and j. Then that pair i j must occur in set of topples here then that essentially.

(Refer Slide Time: 14:42)



So, we have some definitions we say that a constraint or we say that assignment. We will use a bar as an assignment which will be a short form for a set of so a over scope s. So, basically this means that this scope as says on which variables this assignment is done and this assignment selects 1 value for each variable from their respective domain essentially. So, explicitly I would say something like ah x 1 is equal to some value a then x 2 is equal to some value b and so on. But implicitly we will just assume that we will as a vector of a b and so on. And we assume that somehow we are able to specify what the variables are essentially it does not really matter the only important thing is that assignment is over a scope S. The scope S basically says that these are the variables to be giving value essentially. So, an assignment a a over a scope S satisfies a constraint ci if the following wholes that the scope of this constrain which you will call S i is a subset of this scope S which means every variable in the constraint has a value and we will use the term pie to talk of a projection.

So, I am sure you are familiar with a notion of a projection here of a bar over this set S i. So, why by this we mean that from this assignment a which is over some set of variables which you called S select only those values which correspond to those variables which belong to S i. And this is projection of a bar onto the subset of variable is a subset of R i so an assignment satisfies. So, if I given assignment here for example, i put a queen here

and i put the queen let say here and i put queen here So, this assignment says a is equal to 1 for the first queen 4 for the second queen and 2 for the third queen that is an a bar for this particular assignment and I can say that this assignment a satisfies the solution R 1 2 why because if I take the projection of the of the these variables on the first and the second variable which is 1 and 4 I can find that 1 4 of course in my relation R.

So, it satisfy the relation R then we say that an assignment a bar is consistent. If it satisfies all constraints in its scope I will not expand upon this. Basically we are saying this a bar has a scope S and for whichever constraint that scope of that constraint is a subset of S. It must satisfy that constraints which means projection over those variables must be belong to that particular triples essentially. So, you can see that this particular assignment which I have here where first queen is here the fourth queen second queen is here and the third queen is here is consistent. So, I have not written those constraints, but you can see that between 1 and 2. It is satisfying the expected constraints what is the constraints that the queen must know the attack another queen which here express explicitly here you could have express. It has a relation which says x if that is the array then xi not equal to y i or something like that essentially if x 1 xi why location of the queen that they are not on the diagonal. They are not on the same row they are not on the same column you could have express it something like that essentially but we have expressed explicit here.

So, we can see that the first 2 queens are not attacking each other .So, they and be that shown by this the second and the third also not attacking. So, if you are written R 2 3 you could have seen that and the first and the third also not attacking and they would have been present in this essentially. So, we say that this assignment a bar is consistent essentially sometime you use the term partial assignment which means that if an assignment only 2 a subset of the variables. But we will use the time term interchangeably or a partial solution some time we say. So, an assignment is consistent if it satisfies all the constraints which call within the scope in this case there are 3 constraints between queen 1, queen 2, queen 1, queen 3 and queen 2, queen 3 and all the 3 are satisfied. So, this assignment is consistent now observe that just because it is consistent it does not mean it can be a part of a solution. Because you can you surely know that this cannot be extended to a complete solution essentially you cannot put a

value and you this thing you put it here it will attack these 2. If you put it here it will attack these 2 if you put it here it will attack this if you put here it will attack these 2. So, you cannot place a variable.

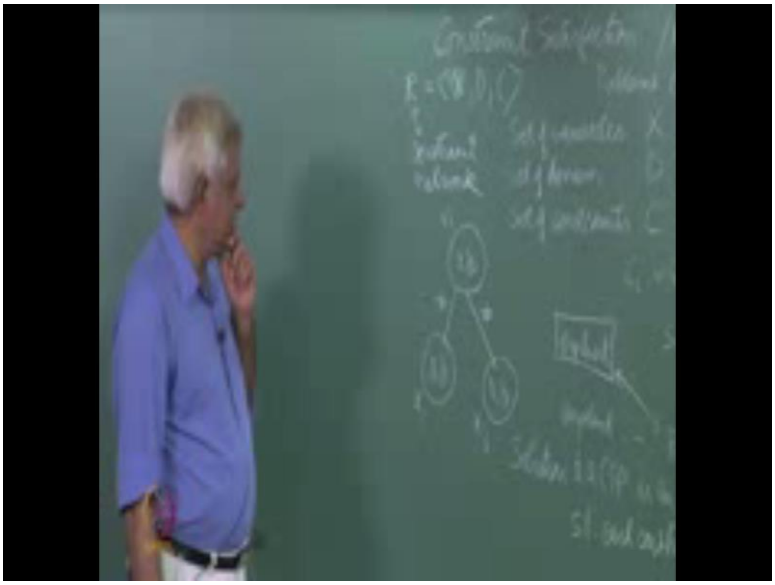
So, this a consistent partial solution but it is not a consistent full solution. A full solution is a consistent assignment to all the variables which is another way of saying that what you are said here that a CSP solution to a CSP assignment to each variable which satisfies all the constrain essentially which is the same thing that is saying that if you assign all the variables and it is consistent it is a solution essentially. So, how do we solve? So, again we want to look at general purpose methods of solving CS P's just like with it for state space or we did not care what the state was and where the moves end came from as long as we had move zen function and as long as we had a goal test function. We said we will use the search algorithm that are used likewise we have just made an observation that many problems can be posed as C S P's I have mentioned sometime during planning. That planning can be posed as a CSP which we will not have time to going here but it can be planning can be also posed as sat which is the special case of C S P.

Both are slightly different formulation and we will look at in due course couple of more problems which can be posed as CSP but we want to look at general purpose ways of solvency C S P's essentially. So, let me discuss 1 more problem before we go just to to highlight what are the issues involved. So, let say we are doing this map coloring map coloring can also be posed as CSP before we come to map coloring we are not said anything about the scopes of this constraints essentially do we have any constraints on the scope of this constraints. I have posed this problem where the scope was 2 variables between queen 1 queen 2 or queen 1 queen 3 or between queen 1 and queen 4 and so on and so forth why not between 3 variables between 4 variables in this. You can express constraints and in more variables for example, if I had express the constraints as a constraint of 4 variables I would actually be expressing the solution itself, because I would have just basically the set of solutions in that essentially. So, in fact, this kind of illustrates the idea that we pursue in constraint satisfaction problems that you do not have to specify the problem completely as long as you give some specification of the set of constraints it is a task of the solver to elicit a solution out of that essentially.

So, what is the solution to this? You know that solution to the CSP there in fact only 2 solution 1 is 3 1 2 4 and the other is 2 3 1 4 2 3 1 4 2. And the other 1 is if I start with 2 and 4 1 and 3 I have only these 3 now this particular relation is a relation on the set of 4 variables this is called a solution relation where R refers to this constraint satisfaction problem. So, correspondent to a constraint satisfaction problem there is a solution relation and it is task of the solver to elicit the this solution from the CSP I have only specify binary constraints here between 2 variables. And the solver will eventually tell me the these are the 2 possible solutions it may not express in this form but it gave me an assignment of all possible variables and we will do that. So, without again loss of generality we will assume that we are working with binary relations.

A binary C S P's and a binary C S P's we mean that C S P's which have scopes of size either 1 or 2 essentially scope of size 1 basically says that I am defining subset of domain scope of size to subset of the cross product of 2 domains. So, binary CSP has scopes 1 or 2 and such a CSP is called binary CSP and it has been shown that any higher order CSP can be converted to a binary CSP by adding more variable essentially. So, I will leave this as a small thought exercise for you to work on how can I converted into how can I convert this. For example, into a binary CSP if this was my original CSP which is the solution itself how can converted into a binary CSP. So, we will stick to binary C S P's because we know the there is a whole lot of methods which obtain those problems and other problems can be express as a binary C S P's. So, this map coloring problem is naturally pose as a binary CSP ao.

(Refer Slide Time: 26:16)

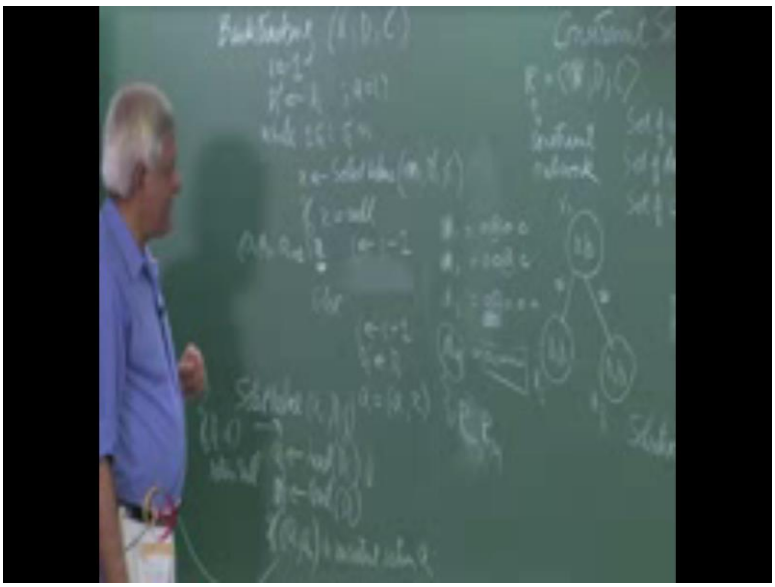


Let say I have 3 countries will not name them and let us say my formulation of C S P's that I have a relation between this, this, and this. So, let us just called this x 1 x 2 and x 3 and let us say they have the same domains which is let say just 2 colors are allowed red and blue. So, I think that I think the domains inside this circles and the relation is not equal to now we can see that this problem has 2 solutions. You can color these 2 red and this 1 blue or you can color these 2 blue and this 1 red essentially which means implicitly there is a constraint between these 2 variables. But that constraint is not explicit it has not been mentioned in the CSP the CSP only says that I have a constrain between x 1 and x 2 and between x 1 and x 3. I do not say anything about x 2 and x 3 but eventually off course, this can be again elicit ate through a process of solving the constraint. And we will try and see whether we get some insight into how we acquire new relations in the in some manner essentially. So, how so let us look at these 2 problems. So, we have now you know in those case 4 variables in this case 3 variables and we have some domains and so on. How can we solve a C S P?

The simplest approach you can go back to state space search. You try you sign a value for the first variable; you have sign a value for the second variable; you assign a value for third variable assign a value for fourth variable and so on. Finish an assignment in state space search we would have finish the assignment and then check whether it is goal

state or not in constraint satisfaction problems. We can backtrack earlier why because the moment you know that some constraint is not being satisfied or some we have defined the lotion of consistent assignment or consistent partial solution. The moment we know that the assignment is not consistent we can backtrack from their essentially. So, you do not have to assign values to all variables. So, in other words you do not have to assign values to all variables. So, in other words you do not have to as in this case of course you will backtrack only when you see the fourth when you try to put the fourth queen. But if you let us try a 6 queen problem, you will see that there is some kind of processing you can do when you can back track only or you might be able to backtrack only.

(Refer Slide Time: 29:24)



So, the simplest algorithm for solving a CSP is called backtracking it is a official name of name of this algorithm essentially. So, everybody uses this term essentially. So, what do we have? We have constraint satisfaction problems given x given d and given c essentially. So, let us this outline this algorithm So, 1 issue that keep in mind is that let say you have assigned some value to x 2 and then you are trying for values of e c. And let say you assigned the third value to x 2 let us say in a set of domain you are trying the third value 2 x 2. Then you go forward to x 3 and you find that there is no values to x 3 and you need to back track to the fourth value of x 2 and then try all values of x 3 again. So, the fact that you every time you go back and forth you have to keep track of what

values left for you to try this algorithm does takes a very approach it makes a copy of the domain every time.

So, it start up by saying i gets the value 1 di is a copy of di in this case d 1 and while i is in this range n it does the following it calls off function call select value. So, let say x gets a value from a function called select value I should also initialize a equal to empty initially I do not have a value for the partial solution given a select value for this x i x i d i c. So, what this select value function will do is it will give me the next value for xi from its domain d i; d i prime actually with this consistent by this you mean it is consistent with all the previous variables that have been given values. So, if xi is equal to 4 let us say then if there is a constraint between 1 3 and 4. Then this value for the first variable must be consistent with those first and third value since essentially the way that you have just defined there essentially. So, it will written 2 things it will either it something called null which means it cannot find a value then we say i goes to i minus 1 D i prime hence i goes to i plus 1 you have got some value for x 1. You got some value for x 2 you got some value for x 3 or let me try and draw more elaborate diagram let say x 1 has theses values x 2 has or d 1.

So, let us say you are looking at x 4 now and let say you have always progressing from let say your select value function always fix the value from left to right, because they are in some list and it has selected this value for d 1 and this value for d 2 and this value for d 3. And now we are trying to find if any of these values for d 4 is consistent with this assignment we have here essentially. So their 2 thing that can happen either it can it will say that no nothing is possible here which means I was go and look for a new value for d 3 here, because we are tried this. So, we are doing simple that first search over this 3 of possibilities where you start with the first 1 here then the first 1 here then the first 1 here and then keep back trap. If you cannot find a variable value here you must go back to this 1 and try the next value for this essentially which means you must set i is equal to i minus 1. So, I do not need to change this right.

So, let see what select value is going to do essentially what it will do? So, let say you are looking at xi from di and c it will say let us say a gets head of di if it is a list it takes a first value and it says di di before that it must have check which says that if di is equal to

empty then return null. So, if any point di becomes empty it must return null otherwise it it should go into the this loops looking for consistent value if a i if a sorry let us call this a a i if a comma a i. It means the whole solution that has been constructed so far which we will as a. So, this 3 values in this examples plus this fourth value that we are we are just picked from its domain we put them all together and that is a new partial solution if this is consistent then return a i. Well I called it x here you could have called it x it does not matter. So, it is in a loop inside taking out values from the domain taking out and showing them to speak essentially. So, when it was doing d 3 that d 3 prime had thrown away these two values from the domain.

So, they were not there in d 3 prime anymore. So, when it backtrack it must try the next value essentially then it must try the next value then the next value. And if you cannot find there is must backtrack that is the standard. Therefore search chronological backtracking that we have been talking about this other form of backtracking that we had mentioned dependency directed backtracking was actually invented for solving for improving this algorithm backtrackings. So, that instead of if you cannot find a value for d 4 just to give an example that let say only constraint that d 4 participate it is in let us call it R 1 2 4 which means this variables 1 and 2 and 4 essentially or if you want to work with binary constraints and let us say R 1 4 and R 2 4. So, let say the only constraints d 4 participates as in with variable 1 and variable 2 with variable 2 then if you cannot find the value for d 4. What is the point of looking for new value for d 3? Because d 3 anyway not influencing the consistency of the choice of d 4 essentially the only thing that d 4 is getting influences by this relation R 1 4 or by R 2 4.

And the fact that you cannot find value for d 4 means that one of this relations you cannot satisfy which means actually you should really jump back here. But of course we will not get into that here, because it needs a little bit more formulization. So, that is the general idea of dependency directed backtracking that if you can keep track of which constraint is being evaluated. Then you can jump back to one of the variables in that constraint in this case they have binary here essentially. But in our example we just to i is equal to i minus 1 which is like doing chronological backtracking if you cannot find a value for d 4 try a new value for d 3 and so on essentially. So, that is a i mean I have written may be it is not quite correct something is missing here. But you can work that

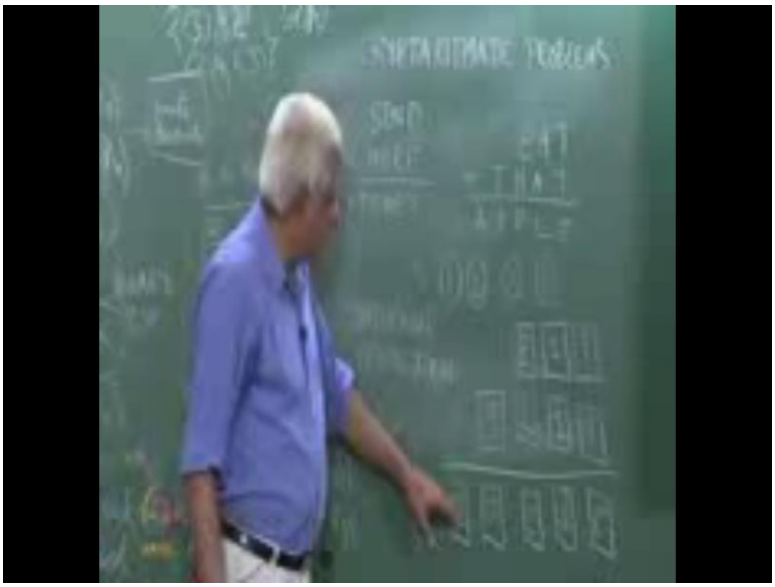
out work out the details, but the basic idea is to go down this set of choices looking for a new value for the is variable. If you cannot find it if you can find it go to the next variable which is what we have doing here i is equal to i plus 1 di copied the domain and solution that we have made and then keep going forward in that session if you cannot find it backtrack. So, it is simple depth first search being done over this structure of the now this algorithm is actually the simplest the starting point for constraint satisfaction problem algorithms.

The other algorithm which improve upon that do the following is that for example, some algorithms when they are doing select value though look head to see whether or given choice will in future conflicts with some vary variable essentially. So, that is one kind of you know improvement which is called look ahead algorithm then we have this kind of intelligent backtracking algorithm. So, dependency directed backtracking algorithm that we are talking about essentially and there are some other things that we will discuss. So, that you get a flavor of that essentially, but this by and large is a the simple algorithm. Now observe that backtracking happens the moment of partial solution is what is a partial solution we are talking about here it is a value for a 1 for a 2 and up to a i minus 1. And we cannot find we cannot extend this to a i or the is value for the is variable or 2 x i. Let say we cannot find a value for this is variable and we backtrack at this point itself we do not go further to see you know, because at this point as you can see no by definition not right no a solution is a consistent assignment to all the variables. Or in other words a solution is an assignment to all the variables.

So, that every constraint is satisfied and a partial solution or a or a partial assignment is consistent. If it satisfies all the constrain for whom values have been selected or who's scope fall within this essentially here. So, we say that constraint ci assignment a bar satisfies ci if the scope of the ci is has a value which means it is already in the assignment. And the projection of this assignment over this scope of S i is a subset of this which means it which actually I should say belongs to this belongs word here belong to this and in. So, that is only for with respect to 1 constraint ci and assignment is consistent if it is satisfies the all the constrain with satisfy this property which means to scope are contained in this and who's projection is contained in that correspondent essentially. So, it can only be that it can never be the case that a that assignment which is

a partial solution which is not consistent can never consistency. So, when I was talking about beginning on CSP one of the thing. So, I had said was that CSP is very interesting because it allows you to combine search with reasoning now that reasoning part web have not seemed. Let me just give you a motivation for that and then in the next class, we will see how that is done. So, one of the another problem which is pose as a CSP you know if you solving a crossword puzzle. Or if you are solving sudoku or something you do a lot of reasoning in fact you are supposed to do more reasoning than search.

(Refer Slide Time: 45:08)



So, they these problems are similar they are called some of you must have looked at such problem and if you just search on the web you will find examples. So, for example, you might say send plus more. So, these are sort of arithmetic problems in which we the digits have not been revealed, but they have been replaced by letters and your task is to find out what digit does each letter stand for. So, we assume that each assignment is distinct that that 1 letter stands for exactly 1 digit and so on. So, how would you solve such a problem essentially? So, if you want solve this or let me take another example. So, here is a another example. So, this plus this should give you apple essentially. So, if you wanted to solve something like this then what you would do is you would create a variable for each a place for each letter of course the number of variables are number of distinct letters in this problem, but you want to find vales for them. So, these boxes that

an creating are boxes for values this a plus and obviously you must create some more boxes which are for carry over.

So, 4 boxes for carry over and you can define the domains of so these variables so the domain of this is 0 slash 1, because we adding 2 numbers we can never have carry over moves and 1 and the domains of these are 0 to 9 essentially. So, the kind of reasoning that Sangeetha was talking about we can do here essentially. So, for example, you can observe that this must be 1, because you can never be get a value of more than 1 which means this carry over must of course we are not interested on this. But never the less the what the moment we have said this a a equal to 1 we know that this is equal to 1 and this equal to 1. We can fill in the value there t is 9, why t 9? And it must be distinct this a p and t are distinct they cannot stand for same digit. So, which means this p cannot be 1 and in any case p cannot be 1 So, if t is nine then what is p is 0 this must be 1 we have 0 here also know and we have nine for t here we have nine here nine here So, this must be eight this must be 1 So, we know this must be 3.

So this is 0, now, we need to fill in this is 8. So, this leaves only 2 for this. So, we have off course, solved in this problem now, and we are solved it through a process of reasoning certain some rezoning with it you know this can only be 1 and know this kind of stuff. Now, if you off course, were not should I say smart enough to do this kind of reasoning you could still have try this algorithm backtracking here you could have said try a value for a try a value for b and so on and so forth. Off course, we have to express the constraint, how do we express the constraints that this plus this must be either this must be either equal to this plus this or this plus this this plus carry over 1 and modulo 10. So, we have to express that carefully essentially, but the point is that we a try to make here is that solving constraint satisfaction problems not necessarily have to be done through a search method like backtracking. They can be done through other processes like reasoning in some way now can this reasoning be done in general purposely essentially.

Of course, here we are exploding the rules of arithmetic and it turns out that yes indeed you can do a certain amount of. So, this process is called constraint propagation. So, we are saying that if this is 1 this must be 0; this is 0; this must be 9 and this must we are

propagating the constraint. So, what are we doing here basically reducing the domains of the variable we have reduce this from 0 to 9 to just 1 value 1 and just to 0 and so on so forth. And the moment we reduce the value of some domain some variable we can propagate a reduction to another domain. This idea of propagation is of course very common in reasoning. And we will see in the next class little bit about how this can be generalized. And we will look at another interesting problem that propagation demands that quite effectively. So, we will stop here with this.