Constraint Satisfaction Problems

Prof. Deepak Khemani

Department of Computer Science

Indian Institute of Technology – Madras

Module - 2, Lecture – 02

Let's continue looking at constraint networks. In the last class we talked about the fact that networks express solutions. What when two different networks express the same solution? Then we call them equivalent. So we say that R and R' express the same solution relation ρ. x is the set of variables over which the relation is defined. So if two networks have the same set of solutions which we call as expressing the solutions, then we say that the two networks are equivalent.

Let's look at a couple of examples. We'll look at map colouring. You remember the notion of a constraint graph. A constraint graph is a graph in which the variables are the nodes and there is an edge between two nodes if the two variables participate in some constraint. The constraint doesn't have to be a binary constraint. It can be a higher order constraint as we saw in the crossword puzzle example but if they participate in a constraint then we draw an edge between them. In a map colouring problem it is straight forward because the map colouring problem is naturally expressed as a binary constraint satisfaction problem because there are two neighbouring countries which are kind of related or regions which are kind of related.

We can represent a constraint network by the constraint graph. So here is one constraint graph. There are three variables x, y and z and each of them can take one of the two colours red or blue. r stands for red and b stands for blue. So it's a two colouring problem and the relation is that you cannot assign the same colours to them. So it's basically the not equal to relation.

So just as a quick recap, how would you express the relation? Here you would express it by saying Rxy which is the relation between x and y is {<r,b>, <b,r>}. And this is also Ryz. Basically if you give r to one variable then you must give b to the other variable and you can assign them in any order.

What is the solution to this problem? There are two solutions as you can see – r, b, r for x, y and z respectively in this order or b, r, b. So this is a constraint network. Let's call this

network R and it has two binary constraints between x y and y z and it has a set of solutions that is available to you.

Now look at this other network with the same set of variables and the same domains but it is a little bit different. There is a not equal to relation between y and z and an equal to relation between x and z. You will see that if you were to solve this, it has the same set of solutions. So you can give a value of r to let's say variable x and it forces you to give a value of r to the variable z because there is an equality relation here as opposed to the inequality relation and then you can choose the other value for the variable y. So this has the same solution as the first network. Both the networks have the same solution so they are both equivalent.

Now one observation that you should make is that if you look at the first network, we don't have an edge between x and z. What does that mean if there is no edge? No edge basically means a universal relation which means we are not bothered to specify which two combinations are allowed. So here for example there is an edge which is missing and because we have not specified in our CSP. We have said there are only two relations in the second network, there is a relation between x and z and there is a relation between y and z. We have not said anything about the relation between x and y which means in the statement of the problem it's a universal relation. You must treat it like a universal relation.

So the first thing is that we can have different networks which are equivalent which means they express the same solution. Then secondly we can make inferences at some point. So which graph is better? The one which has more constraints or the one which has less constraints? That's an issue that we will look at as we go along but imagine a simple search algorithm which looks at variables in a given order. For example, it looks at variables in the order x and y and z. Look at the second network here at the bottom of the diagram. You can give a value to x. So some search algorithm will say x = r. Now because there is nothing specified between x and y you can choose any value for y. So you can say for example x = r and y = r. Nothing stops you from doing that. Nothing stops you meaning we don't have a constraint between x and y.

So we have in effect a universal relation which means all combinations are allowed so you can choose any combination. But you can see that that is not a good thing because once you choose x = r and y = r, you cannot choose a value for z because z must be equal to x and at the same time it must be different from y but by the time we come to z we have already made the mistake of giving y the value of r. Whereas if we had the constraint between x and y then

we would not have made the mistake because we would have allowed to give only consistent values to y.

So we can infer such missing constraints and the inference can be done by a process of composition. So those of you who have not looked at sets and relations and functions and that kind of basic discrete maths, I would suggest that you go and revise that because we use it once in a while. Also it helps if you are familiar with the notation of relational algebra which you must have studied if you have done a database course. So we'll use alternate notions at different times and we will keep using them so you must kind of revise those things.

So we can infer a new relation. So let's look at the first example here where we have a relation between x y and y z but we don't have a relation between x z. So we can infer a relation Rxz. It is a set of pairs <a,b> such that a $\in$ domain of x and b $\in$ domain of z and $\exists$ a c where c $\in$ domain of a third variable y, Dy, and the pair <a,c> $\in$ relation Rxy and the pair <c,b> $\in$ relation Ryz. This is a generic definition. It just so happens that x, y and z are the variables that I've used here. So you can use it as an example but in general you can infer any relation between x z.

So if you have two relations Rxy and Ryz available to you as in the example on the top left then you can infer a third relation by simply doing the composition of those two relations. We say that an element a in the domain of x is related to an element b in the domain of z if there exists some element c which exists in the domain of y and <a,c> belongs to relation Rxy and <c,b> belongs to relation Ryz. In other words, if you're looking at it as a graph then if there is a path from x to y and from y to z then you can find a path from x to z.

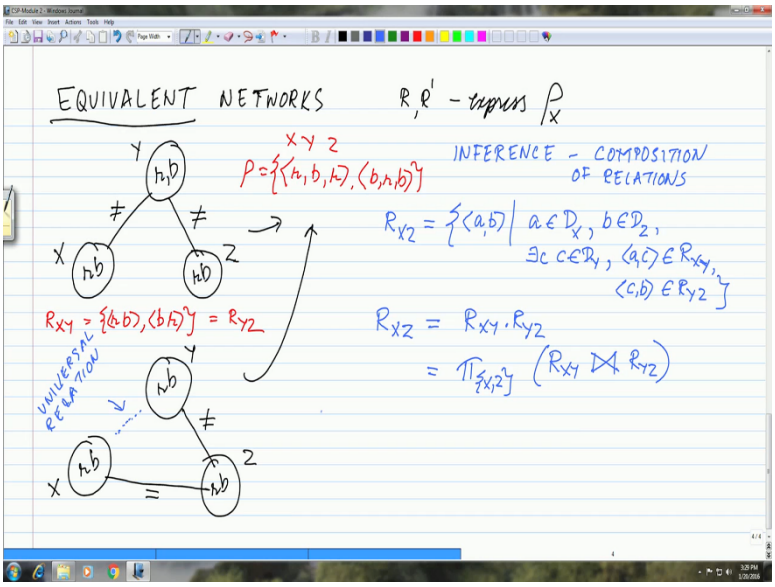
You can also express this in the relational algebra terms. So we can say Rxz is equal to the composition of Rxy and Ryz. π stands for projection, over the variables x and z. You basically take only those values which correspond to the variables that you are talking about. In this case you are talking of variables x and z and you're projecting the join between Rxy and Ryz. This is a natural join.

So joining the two relations Rxy and Ryz will basically give you all triples which are related to each other and from those triples you can extract or project the pair between x and z. So we have inferred a new relation essentially.

For the above example, x can take the value r and y can take the value b. And since y can take the value b, z can take the value r then we can say that whenever x can take the value of r, z

can take the value of r. So this relation that is missing in the top diagram can be inferred. So we can infer new relations essentially.

(Refer Slide Time 13:32)



We just saw the notion of equivalent networks. We have a notion of tighter than. We want to work towards the notion of a minimal network. And by minimal we'll also mean tightest. Let's define what that means.

We say that a network R, so you must get used to this idea; whenever you say network we are talking about a constraint satisfaction problem, we are just using this notation; is tighter than a network R' if for each relation Ri, Ri $\subseteq$ Ri' essentially.

So let me take the same example as above. We have x, y and z and we do the composition operation to infer a new relation which gives us a new relation. Now we have three relations here, two are not equal to as before, one is equal to, and everything else is same. So there are two networks. They are over the same domains and same variables. One of them has two relations, Rxy and Ryz. The other one has an inferred relation – Rxy and Ryz and also Ryz.

Now you can see that these two networks are equivalent to each other in the sense that they express the same set of solutions. We will call the first network R' and the second network R, then we can see that R is tighter than R'. What does this mean? This means that Rxy is a subset of Rxy', Ryz is a subset of Ryz' and Rxz is a subset of Rxz'.

So you must remember that Rxz' is a universal relation, which means you are allowed all combinations. You're allowed <r,r>, <r,b>, <b,r> and <b,b>. Whereas Rxz, the relation that we inferred is that they must have the same values – <r,r> or <b,b>. So we have this notion of one network being tighter than another network. This is for each relation Ri. However, it's not necessary that given two relations, one of them is tighter than the other.

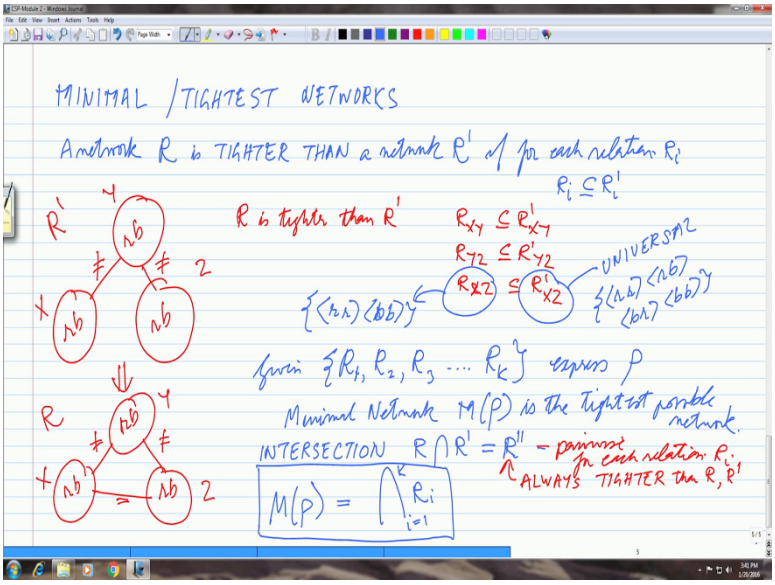
Supposing you are given a set of k networks R1, R2, R3, … , Rk and let all of them express a relation ρ, which means they are all equivalent. The set of solutions for each of these networks is ρ . Then we say the minimal network M(ρ) is the tightest possible network. How do we get this?

We use the intersection operation. R ∩ R' = R''. This is done pair-wise for each relation. So if you have two networks and if you do a pair wise intersection of all the relations you get a new network which is called the intersection of R and R' which is in this case R'. The result is always tighter. It is tighter than both R and R'. So if you take an intersection of any two networks you always have a tighter network.

And then we can see that the minimal network M of rho can simply be taken as an intersection over i going from one to in this example, of Ri. So the intersection relation, the intersection of networks or the minimal network is a network which has the fewest number of tuples inside it.

In this example on the top where there were two relations Rxy and Ryz, implicitly there was a tuple xz but it said that all four combinations were allowed. In the bottom network which has three relations the relation Rxz has only two tuples where as in the top network implicitly it had four tuples.

(Refer Slide Time 21:22)



We'll come to another way of arriving at the notion of minimal networks for binary constraint networks but before that I want to raise a question. Let there be n variables, each of domain size k. The question that I want to ask is, given a set of variables can we express every relation as a BCN?

We said at some point that binary constraint networks are easier to handle and lot of algorithms have been developed. The question that we want to ask is that can you express some given relation ρ as a binary constraint network?

It's basically a counting argument. And the answer to this is no. The answer was given by Montanari in 1974.

If we have n variables and each with k values you can choose the first value in k ways. You can choose a value for the second variable in k ways. So x1 has k ways of choosing a value. x2 has also k ways of choosing a value. And therefore there are k^n ways of choosing an n tuple or k^n combinations. Now which of them you allow and which of them you don't allow determines the relation that you are talking about. The power set of this set is the number of relations you have which is 2^{kn} (2 to the power k to the power n).

Whereas if we talk about binary constraint networks, each relation can get values in k^2 number of ways because there are two variables. You can choose the value for the first variable in k ways and the value for the second variable in k ways. So you can choose them in k^2 ways. And there are n^2 relations or n^2 pairs of variables.

In the first example there is only one set of variables. All n variables you have to count at the same time. In the case of binary constraint networks there are n^2 pairs of variables. And each

pair of variable can get a value in k^2 ways. So the number of relations or the number of distinct binary constraint networks is 2^{k2n2} essentially. And you can see that this is much smaller. 2^{kn} (2 to the power k to the power n) >> 2^{k2n2}.

So if we are given n variables, each with k possible values, and you're interested in some relation ρ on those n variables then very few of the possible relations that you want to express can be expressed using binary constraint networks. So the number of relations that you can choose from is much much larger than the number of binary constraint networks that you can construct.

So that's of course a kind of a hindrance that you can't solve all problems using binary constraint networks but we will try to look at this from a different perspective and we say that can we get an approximation of a relation using a binary constraint network and that will lead us to the idea of projection networks which we will look at in the next class and then we will proceed from there. So you must be aware that binary constraint networks are nice to work with but you can't express every relation using binary constraint networks.

Now you might remember we looked at the crossword puzzle where there were 13 letters that you had to fill in the example that we saw in one of the earlier classes and there were relations between the letters. So the first letter, first word for example had five letters and the relation was between x1, x2, x3, x4, x5 where for example you can use the word sheep or any five letter word. And then we had a constraint network which was not binary because relations were between the number of letters making up a word.

Then we saw that corresponding to every such problem, there is a dual network or there is a dual constraint graph which is made up by taking the variables as the scopes of the original constraint graph and the relations if they shared a variable. We saw in the case of the crossword puzzle that we can always have a dual and the dual was always binary. So is there a contradiction here? We said that a crossword puzzle can be represented like a general network but the same crossword puzzle can also be represented as a binary constraint network.

If you think carefully about this, it is not a contradiction because the number of variables is different. In the first or the primary constraint graph that we constructed there were 13 variables and each had domain sizes of 26 which means you could put any of those 26 letters. Whereas in the second formulation, there were only 6 variables because the crossword allowed 6 words to be fit in and for each word there was a variable and then the domain sizes

were different because the number of words that your lexicon allows would determine the domain size. So even though the problem was same, the formulation was different. One had 13 variables, the other had 6 variables and the domain sizes were also different.

So the fact that you could have a binary constraint network for every crossword puzzle doesn't mean that Montanari was wrong. Montanari is saying that if you have n variables and each is of size k then there is only a small subset of relations that you can express using binary constraint networks but for other relations we have the notion of approximate networks or projection networks.

(Refer Slide Time 27:53)