Constraint Satisfaction Problems

Prof. Deepak Khemani

Department of Computer Science

Indian Institute of Technology – Madras

Module - 2, Lecture - 01

Okay so we are looking at constraint satisfaction problems and now let's start our study of constraint networks or CSPs – as we call them. We know that a network R is made up of 3 things. A set of variables X, a set of domains D and a set of constraints C.

So let's just quickly recap this. X is a set of variables. Let's say that there are n variables and D is a set of domains, one for each variable. Di is the domain of Xi essentially. And C is a set of constraints. Any number of constraints can be there. So let's say there are r constraints and each constraint is made up of a scope Si and a relation Ri.

So Si is the scope of a constraint and it's a subset of the set of variables X. Let us say Si is equal to xi1, xi2, xip. That means there are p variables over which the relation holds. And Ri is a subset of the cross product of the domains of the variables in the scope, Di1 X Di2 X … X Dip. So that is the standard notion of a relation, that the relation is a subset of a cross product of whatever it relates essentially. In this case there are p variables that are being related and so Ri is a subset of the cross product of those domains.

Let's take a small example. Let's say X has 3 variables, x1, x2, x3. Domains are sets. The first one has, let us say, <1, 2, 3, 4> and the second one also has some numbers, <1, 2, 3>. It doesn't have to be the same thing. So I can also have <a, b, c>.

So <1, 2, 3, 4> is the domain for x1, <1, 2, 3> is the domain for x2 and <a, b, c> is the domain for x3 essentially. Then let us say we have some constraints.

One thing that we will follow is when we have variables which have indices with them like x1, x2 and x3, we will use the indices as indices for the constraints as well essentially.

So let's say I have three constraints. Let's for the time being call them C1, C2 and C3. I might describe C1 by saying that the scope of C1 is S12 and the relation is between R12 essentially. So what does this mean? This means x1 and x2 is the scope of R12. So we will just use the indices to describe the scopes. We won't explicitly say that the set is x1 and x2. Instead we

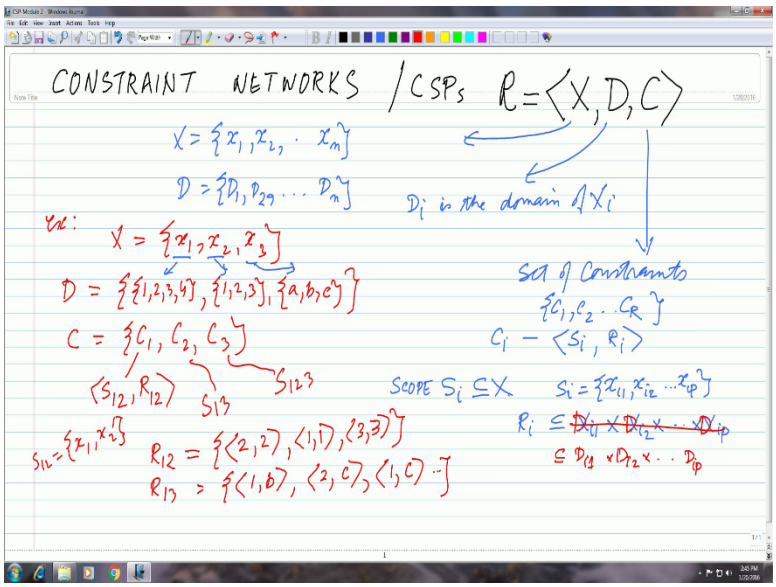
will simply say the scope is S12. S12 automatically means it's variable 1 and variable 2. So for example I might say S13. So it's possible that you can have any kinds of scopes and so on.

Then let's just give an example. For example, R12 is a relation between variables 1 and 2 which means it can take the first value from variable 1 and the second value from variable 2. So it's basically a set of tuples. So for example I might say <2, 2>. That's allowed, for example, or let's say even <1, 1> and <3, 3>. That's my whole relation.

You can see that actually I can express this relation in a short form. I could have said x1 = x2. But we don't do that. When we talk about finite domain constraints, we explicitly list all the possibilities and then say that this is the relation between them. So we assume that if you are writing a program to do this, then you will have some component which will convert the intentional from of the relation to an extensional from in which the relation is explicitly specified.

I could say for example, that for R13, 1 is allowed to go with b and 2 is allowed to go with c. There can be repetitions of course. 1 is allowed to go with c and so on. It's just a relation and not a function or anything from one variable to another. Any subset of pairs is allowed.

(Refer Slide Time 8:02)



Next let's talk about assignments. So assignments are also called instantiations and these are sets of variable value pairs essentially. So I'll just write it like this – {<x1 = 2, x3 = c}. I'm

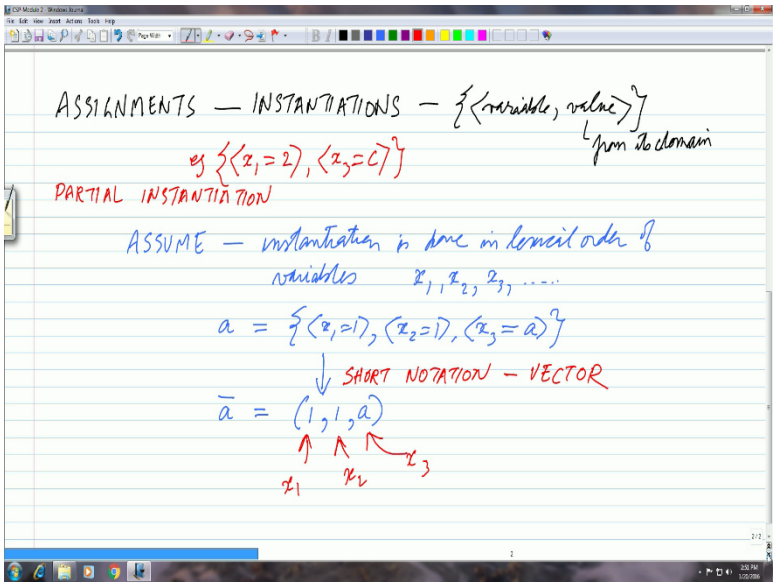
using this notation to say that it's a set of variable value pairs. Essentially you're saying that for some variable I'm choosing a certain value from its domain.

So for example, I might say {x1 = 2, x3 = c}. So this is an assignment or it's a partial assignment or partial instantiation. And if you assign values to all variables then it's a complete assignment or a complete instantiation.

We will assume that instantiation is done in, let's call it the lexical order of variables – x1, x2, x3 and so on. Which means that an assignment is like x1 = 1, x2 = 1, x3 = a. So this is from the domains that we saw in the small example.

Instead of writing it like this, we will use a short notation. And we will represent it as a vector. We will simply write it as (1, 1, a). And we will use a vector notation here. So it is understood that 1 is for x1, 1 is for x2 and a is for x3. So this is only for the sake of discussion. You can imagine that you have an algorithm which freezes the order in which you will assign values. Essentially we're going to look at search algorithms and then you can talk of the partial assignment and so on and so forth.

(Refer Slide Time 11:55)



So now we want to talk about what we can say about a given partial assignment or assignment. We will use the term assignment and partial assignments interchangeably. We want to talk about consistent assignments.

So to do that let me use an example. Let's say we have the 9 variables x1, x2, x3, x4, x5, x6, x7, x8 and x9. And let us say the constraints are as follows. C1 has scope x1 and x2. C2 has the scope x2, x3 and x4. C3 has the scope x2, x3, x7 and x8. And C-four has the scope x4, x5 and x6.

So let me just write their scopes explicitly. C2 is S234. Instead of writing S I could have written R also because it's quite clear that we are talking about a relation over the variables x2, x3 and x4 essentially. C3 has the scope S2378. And C-four has the scope S456. We are not talking about what the relations are but the scopes essentially. The relations will be subsets of the cross products of the values of x1, x2 and whichever are the variables essentially.

Let's say we are talking about a particular assignment a~ which gives values to x1, x2, x3 and x4, for example, some values in whatever the domain is – (2, 4, 3, 7). So this is for x1, x2, x3 and x4. So we have an assignment. We will use this a~ to stand for an assignment in general. What can we say about this assignment? We can say that a~ satisfies a constraint C in general if first the projection must be defined over S-C, which means it must give values to all the variables which are in the scope of that constraint. And two, if I take the projection of a~ on to the scope S, then that belongs to the relation that is for that particular constraint essentially.

So in this particular example a~ has got values to 4 variables x1, x2, x3, x4. So it covers the scope of C1 and the scope of C2 but it does not cover the scope of C3 and C4 because there are variables in the scope of C3 and C4 which it doesn't give a value to. If it doesn't give values to some variable in the scope, then we can't talk about satisfaction with respect to that. So only if the assignment is defined over the scope of all constraints and secondly if it is a consistent assignment, we say that the assignment is the solution of the constraint network, or in this case, a~ is a solution of the constraint network R if it is a complete assignment and consistent assignment. So if it gives a value to all the variables and satisfies all constraints then we say that a~ is a solution essentially.

We will use the character R to stand for the network or to stand for the CSP. CSP and network are the same thing as far as we are concerned and is equal to the set of solutions of R. The set of solutions of a constraint network is a relation which we call ρ and it's a relation over all the variables of the network and it contains all the solutions. So in some sense that's a goal of constructing the network, to describe this relation ρ because this relation ρ says that these combinations are allowed essentially.

So if you remember the example that we had done for 4 queens, ρ = { <3, 1, 4, 2> , <2, 4, 1, 3> }. So this is the solution relation. This is the set of all possible solutions to the four queens problem. If you remember we tried to model the n queens problem as a binary constraint network which said that we basically look at all pairs of two queens such that they don't attack each other but that was a binary constraint network which means the relations were expressed over two queens essentially.

So we had a relation between, for example, if you are looking at the four queens then relation between queen 1 and 2, queen 1 and queen 3, queen 1 queen 4, queen 2 queen 3, queen 2 queen 4 and queen 3 queen 4. So all these six binary relations we would have. And the individual constraints would have many more combinations than what you see in the solution relations. So in some sense the task of finding the solution is to arrive at this network or this relation or at least one of them essentially.

We also say that R expresses the relation ρ essentially. In some sense we are saying that the network that you have given us corresponds to a set of solutions which is ρ and R expresses ρ. So this gives us kind of a linguistic way of comparing networks. So if you give me two different networks, then we can say that the networks are equivalent if they express the same solution and we'll talk about it in the next class.

So obviously you can see that this solution relation ρ that we have expressed here for the four queen's problem which has got 2, 4 tuples is also a constraint network. You can say it's a constraint relation between four queens essentially. But the original problem we might have posed as a relation between 2 pairs of queens essentially. So all these are different networks and we say that they are equivalent if they express the same solution.

So while we are talking about n queens, let me also give you a small example here which says that just the fact that an assignment is consistent does not mean that it's going to be part of a solution. So for example, if I had a queen placed in the first row, a queen placed in the fourth row and a queen placed in the second row and if I had expressed my four queen problem as a binary constraint satisfaction problem, then this is actually consistent. It's a consistent assignment because you can see that no queens attack each other

So binary constraint formulation simply talks about two queens not attacking each other and in this example you can see that the 3 queens that are part of my solution don't attack each other. They respect the constraint C1 between queen 1 and queen 2, queen 2 queen 3 and

queen 1 and queen 3 essentially. So it's consistent. But you can also see that this is not extendable to a solution because we cannot actually place anything in the fourth column.

So the very fact that an assignment is consistent doesn't necessarily mean that it's part of a solution essentially and that kind of gives us an indication that constraint satisfaction problem is computationally not an easy problem to work with and therefore it's a hard problem. It's basically a combinatorial problem. If you just look at these 9 variables x1 to x9, let's say each of them has 5 values, then one brute force way would be to just try all combinations of values. Give x1 its first value, x2 the first value from its domain and so on. See if it satisfies the constraint and try all combinations. So it's basically a combinatorial problem in that sense or it's an exponentially growing search space essentially and our goal would be to try and find efficient ways of solving constraints.

So in the next class we will continue a little bit more with definitions and we will try to look at some properties of networks before we finally move on to algorithms. I'm sure algorithms are more interesting but only once you have the foundation which is ready for that. So we'll end here and take it up in the next class.

(Refer Slide Time 23:26)