Artificial Intelligence: Constraint Satisfaction Problems
Module 1: Introduction
Lecture 1: Constraint Satisfaction Problems (CSP)
Professor: Deepak Khemani
Department of Computer Science and Engineering, IIT Madras

Keywords: constraint satisfaction problem, variable, domain, constraint, scope, relation, binary CSP

So welcome to this course on constraint satisfaction problems. This is third in a series of courses that we have done in Artificial Intelligence. And we also often refer to this sometimes as Constraint Processing. Okay, at the very outset let me start off by giving you the references, the books for this course. So, there are 2 books essentially, one is a book by this name constraint processing by Rina Dechter. I have a copy of this book here and I think we have a copy in the library. And from this, we will do chapters 1 to 6 and maybe chapter 12, and maybe one or two more essentially. So that's one textbook and the second book that we will use is my book which is a first course in AI, from which we will use chapter 9, only one chapter, but it basically covers most of the stuff that I want to cover in this course so that's my book here, and we will use these two. By and large, most of the material we cover will be from these two books essentially. Okay so, when we talk of artificial intelligence, essentially we talk about problem solving, right. And we have already done 2 courses on this topic they are available online on the NPTEL site. The first course was problem solving using search and essentially it's a method which uses search techniques to find a solution essentially. And the second course was knowledge representation and reasoning, and here the focus was on reasoning and in one course the focus was on search essentially. But in general, when we talk about problem solving in AI we are interested in general-purpose methods, or methods which can be applied to many different problems and so on. And from that point of view, constraint satisfaction problems is a very good way of formulating problems because it is a very uniform way of doing things and once you have formulated a problem as a CSP problem, so we will use the term CSP. Once you have formulated a problem as a CSP problem, then you can just take a standard CSP solver and use that to solve the problem essentially. And the nice thing about CSP is that it combines both these approaches, or it allows you to combine both these approaches, so in a sense that you can do search to find a solution and we will shortly define what do we mean by a CSP problem, or you can even do reasoning of the kind we did in, for example, logical reasoning, making inferences, and saying that if this is true and this is true then this is true. Both kinds of things can be nicely combined in one formulation which is the CSP formulation.

So let me try to illustrate that. So, henceforth we will refer to this as CSP. Now the basic idea of formulation here is that as a set of variables. That's a very high level way of looking at things, that when you talk about constraint satisfaction problems, you are essentially talking about at a set of variables essentially, and which can take values from their own domains. Okay so basically we are saying that each variable will have its own domain and you can get a value from its domain essentially. So that's a very high level way of looking at things essentially. What is interesting here is that we have this notion of constraints. So a constraint is a restriction on the combinations or on some combinations combinations of values that the variables can, that the variables can take. We'll formalize this notion shortly as we go along essentially, but this is really the basic idea behind constraint satisfaction problems is that you formulate the problem as a set of variables, each variable can take values from some domain of its own and a constraint or a set of constraints. Each constraint is a restriction on the combination of values that the variables can take, and when you solve CSPs, it basically means assign values to all variables such that all constraints are respected. The term which we use is satisfied, but you can say respected, essentially. So you can see that the problem formulation simply says that you have a set of variable and each variable can take some values from its own domain and there are set of constraints on combinations of values and as long as you can assign values to all variables which satisfy all the constraints then you have a solution.

We are not talking about what is the methodology for finding the solution. Now clearly there are different ways of doing this. One way would be to use search, just try all combinations and just keep testing all constraints. That would be basically a search based approach. But fortunately, this formulation allows us to do also reasoning essentially. So that's a nice thing. You can combine search and reasoning. I would like to illustrate that with a example. But you must keep this in mind, that basically this is what a CSP is, a set of variables, and a set of constraints and then you have to find values for the variables from their constraints.

So let's look at an example which is from the domain of numbers. And I want to essentially illustrate the fact that you can combine reasoning with search essentially. So cryptarithmetic puzzles are a class of puzzles where you are given some mathematic equation, but you are not given the numbers, you are instead given placeholders for those numbers. And they kind of look interesting, so they are interesting and you what you have to do is to find a solution to that. So examples of this, for example, you can say that so each of these letters, so domains for all variables is the digits digits upto 9 essentially. So which means for each of these letters, each of them is a variable. You can, you need to find a value for, from its domain and we are assuming in this problem that all domains are the same which is the letters from 0 to 1, 0 to 9. And then you can say I want to add these numbers, and answers should be, Okay, so essentially we are giving us a addition problem, but we are not telling you what the numbers are and all we are saying is that each letter is a variable, values must be distinct. That's an additional constraint which I'm not stating explicitly here that each each variable must get a different value. You can't get the same value. Because if you are allowed to give the same values then all you would do is to give zero to everything and then you would have a sum. So what do you mean by saying you would have a sum? There are constraints which I'm not writing here, but these are basically says that if these letters were replaced by the numbers, then the sum would be arithmetically correct essentially. So whatever I plug in for S and E and N and D and so on, what I will get is a valid sum, summation, addition problem essentially. So basically the task is to find values for that essentially. So how do we solve this essentially? Obviously, we can see solution for this. One is search, and you can do a brute force search essentially, you can say S is equal to 1 then 2 then 3 and so on. And then E is equal to one then two then three. I'm just writing it in a cryptic manner, when you say S is equal to one then E cannot be 1 and so on, and all those things are there. And you can just try all combinations and see if actually the sum works out or not essentially. The other approach is by reasoning. Okay so what is the kind of reasoning we can do? We can say that, for example, this M must be 1. Why? Because you know that's the only way you can get this value. Of course there are some hidden variables which I have not spoken about but let me take another example and we can try to illustrate that completely essentially. So another example is this, 6 plus, these are interesting because of the fact that the English reading of this problem also is quite meaningful essentially. So when you say send plus, send more money, it's like a meaningful sentence. Or if you say 6 plus 7 plus 7 is 20, then it is a meaningful sentence. So as I was saying, what you need here is some hidden variable. So there must be something here which we call as carry variables and they may take values of between 0 and 1. But we are not worried about that. Of course they will take values like 0, 1 or 2 also in this case. But we are not really concerned about that. We want values for these letters, six, s i x s e v e n, and so on, so that the sum is correct. That's the basic constraint which I am not expressing. So let me take one more example, and we'll try to solve this as we go along. So, this is E. So let's create placeholders for these where the values will come in. So we want values for each of these. And we are also allowed to have carry for these. So let's try to solve this online by using a reasoning approach and we try to fill in whatever numbers we can somehow argue. So just as we said that M must be one, here we can say that A must be 1 because that's the only way when u add T plus P, you're getting a carry which is coming as a letter A. And so therefore the only carry that you can get by adding two integers is 1, so the maximum we can do is 8 plus 9 is 17 and you can't go beyond 17. So that carry will be always 1. So we know that A is 1. And once we know that A is 1 and we know that this carry is 1, and then we can fill up a value for A 1 here also. Because whenever there is a A, it must be the same value essentially. Okay,

now the fact that we added something to T and got a P means that there must have been a carry above that as well. And then what can P be? P can only be 0 essentially okay. Because you're adding 1 to something and you're generating a carry. So you can only add 1 to 9 to make it a two digit number. If you add 1 to 8 you will get 9, and you won't get a carry so you won't get this thing essentially. So we can figure out that P must be 0, so we fill in 0 here, we fill in 0 here, and consequently T must be 9. So now we know that 1 plus 9 would have got us 10 and then this 1 would have got carry and that is the A there. So once we know that T is 9, we can fill 9 here 9, 9 here. And then we must know that E must be 8 as 9 plus 9 is 18, and this must be a carry here, which is 1. And then we can see that 1 plus 1 plus 1, so this must be 3, L, and E is 8 here and 3 does not generate a carry, so this carry must be 0. And then we can figure out that the only value that H can take is 2 because that will generate a carry. So this particular problem we managed to solve. So what is this problem? This problem says that 819 plus 9219 is 10038. So this is a solution. And the problem was eat that apple essentially. Now in this example what I try to illustrate is the fact that you can use reasoning to arrive at the solution. We didn't try out all combinations for values for the different letters. We can do that also and that is another approach to solving this problem essentially. Not only that, in this particular example we could solve the whole problem using reasoning. If we had got stuck on the way, or somehow if we could not figure out what the next thing is, then we could resort to search as well essentially. So you can do reasoning part of the way, then search, then so on and so forth essentially. You can see that there are many problems like this. So another example is Sudoku. So typically one expects to do more reasoning on sudoku. And that's the whole exercise of doing the thing, otherwise if you just give it to a small computer program, it will solve it within half a second for you essentially. So the whole thing is that you can combine these two things essentially. So what I want to do is to give you a few more examples to give you a flavor of the fact that you can model different kinds of things as constraint satisfaction problems, and then after that we will kind of move over to solving CSPs essentially, algorithms for solving CSPs. And algorithms for solving CSPs will be a combination of these kinds of things that I'm saying, search and reasoning. In the language of constraint satisfaction we don't say reasoning so much, we say propagation essentially. So there is this whole field called constraint propagation, that you can propagate restrictions to other variables and in that process we can do that essentially. So here for example, in this eat that apple example, we started with a restriction on A, the leftmost A in the answer. And then we propagated that, that we said that because A is 1, T must be 9, and E must be 0 and so on and so forth essentially. So this kind of things we will do essentially okay.

So a CSP, a CSP problem is basically defined as a triple $<X, D, C>$, where X is a set of variables, so let's say x_1, x_2, upto x_n. And D is domains for those same variables. So there must be d domains, sorry n domains, and C is a set of constraints. So let's say we have some k constraints where each c_i is made up of two things called S_i and R_i. So this is basically the formalization of this problem . We will come back to the examples soon. Where S_i is the scope of the i^{th} constraint and it's basically a subset of X. So some variables on which the constraint is defined and R_i is a relation which is a subset of $X_{1i} \times X_{2i} \times X_{si}$, if you can make sense of this. So essentially, I should say modulus X_{si} there. So if the scope has, let's say P variables, so if S_i is equal to P, then we are saying that R_i is a subset of cross product of X_{1i}, X_{2i}, X_{Pi} essentially. Because there are P variables, so we take the cross product of that and we have the scope of that. So essentially this is a formulation of a constraint satisfaction problem. Where the scope of a constraint tells you how many variables are participating in that constraint and the relation of the constraint, the actual constraint, is the relation on the variables which are participating in that constraint essentially. Now, we want to look at some restriction of this very generalized formulation of a CSP, or what is the kind of problems that we will address. But before we come to that let me define a special class of CSPs which is called a binary CSP or BCSP. And in binary CSP the basic thing is that scope is less than or equal to 2, for every variable scope is less than or equal to 2 essentially. Which means that upto 2 variables participate in a constraint. So very often we will refer to, in binary CSPs, we will refer to scopes as follows. So for example we can say S_{12} or S_{34}, so what this means is that it is a short form x_3 and x_4

belongs to S_{34}. So this indices of variables we will use directly. Or we can, yeah, so something like this essentially. And the class of binary constraint satisfaction problems is of particular interest because there is a large number of, large set of algorithms that have been developed for that essentially. So I will stop with this lecture and in the next lecture we will start looking at some more examples of CSP and we will try to restrict the kind of CSPs that we will address in this course essentially.