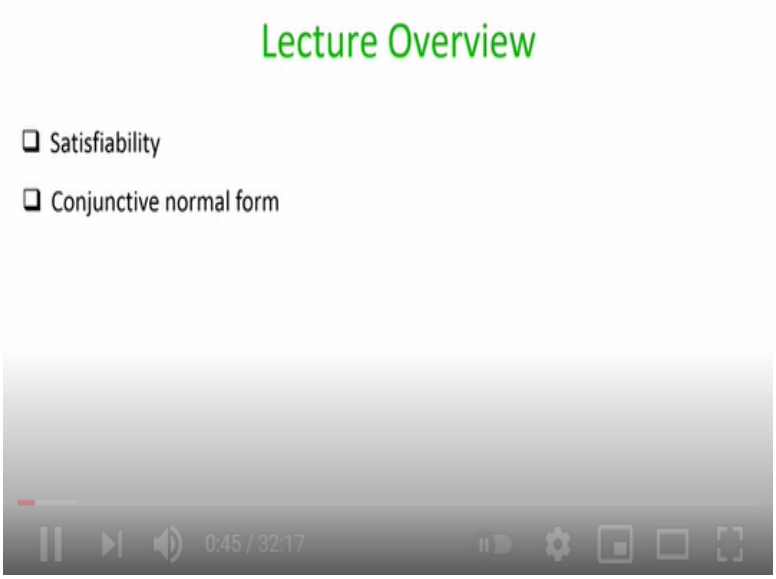
**Discrete Mathematics**
**Prof. Ashish Choudhury**
**Department of Mathematics and Statistics**
**International Institute of Information Technology, Bangalore**

**Lecture -03**
**SAT Problem**

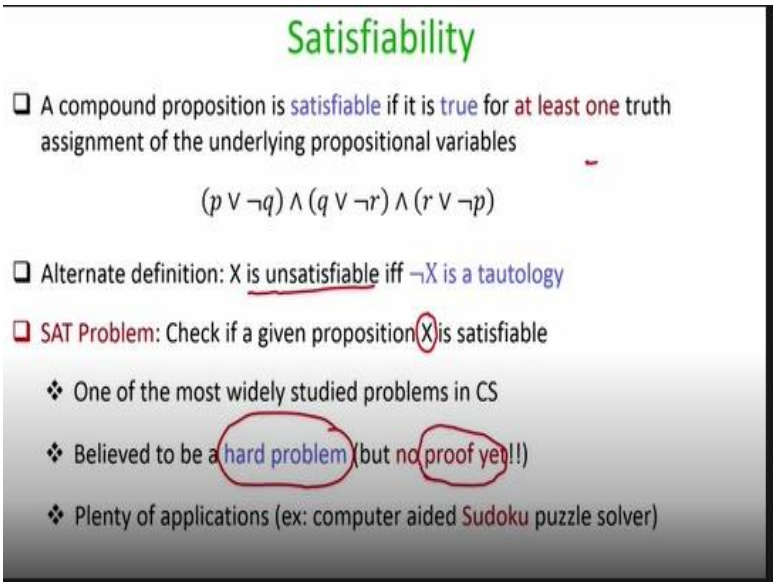
Hello everyone. Welcome to this lecture on SAT Problem.

**(Refer Slide Time: 00:24)**




Just to recall in the last lecture. We discussed tautology, contingency, contradiction, we discussed about logical identities, logical equivalent statements. In the plan for this lecture is as follows. In this lecture, we will introduce the satisfiability problem which is also called as the SAT problem and we will discuss about the conjunctive normal forms.

**(Refer Slide Time: 00:50)**



So what is the satisfiability problem? So first let us first define, what do we mean by a satisfiable proposition? So imagine you are given a compound proposition X. So this is an example of compound proposition X. The compound proposition X will be called satisfiable if it is true for at least one truth assignment of the underlying propositional variable. So in this case, this is my expression X and it turns out that if I make p to be true and r to the true and q to be true then the overall expression X becomes true.

Because for this truth assignment this p disjunction ¬q becomes true. The disjunction of q ¬r becomes true, the disjunction of r ¬p also becomes true and conjunction of true, true, true is overall true. So, I have one truth assignment for p, q and r for which the statement X becomes true. Of course there might other possible assignments of p, q and r as well for which it might be true.

The question here the definition say that even if you have one true assignment for which X becomes true, my statement X will be called as satisfied. An alternate definition here for satisfiability is the following. I will say that my compound proposition X is unsatisfiable. I stress unsatisfiable if and only if negation of X is the tautology. And why so? Because if X is unsatisfiable that means it is always false, right?

Because satisfiable means at least one truth assignment is there for which X will be true, but unsatisfiable it means it is always false or in other words it is a contradiction and if it is contradiction then you take the negation of that it will be always true. That is why this alternate definition is correct. Now the SAT problem is the following. You will be given a compound proposition X here and you have to verify whether the proposition X is satisfiable or not.

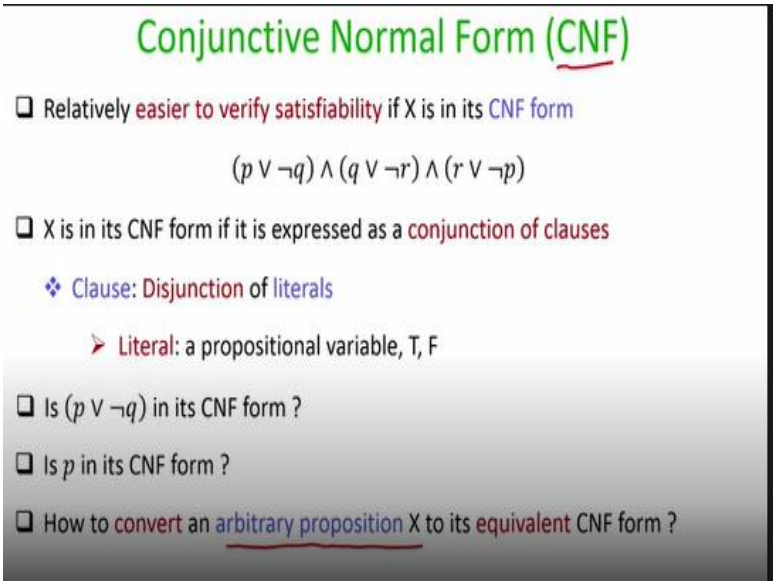
That means you have to give me a yes or no answer, yes means yes there exists a true assignment for which the proposition X is true, no means ok there is no truth assignment for which X is true. And this is one of the widely studied problems in computer science. There are plenty of applications of this SAT problem we will see one such application in this lecture and it is believed to be a hard problem, what do I mean by hard problem?

That means by hard problem informally I mean here that we do not have efficient algorithms practical, algorithms to verify whether the given proposition X is satisfiable or not. What do I

mean by practical? By practical here roughly I mean that it gives you the answer very quickly, and of course we do not have any proof yet that this is indeed a hard problem. It says that we do not have any efficient algorithm as of now to verify whether a given arbitrary proposition is satisfiable or not.

But we strongly believe that it is indeed a very difficult problem to solve and as I said there are plenty of applications, we will see one of the applications namely the computer aided Sudoku puzzle solver.

**(Refer Slide Time: 04:44)**




Before that let me introduce what we call as Conjunctive Normal Form or CNF form for a compound proposition. The motivation for studying the Conjunctive Normal form is that if you are compound proposition X is given in its CNF form then it is relatively easier to verify whether X is satisfiable or not, so what do I mean by a conjunctive normal form? In some books, they also called this as product of sum.

So here is an example of an expression which is in conjunction normal form. You have several parentheses here inside parenthesis, you have disjunctions of variables and each parenthesis is connected by this conjunction operator. That is why it is called product of sum, outside you have the products, you have products of various brackets here and each bracket you have the sums.

So now let us formally define what do we mean by a conjunctive normal form? So an expression X is said to be in its conjunctive normal form if it is expressed as a conjunction of

clauses, so you can see here you have the conjunctions appearing here. Each such parenthesis here is called a clause. Now, the question is, what is the clause? Well, it turns out that the clause is a disjunction of literals, because informally in each clause, inside each clause you have disjunctions of various variables.

Those variables might be either without the negation operator or with negation operator. So that is why we define a clause to be disjunction of literals. Now the next question is what is the literal? And my definition of literal here is a literal is either a propositional variable or the constants true or false. So here T stands for true F stands for false, and these are my constants.

So my definition of literal here is it can be either a propositional variable or it can be one of these two constants true or false. So let us verify whether this expression X which we have written here is indeed in its conjunction normal form as per this definition that we have given here, that is what we want to check quickly. This is my X here, so the definition says that X should be a conjunction of clauses.

So we have conjunctions appearing here, we have to verify whether indeed we have the conjunction of clauses appearing here or not. So we have to verify whether each of the expressions within the parenthesis is a clause or not. What is the definition of clause? It says it should be a disjunction of literals. So let us check whether the first sub expression, namely the disjunction of p and ¬q is a clause or not.

So p is a literal and negation q is also a literal remember because what is the definition of a literal? A literal is a propositional variable. So negation of q is also a propositional variable. Do not think that propositional variable just means the positive form of that variable. So if q is a propositional variable negation of q is another propositional variable. I can represent it by r. So you can imagine that I have the disjunction of p and r appearing here and that is why it is a literal.

And in the same way the second sub-expression namely the disjunction of q and ¬r. It is also a clause so, this is the propositional variable and negation of r is another propositional variable. In the same way the third sub expression the disjunction of r and negation p is also a

clause. So I have three clauses here and we have conjunctions of these three clauses and hence this expression X is in its conjunctive normal form, it satisfies my definition.

So now let us answer this question, is this expression the disjunction of p and negation q is in its Conjunctive normal form? So you might be wondering well there is only one clause here and as the definition says that the expression is in its CNF form only if it is a conjunction of clauses. So we do not have any second clause here we do not have any conjunction appearing outside this expression.

And hence it might look that this expression is not in its conjunctive normal form, but that is not a right argument. Indeed this expression is in its conjunctive normal form. Because if you see closely this expression can be, logically equivalent to this expression. You can verify that and I can put a parenthesis around this T and I can further say that this expression (T) is logically equivalent to (T disjunction T)

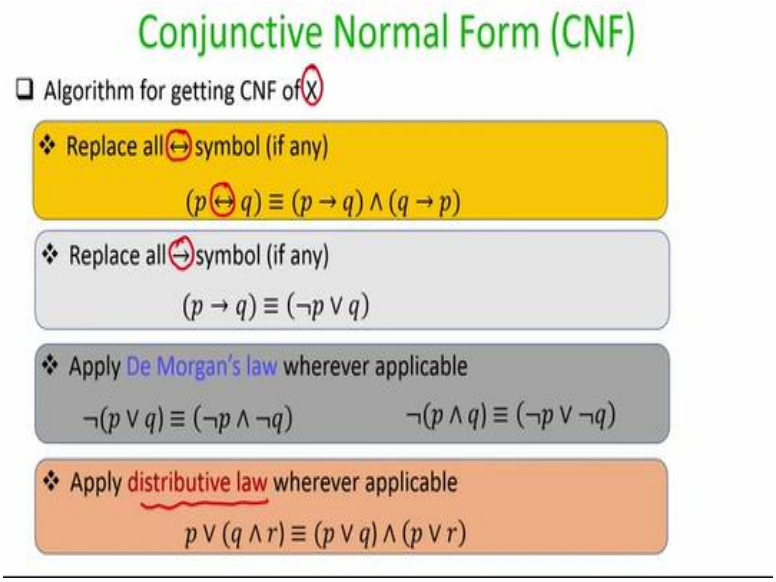
Now this is a Clause this first sub expression and this second sub expression is also a clause. I have conjunction appearing between these two clauses. So even though there is no explicit second clause here. I can always interpret that I have this constant true which is appearing in conjunction with this expression and hence this expression the disjunction of p and negation q is also in its conjunctive normal form.

Now let us ask whether this expression p is in its conjunctive normal form? So remember the expression the compound p itself can be interpreted as a compound proposition. So again, it might look that there are no clauses here and there and there are no parentheses here. So that is why this is not in its conjunctive normal form. Again, that is an invalid argument. This expression is indeed in its conjunctive normal form.

Because I can always state p to be equivalent to the disjunction of p and the constant false and this is a clause and that is why the expression p is already in its conjunctive normal form. Now the question is, well if my expression X is not given in its conjunctive normal form can I convert it into another expression which is in conjunctive normal form and which is logically equivalent to my original expression X and answer is yes.

You give me any expression X. If it is already in its conjunctive normal form then well and good but if it is not and it is conjunctive normal form, then there is an algorithm which I can use and convert my expression X into another expression Y, where Y will be logically equivalent to X and Y will be in conjunctive normal form.

**(Refer Slide Time: 12:50)**




So let us see that algorithm. So the input here is some expression X which may or may not be in its conjunctive normal form. The first step here is the following. I will check the expression X and wherever there is an occurrence of bi-implication. I will substitute that bi-implication by using this logical identity. I can use the logical identity that the bi-implication of p and q is logically equivalent to the conjunction of p → q and q → p.

So by applying this law repeatedly I can get rid of all the occurrences of bi-implications in my expression X and the new expression that I will obtain will be logically equivalent to my original expression. Remember at each step of the substitution my goal will be to substitute something with a new thing that a new thing should be logically equivalent to the old thing, otherwise the whole process will give you an incorrect answer.

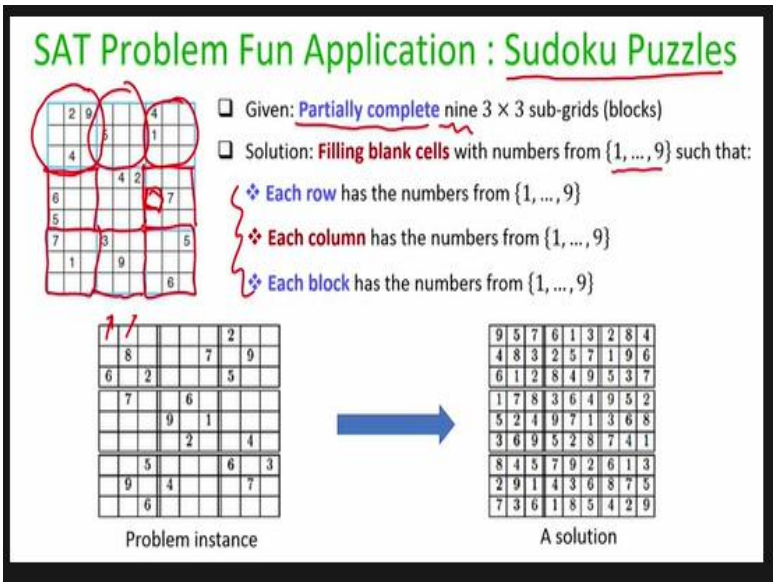
Next what I do is by applying the previous step in my expression, there will be no more occurrences of the bi-mplications, but what I might have done is I might have introduced some implications and there might be already some existing implications symbols in my expression. So I have to get rid of all of them and how do I get rid of all of them? By applying this logical identity.

So there is a well-known logical identity which says that p →q is logically equivalent to the disjunction of ¬p and q. So you apply this law repeatedly in your expression and get rid of all the implication symbols. Next what I do is I apply the De Morgan's law wherever applicable. That means wherever there is an expression of the form negation and inside I have a disjunction, what I will do is I will take the negation inside and split the negation over the disjunction and same way I do over the conjunction.

So wherever there is a scope of applying the De Morgan's law, apply it. And then finally you apply the distributive law wherever applicable; that means you distribute your disjunction over conjunction. Well, I am not going to distribute conjunctions over disjunctions because remember finally I have to bring my expression in the conjunctive normal form and conjunctive normal form means I should have this conjunction of clauses.

So that is why I apply the distributive law where the disjunction is distributed over conjunction. If I apply the other distributive law then I may not get the expression in the conjunctive normal form and now if you apply these four steps there is always the guarantee that you get a logical equivalent expression which is in its Conjunctive normal form.

**(Refer Slide Time: 15:56)**




Now let us see some fun application of the SAT problem, of course as I said there are plenty of advanced applications of the SAT problem. The SAT problem is encountered very frequently in AI applications. But I am going to demonstrate application the SAT problem with this fun application, which is solving the Sudoku Puzzle. So what is a Sudoku puzzle? So you will be given a partially complete nine 3 X 3 sub-grids.

So what do I mean by sub-grids here? So this is sub grid number 1, then this is sub grid number 2, sub grid number 3 and so on. You are given 9 such sub-grids, okay? Well you can generalize it to any n X n. I am taking here the case of 9, 3 X 3 sub grids, but it can be always generalized to higher dimensions, this is just for simplicity. So each 3 X 3 sub-grid here will be called as a block.

Some of the cells here, so each cell will be either filled or it might be blank, so some of the cells here are already filled and that is why it is a partially complete collection of 3 X 3 sub grids, rest of the cells are incomplete here. So what the solution of this Sudoku puzzle demands, it demands from you the following you have to fill the blank cells here. And you can fill the cells the empty cells with any number in the set 1 to 9 but there are some restrictions.

The restrictions are the following, you have to ensure that once you fill the empty cells here then each row should have the numbers 1 to 9 exactly once, each column after filling all the empty cells should have each of the numbers 1 to 9 exactly once and each block namely each 3 X 3 sub-grid after filling all the empty cells should have all the numbers 1 to 9, of course exactly once. So these are the restrictions for it.

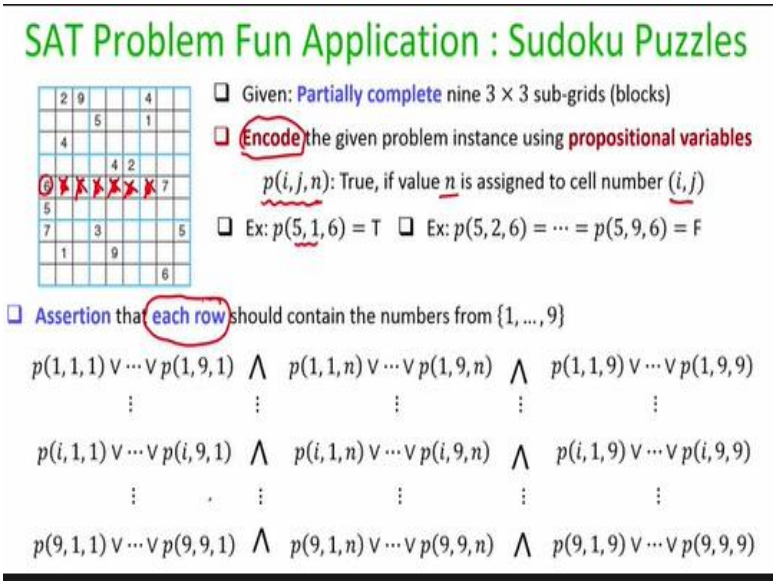
So for instance, this is a problem instance which might be given to you and you have to come up with a solution. Solution means you have to come up with possible values, which you can fill in this empty cells satisfying the above restrictions, it turns out that one of the solutions is this. So very often we write computer programs which take problem instances of Sudoku puzzle and come up with possible solutions.

Of course, it might be possible that the given problem instance do not have any solution that means my partially complete problem instance is such that I cannot satisfy all these three conditions simultaneously. In that case I will say that no solution is there. So a Sudoku solver or a computer program which tries to solve a Sudoku puzzle will try to come up with one possible assignment; if at all it is there which satisfies all these three conditions simultaneously.

So now you can think in your mind that how exactly you will proceed to solve an instance of Sudoku puzzle. You will think in your mind, okay, what if I fill this cell with 1 and then what if I fill this cell with 2 and so on and then for each possibility you will check whether I am successfully able to satisfy all the three conditions if not then let me go back and change one of the possible assignments and then repeat the process.

That is the rough algorithm you will follow in your mind and that is what precisely computer program for solving the Sudoku puzzle will try to emulate.

**(Refer Slide Time: 20:08)**




So what I am going to do here is I am going to represent an instance of Sudoku Puzzle using a compound proposition and I will show how I can convert an instance of solving this Sudoku puzzle into an instance of SAT problem. So the first step here will be that I will do some encoding here. I will encode this given problem instance using propositional variables, so I introduce a propositional variable p(i, j, n).

So this entire thing is a propositional variable, p is not only propositional variable. My entire propositional variable is p(i, j, n) and this propositional variable p(i, j, n) will be true provided n is assigned to the cell number i, j. It is like saying the following if in my mind I am thinking that ok I am going to assign the value one here then that means the propositional variable p(1, 1, 1) is true, otherwise p(1,1,1) is false.

So since I am already given some filled cells here, in this particular case, for instance if you take the fifth row and the first column, it is already occupied with the value 6. That is why the

propositional variable p(5, 1, 6) is true. And since 6 is already there in the first column of the fifth row, I cannot put 6 in any of the remaining columns of the fifth row. That is why all the remaining propositional variables p(5, 2, 6), p(5, 3, 6) p(5, 9, 6) will be automatically false.

So that is how I can encode the initial conditions which are given to me by this propositional variable. So now remember a solution for the Sudoku puzzle has to satisfy three conditions, namely, each of the values 1 to 9 should occur exactly once in each of the rows. Each of the values 1 to 9 few occur exactly once in each of the columns and so on. So let me try to encode the assertion that each row should contain the numbers from 1 to 9.

And I want to represent this criteria, this condition, this restriction by a compound proposition using the propositional variable that I have introduced here. So my claim here is that this expression namely this disjunction of 9 variables, the variables are here p(i, 1, n), p(i, 2, n) up to p(i, 9, n) I am focusing on row number i here where I of course ranges from 1 to 9 because I have 9 rows here.

So my claim here is that the disjunction of these 9 variables represent a fact that or represent a condition that my row number i should contain the number n in one of its columns. So remember I would like the number n to be either available in the first column of the $i^{th}$ row or in the second column of the $i^{th}$ row or in the ninth column of the $i^{th}$ row, how do I express this condition ? By this compound expression.

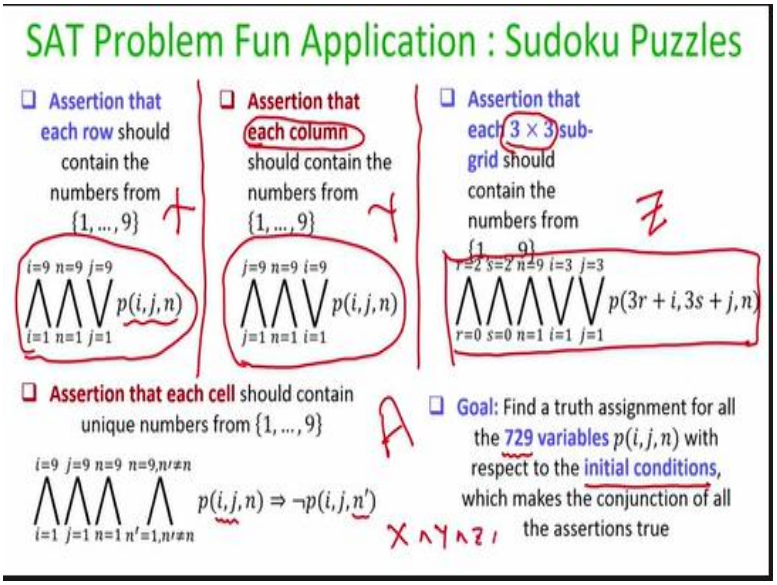
So that means now if I make my n to be 1, this disjunction of 9 variables represent that I want the number 1 to appear in at least in exactly in one of the columns of the $i^{th}$ row, it could be either in the first column or in the second column or in the ninth column. In the same way the disjunction of this 9 variables represent that I want a number n to appear in one of the 9 columns of the $i^{th}$ row.

And disjunction of this 9 variables represent that I want a number 9 to appear in one of the columns of the $i^{th}$ row. So I am fixing the $i^{th}$ row and I am stating the fact that my $i^{th}$ row should have 1 appearing somewhere, 2 appearing somewhere, n appearing somewhere and 9 appearing somewhere by this various expressions. And now if I take the conjunctions of all this expressions separately, that represents the fact that I want to assert here that my row number I should have all the numbers from 1 to 9 appearing somewhere, that is what the

conjunction of this 9 sub expressions will represent. So now what I can say is that if I iterate over i = 1 to 9, so remember whatever I did till now is with respect to the row number i. But now I have 9 possible rows so my i can be 1, my i can be 2, my i can be 3, and my i can be 9.

So if I iterate over various possible values of i and if I take the conjunctions of all this disjunctions separately then that represents my first restriction of the solution or the first assertion that my solution should satisfy namely each of the rows should have the numbers 1 to 9 somewhere.

**(Refer Slide Time: 25:59)**




So in the short form I can represent whatever we have discussed till now by this compound proposition. So I am iterating over the row i =1 to 9 and column j = 1 to 9 and possible values in the cell from 1 to 9 and I want the value n to appear somewhere in the $i^{th}$ row in one of the columns of the $i^{th}$ row. In the same way we can do a similar exercise and come to the fact or come to the conclusion that if I want to state the assertion that each column should have the occurrence of 1 to 9 somewhere;

Then that can be represented by this compound proposition using exactly similar exercise that we have done for deriving the compound proposition to represent the assertion about our row condition. So I am not going to do this I leave it to you as an exercise and what is our third assertion the third condition that our Sudoku puzzle should satisfy? Well our Sudoku puzzle should satisfy the condition that each three cross three sub-grid should contain the numbers 1 to 9.

Again I leave it as an exercise for you. You can verify that this condition can be represented by this compound proposition. It might look complicated but again if you do it slowly for each of the blocks you try to derive the condition and then you iterate it over all possible 9 blocks, you can verify that this compound proposition represents the assertion about the 3 X 3 sub-grid that your Sudoku puzzle should satisfy.

Finally, we also need to ensure the following. We need to ensure that our solution for the Sudoku puzzle should assign only unique values to each cell. It should not happen that a solution says that a particular cell should take the value say 8 and simultaneously the same cell should take value 7 that should not happen because that is an invalid solution. Once a cell is occupied with the value it cannot take any other value.

The reason we have to separately state this condition is. You might try to separately find the solution satisfying the row condition. You might try to find a solution satisfying your column condition and then in the same way you might try to find the solutions satisfying your block condition. Now what may happen is that the solution you try to find for the row condition might conflict with the solution that you are trying to find for the column solution.

That means the row solution may say that okay, you assign the value say 1 to the first row and first column, whereas the column condition says that you assign the value 2 to the first row first column that should not happen. So that is why we have to explicitly state this and it is very easy to verify that this condition can be represented by this compound proposition. This compound proposition says that once you have assigned the value n to a cell i, j then you cannot assign a different value n' to the same cell i, j.

Where n and n' or n and n' are different. So now we have four compound propositions I have the compound propositions say X, I have to compound proposition Y, I have the compound propositions Z and I can call this compound proposition as A. Now the goal for a Sudoku puzzle solver is the following it has to find the truth assignments for the propositional variables p(i, j, n) and how many such propositional variables are there?

There are 729 propositional variables because i ranges from 1 to 9, j ranges from 1 to 9, n ranges from 1 to 9. Namely I have to find the truth assignments for each of the 729 cells that is why I have 729 variables. Some of these propositional variables might have already taken

truth assignments because of the initial conditions. So remember some of the cells of the problem instance might be already filled that means it has already fixed the truth values for the particular cells.

Those cells I cannot fill with any other value, apart from those propositional variables I have to find the propositional truth assignment for other propositional variables, so that the conjunction of X, Y, Z and A becomes true. That means now I have to verify whether this compound proposition which is the conjunction of X, Y, Z and A is satisfiable or not. So now you can see that an instance of Sudoku puzzle is reduced to an instance of SAT problem if this expression is satisfiable.

That means the problem instance, the Sudoku puzzle problem instance that is given to you is indeed solvable otherwise, it is not solvable. So that is all you can represent, you can encode real world problem instances wherever you have to find problem solutions right for computational problems into an instance SAT problem and then you can solve the SAT problem. The solution for the SAT problem will give you the solution for the computational problem that you are interested to solve.

So that brings me to the end of this lecture. Just to summarize, in this lecture we introduced the satisfiability problem there are plenty of applications of the SAT problem. We saw a fun application of the SAT problem namely the Sudoku puzzle solver, thank you.