Constraint Satisfaction Problems

Prof. Deepak Khemani

Department of Computer Science

Indian Institute of Technology – Madras

Module - 2, Lecture - 03

We are focusing on binary constraint networks and let's do a very quick recap of what we have done so far. We looked at the process of composition of relations. I want to look at an example today and also introduce the notion of a matching diagram. We have seen that there is one graphical way of representing a constraint network which is the constraint graph. In a constraint graph, every node is a variable two nodes have an edge between them if the two variables participate in the constraint.

The matching diagram is also a graphical representation but in this representation, we look at values as to which value is related to which value and it is quite a useful device to explain many of the algorithms that we will be looking at.

Let me take an example again. We have a binary constraint network between variables x, y and z and I hope the dual use of x doesn't cause a confusion. One is the set of variables and one is a variable here. And then to represent this binary constraint network using a matching diagram we have these three variables. Let us say there are three values a, b, c in x. It means Dx = {a,b,c} and likewise you can define Dy and Dz.

Let us say that we have Dy = {1,2,3,4}. So Dy is made up of these four values and let us say that a is related to one and b is related to three and b is related to four. What does that mean? We can express that as Rxy = {<a,1>,<b,3>,<b,4>}. And let us say z has three values also. Dz = {A,B,C,D} and let us say one is related to A, two is related to B and four is related to C. There is another value D which is not related to anyone here and another value d in Dx also which is not related to anyone here. So Dx = {a,b,c,d}.

Then we have Ryz, so I'll just use the shorter form now, Ryz = {1A,2B,4C}. Now if you remember we had said that the composition of these two relations which is Rxz is the projection on to xz, so Rxz should be a set of the natural join of Rxy and Ryz.

So what is the natural join here? The natural join part is that whenever they share a value, so Rxy has a one and Ryz has one A, so the join will contain the triple <a,1,A> and it'll also contain the triple <b,4,C>. This C is an upper case C because it comes from a different domain. And then you project it on to πxz, i.e., on variables x and z. So form the join we want to take x and z and what we get then is the relation {<a,A>,<b,C>}. This is the relation Rxz. So this is the composition relation and essentially what have we done? We have inferred a new relation or you can say constraint because relation and constraint are the same thing essentially.

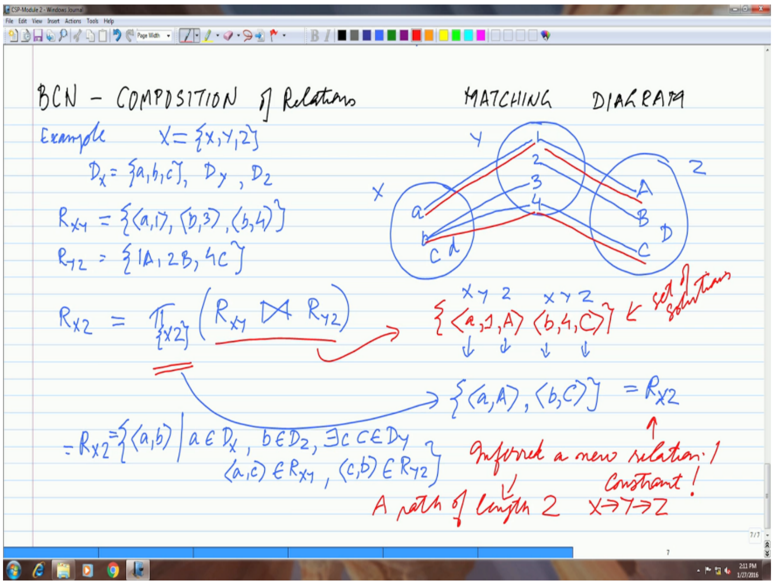
We had a relation Rxy and a relation Ryz and we have inferred a new relation Rxz. Now you remember we had given an alternative definition of this which I will just state here. So Rxz is the set of all such pairs <a,b> where a, b are variables, where a є Dx, b є Dz and there exists a c where c є Dy and there is the relation which means <a, c> є Rxy and <c, b> є Ryz. This is an alternate way of defining the same relation. So the first expression is from relational algebra. The second expression is a more explicit set based relation.

You can also see that the new constraint is a path of length two and the path starts with the variable x goes to the variable y and goes to the variable z. So we have composed Rxy and Ryz and we have found Rxz. And it is essentially a list of all paths of length two which go from x to z via y and you can see in the diagram the two paths that we have.

We had defined the notion of equivalent networks. We had said that two networks are equivalent if they express the same solution. So in this original network that we are talking about there are two relations Rxy and Ryz. We have now made this equivalent to a new network in which there are three relations Rxy, Ryz and also Rxz and the set of solutions for all these three are the same. Since both the network express the same solution we say that they are equivalent.

So we had defined the notion of equivalent networks. We had defined the notion of tighter networks. So tighter networks are those which are more explicit in some sense, whose relations are subsets of the less tighter network essentially. And then we had defined the notion of a minimal network. That was the tightest possible network that you had over this.

(Refer Slide Time 9:29)



We had also discussed the result by Montanari. Let me just do that again before I define the projection network which is what we're interested in. Given n variables each having k values, he tried to compare the number of binary constraint networks you can have and the number of arbitrary networks or CSPs that you can have and he showed that the number of arbitrary CSPs is much much larger than the number of binary constraint networks over this set of n variables and k values. And which is to say that if you are trying to define a CSP over this then not every CSP can be defined using a binary constraint network.

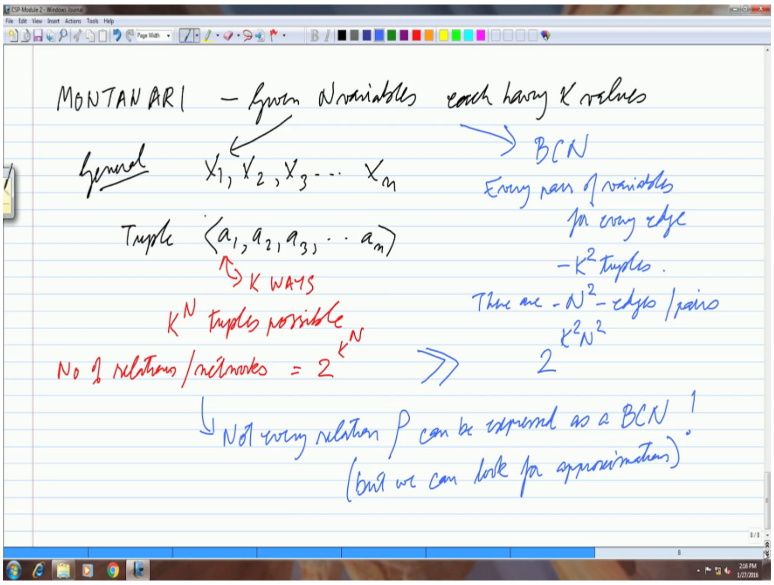
So the general thing was as follows. You have these three variables x1, x2, x3,…, xn and from this you choose a value. So one tuple would typically look like <a1, a2, a3,…, an> and each of these can be done in k ways. Each value we can choose from its domain in k ways because there are k values and therefore (k * k * k * k)^n. So there are total of k^n tuples possible.

A subset of this k^n tuples defines a relation. So the number of relations or networks or constraints is essentially any subset of this k^n tuples essentially and the number of subsets is two raised to this value. So the number of relations is 2^(k^n).

Now if you're talking about binary constraint networks then the important thing is that every relation is a pair essentially. So for every pair of variables or you can say on every edge, because the binary constraint network naturally can be represented as a constraint graph, there are k square tuples and then there are n square edges or pairs. So the total number of pairs that we can have is k^2n^2 and any subset of this is a binary constraint network. So you again have 2^((k^2)(n^2)) and what Montanari pointed out was this value is much much smaller than 2^(k^n).

Remember that we talked about relations as ρ. That's the solution relation. That's the relation we're trying to express as a constraint network. So not every relation ρ can be expressed as a binary constraint network but we can look for approximations. And the approximation that we are interested in is called a projection network.

(Refer Slide Time 14:40)



The idea of a projection network is as follows – given a relation ρ over some variables X, we can define a binary constraint network which is called P(ρ) where P stands for the projection network which is made up of a set of variables X which is the original set of variables, a set of domains D and a set of constraints, let's call them P because these are binary constraint networks.

Now here each Di = π_i(ρ). So basically what is the domain? Which are the values in the domain which participate in these constraints? And each Pi, or Pij I should say because it's a binary constraint network so every relation is a binary relation, is equal to πij. These are the constraints that are described by the binary constraint network.

So let me use an example. Let us say X = {x1, x2, x3} and let my relation which I'm writing as a table form is something like this – {<1,2,2>,<2,1,3>,<2,3,1>,<1,3,2>}. You can infer the domains of the variables. We'll do that in a moment.

You can see that for this relation Dx, let's call it D1 which stands for the domain of x1 is {1,2}. D2 = {1,2,3} and D3 {1,2,3}. So, these are the domains. And what are the relations?

So we want three relations here – x1 x2, x2 x3 and x1 x3. So R12 = {<1,1>,<1,2>,<2,1>,<2,3>,<1,3>}.

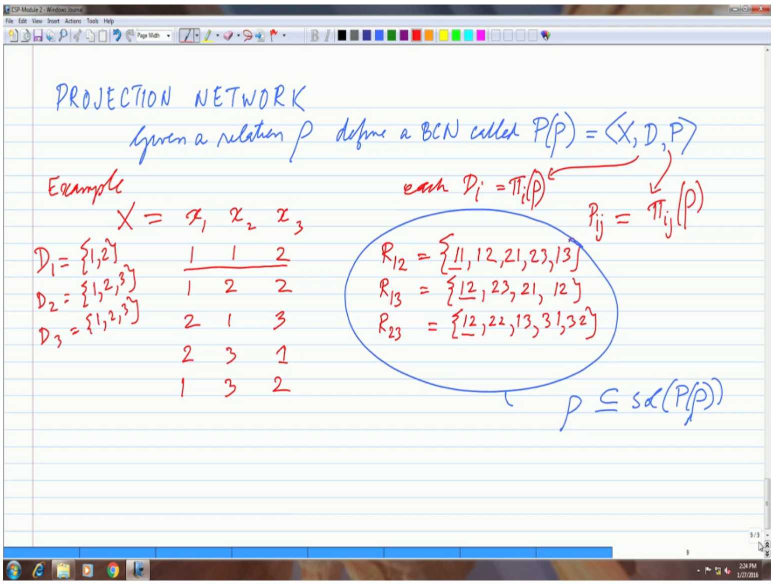
Maybe I should have chosen a shorter example. Anyway, R13 = {<1,2>,<2,3>,<2,1>,}. So I hope you've got how we are getting this. For R23 for example, we are simply taking the second column and the third column and removing duplicates from there. So that's a projection network essentially.

Now if we were to take the solutions of this projection network, then we will find that the original relation that we wanted to express is possibly a subset, it could be equal to but possibly a subset of the solution relation of ρ. So what, maybe you should treat this as a small exercise, what is the solution for this projection network that we have created? We have these three relations given to us and we want to find values.

For example you will see that of course these tuples will be part of the solution because by definition all these constraints are represented in the projection network. For example <1,1,2> is represented because we have <1,1> here and <1,2> here and <1,2> in the third. So, in fact the first set of tuples in the three relations come from the first relation.

So likewise, you will see that every row will be represented but there are extra things which are not in the original solution which will also be represented. So if you were to extend this you will see that you will get <1,1>. This <1,1> comes from relation R12. And then if you look at thw third row it's got <1,3>. Now that's a wrong example here. So <1,1,3> will not be present but I have to now look for a tuple which is present here but not here. Now maybe this is an example where I think but I will leave it to you as an exercise, this is an example where the solution is equal to the original relation but let me show you another example where that is not the case.

(Refer Slide Time : 21:34)



If X = {x1, x2, x3} and you have {<1,1,2>, <1,2,2>, <2,1,3>, <2,2,2>}. So there are four rows in this relation and if you do the projection network for this, you will get R12 = {<1,1>, <1,2>, <2,1>, <2,2>} because all four combinations are here. R13 = {<1,2>, <2,3>, <2,2>}. R23 = {<1,2>, <2,2>, <1,3>}.

Now if you want to solve this then the solution will add a new row to this and the new row is <2,1,2>. So this is the solution relation of the projection network of this ρ essentially. So where does this <2,1,2> come from? This <2,1,2> comes from the fact that <2,1> is present in between x1 and x2. <1,2> is present between x2 and x3. <1,2> is here and <2,2> is present in between x1 and x3. So this last row of my relation, <2,1,2> is added to the solution of the given network. So what is this network? This is the binary projection network of ρ and what is ρ? ρ was the original network which was in red. The four relations. That is what I'm trying to express as a binary constraint network.

So when I created the projection network using those four rows, I ended up with the binary constraint network which is R12, R13, R23. When I solve that binary constraint network, its set of solutions contains a fifth element which is this <2,1,2>. Why does it contain it? Because it, as we have marked in the blue lines, every one of these three constraints is satisfied by this instantiation and therefore it's a solution. So what we have shown here is an example of the case where ρ is a strict subset of the projection network essentially.

So if we take any arbitrary n-ary relation, in this case n is three, and convert it into a BCN and then solve the BCN, you may get a relation which is a superset of the original relation which is expressed by this relation here. But it has been shown that the projection network is the tightest, or it's the minimal equivalent relation or network. Which means you can't get a network which is tighter than that which says that there does not a network R' such that – so remember I'm saying there does not – such that ρ is a subset of the solutions of R' and R' is a subset of the projection network or the solutions of the projection network of the original ρ essentially.

So there is no network which is tighter than the projection network which contains the original relation. So it's an approximation or an upper bound on the original relation but it's the smallest upper bound. So that's been shown. We will not go into the proofs here but we'll accept it as that.

For a binary constraint network, it turns out the projection network is minimal which leads us to this thing which say that it's a binary constraint network, the minimal network which expresses the solution is equal to the projection network of rho. So it's a tightest upper bound network that you can construct using a binary constraint network. This basically gives us an example of how you can approximate any arbitrary relation with a binary constraint network. We will use projection networks as we go along.

Now we'll stop with the basic definitions here and move on towards solving CSPs and as I have mentioned earlier, the basic idea of solving a CSP is search based, that we try different values for the different variables in such a manner that they respect the constraints that you are talking about and if necessary you backtrack. But we have also mentioned and as we looked at some examples, the cryptarithmetic puzzle or the Huffman Clowe's labelling, you can do propagation.

We want to work towards this idea of combining propagation and search and first we will look at the notion of how to construct tighter networks because tighter networks will have the advantage that the amount of search that you have to do is less. We will sort of expand upon this idea as we go along but we'll look at the simplest notion of consistency which is called arc consistency in the next class.

(Refer Slide Time : 29:10)

CSP-Module 2 - Windows Journal

Example 2

$X = x_1 \; x_2 \; x_3$

ρ:

x_1	x_2	x_3
1	1	2
1	2	2
2	1	3
2	2	2
2	1	2

$\rho \subset Sol(P(\rho))$

$R_{12} = \{11, 12, 21, 22\}$

$R_{13} = \{12, 23, 22\}$

$R_{23} = \{12, 22, 13\}$

← PROJECTION NETWORK of ρ

↓

is the MINIMAL equivalent network.

↓

There does not exist a network R' s.t. $\rho \subseteq sol(R') \subseteq sol(P(\rho))$

Solve

For a BCN the projection network is MINIMAL

$$M(\rho) = P(\rho)$$

10 / 10

2:31 PM 1/27/2016