Fundamentals Of Artificial Intelligence
Shyamanta M.Hazarika
Department of Mechanical Engineering
Indian Institute of Technology – Guwahati

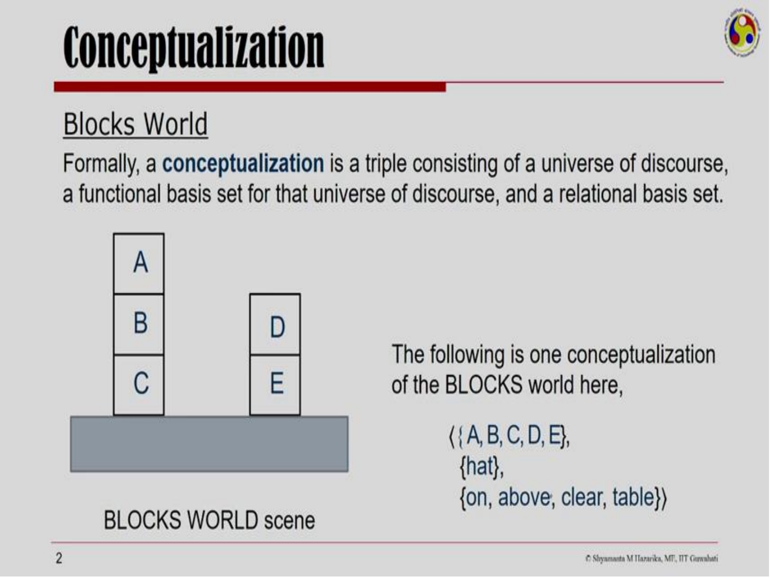
Lecture – 14
Inference in First Order Logic - I

Welcome to fundamentals of artificial intelligence we continue our discussion on knowledge representation and reasoning and our focus on first-order logic. Unlike propositional logic where every proposition is treated as a single individual unit, first-order logic is a formal language to express the content of a proposition. We have looked at the syntax and semantics of first-order logic.

First-order logic or first-order predicate calculus is called first-order precisely because it does not allow quantification over predicate symbols or function symbols. This is what distinguishes first-order logic from higher-order logics. Notwithstanding the fact that there are no predicate variables first-order logic is the knowledge representation and reasoning formalism most widely used by the artificial intelligence community.

Inference is the process of arriving at new sentences from existing sentences. This is the focus of our lecture today.

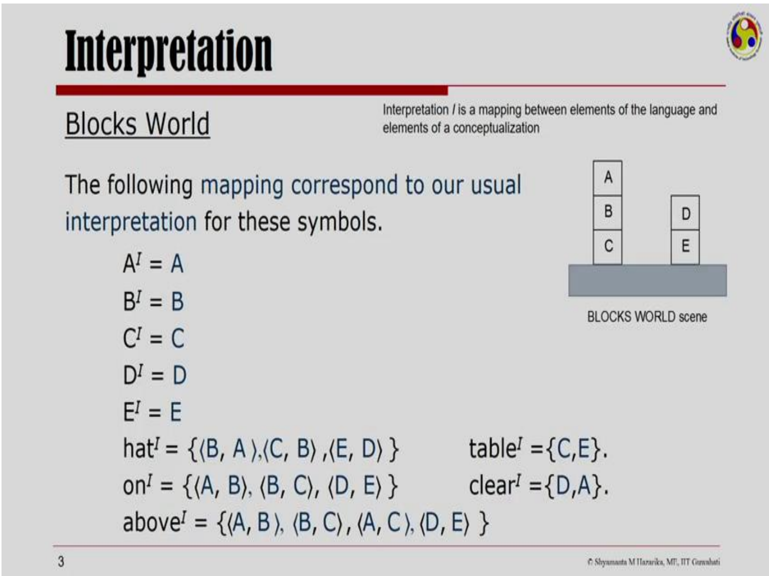
(Refer Slide Time: 02:25)



In the last class recall that we had looked at conceptualization which is about formalizing declarative knowledge. A conceptualization consists of the objects functions and relations. A conceptualization is a triple which includes A, a universe of discourse the set of objects for which knowledge is being expressed. B, a functional basis set, the set of functions being emphasized in the conceptualization and C, a relational basis set, the set of relations being emphasized in the conceptualization.

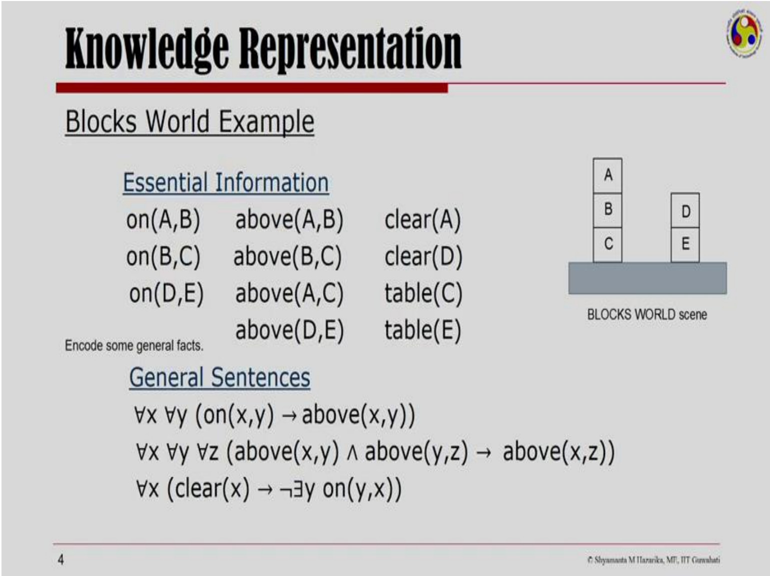
In the blocks world example that we were looking at we have the universe of discourse as these 5 blocks A, B, C, D and E. We have a function hat and four relations. The relation on holds if and only if one block is immediately above the other block. Above is a relation between two blocks if and only if one is above the other. We had looked at a relation called clear which means no block is on top of the block. And then a fourth relation table to mean a block is on the table.

(Refer Slide Time: 04:22)



An interpretation is a mapping between the elements of the language and the elements of a conceptualization. To continue the example if the first-order predicate calculus language has five object constants, then interpretation I would map the object constants to the objects in the world. The function constant hat is mapped to the tuples corresponding to the function and the relation constants are mapped to each of their extensions.

(Refer Slide Time: 05:05)

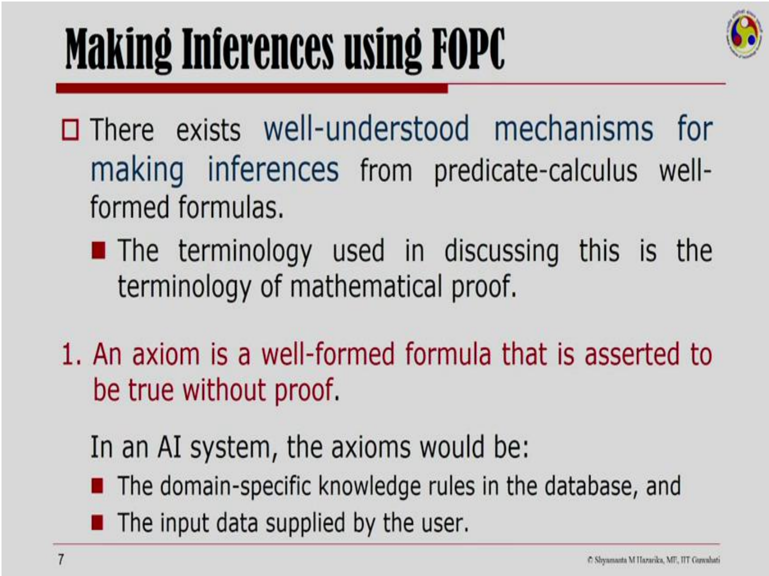


Coming back to the blocks world example, with four relations we could write down the essential information for the blocks world scene shown on the right of the screen. Apart from the essential information we could encode some general facts. The first here says that if I have a block on top of another block, it would mean that the first block is above the other block. A second general statement would be like for three blocks x, y and z. If x is above y and y is above z it would mean that x is above z. So, we are trying to capture here the knowledge that the above relationship is transitive. We have a third general statement here which says that a block which is clear would not have any other block on top of it. So, this general statements one need to realize that it also applies to block scenes other than the one pictured here.

Given the general statements and the on relation, we may not have to explicitly include all the above relations because given the on, I could always derive above. Now having removed the above relation from my list of essential information and listed these three general sentences, a conjunction of these formulas can serve as a description of the world state. Now let us say the problem is to show that a certain property is true in the given state.

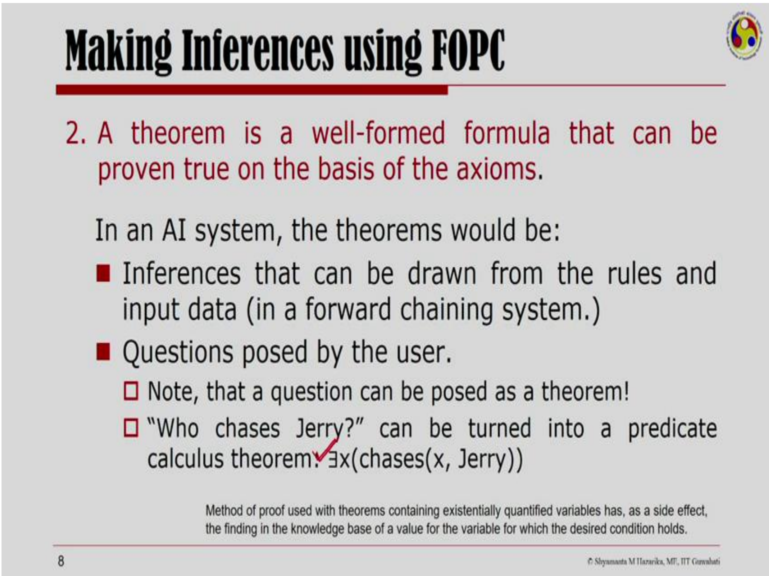
For example, let us say I want to conclude from the given information and the general statements to me that there is no block on top of block A; that is block A is clear. So, we would look for a sentence which would say there is no y on (y, A). We can deduce this fact showing that the formula logically follows from the state description. Equivalently the formula could be derived from the state description by application of sound rules of inference.

(Refer Slide Time: 08:09)



So, there exists well understood mechanisms for making inferences from predicate calculus well-formed formulas. The terminology is the terminology of mathematical proof. So, we start with an axiom. Here an axiom is a well-formed formula that is asserted to be true without proof. It could be domain-specific knowledge certain rules in the database and also include input data supplied by the user.

(Refer Slide Time: 08:55)

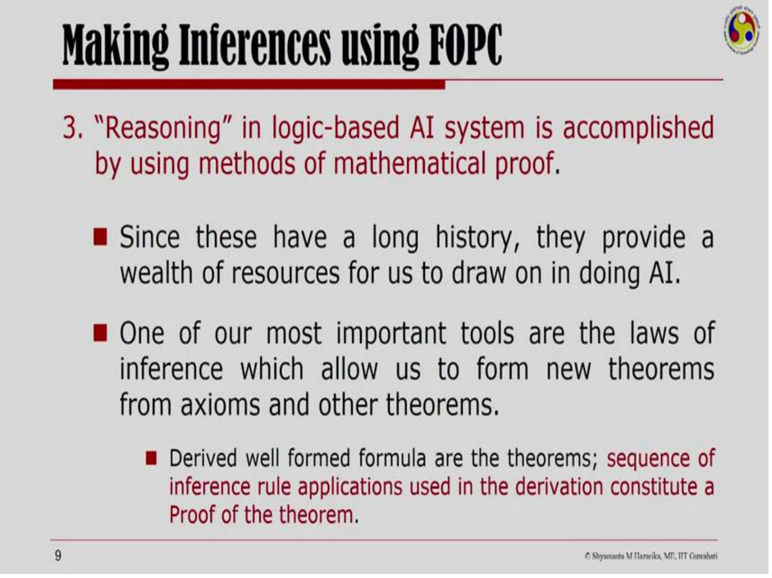


Thereafter we have a theorem which is a well-formed formula that can be proven true on the basis of the axioms. The theorems would be inferences that are drawn from the general statements or the rules and the input data. It is interesting to note that questions posed by the user could also be posed as a theorem. Like for a Tom and Jerry cartoon series type scenario, I could ask a question: Who chases Jerry?

And this could be turned into our predicate calculus theorem something like there exists an x chases x Jerry to mean that x chases Jerry. Now when I am doing proof of theorems containing existential quantifiers, that is, existentially quantified variables in them, there is a very interesting side effect. What we are actually doing is also finding in the knowledge base a value of the variable for which this desired condition holds.

And that value of the variable is the answer to the question who chases Jerry. This is referred to as answer extraction. And we will look at answer extraction in more detail in one of the subsequent lectures.

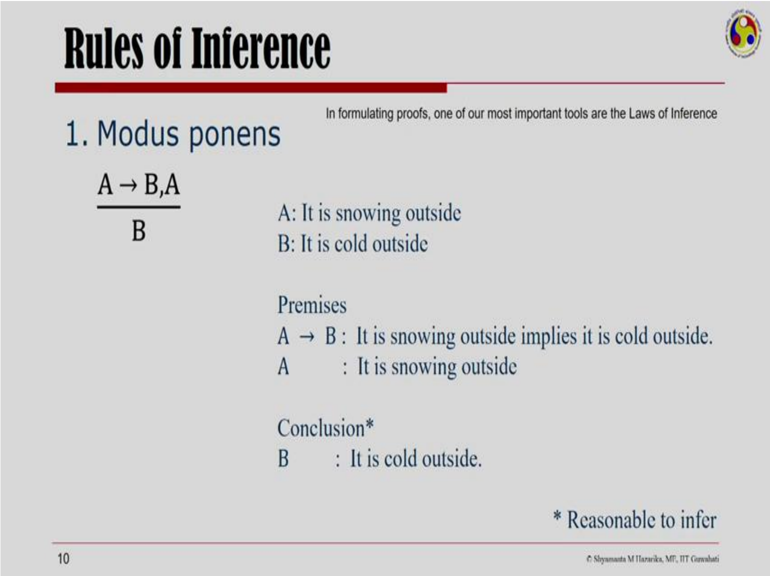
(Refer Slide Time: 10:48)



Reasoning in logic based systems is as we have been emphasizing accomplished by using methods of mathematical proof. Since, this has a long history they provide a wealth of resources for us to draw on in doing the same with an artificial intelligence. One of our most important tools for this are the laws of inference which allows us to form new theorems from axioms and other existing theorems.

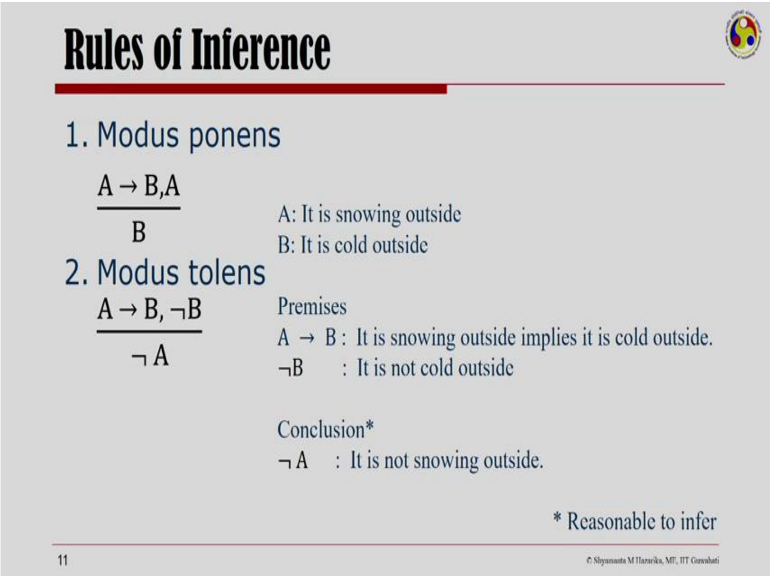
The derived well-formed formula is referred to as theorems and the sequence of inference rule applications using the derivation constitute a proof of the theorem.

(Refer Slide Time: 11:42)



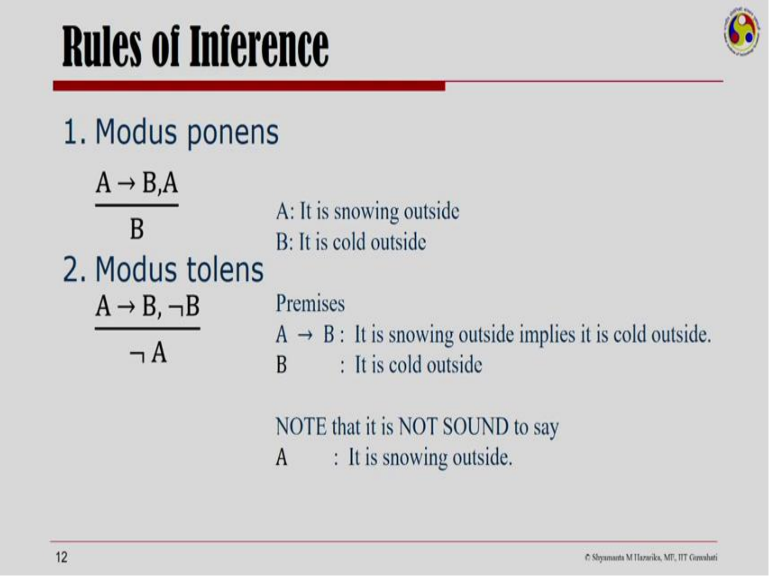
Now let us focus on a couple of rules of inference from propositional logic. This is precisely because the rules of inference that we have discussed in propositional logic is also applicable to first-order logic. And to reemphasize, in formulating proofs one of our most important tools are these laws of inference. So, here is Modus ponens A implies B, A , therefore B, that is, if I have the premise we say something like that it is snowing outside implies it is cold outside. And I know it is snowing outside then it is reasonable to infer that it is cold outside.

(Refer Slide Time: 12:40)



The next rule of inference that we would look now is modus tolens which says A implies B not B therefore not A. Let us say it is snowing outside implies it is cold outside and I am also told it is not cold outside. So, again it would be reasonable to conclude that it is not snowing outside. Let us now take a moment to realize that instead of not B ,that is, being told that it is not cold outside, if I was told that it is cold outside could I then infer that it is snowing outside.

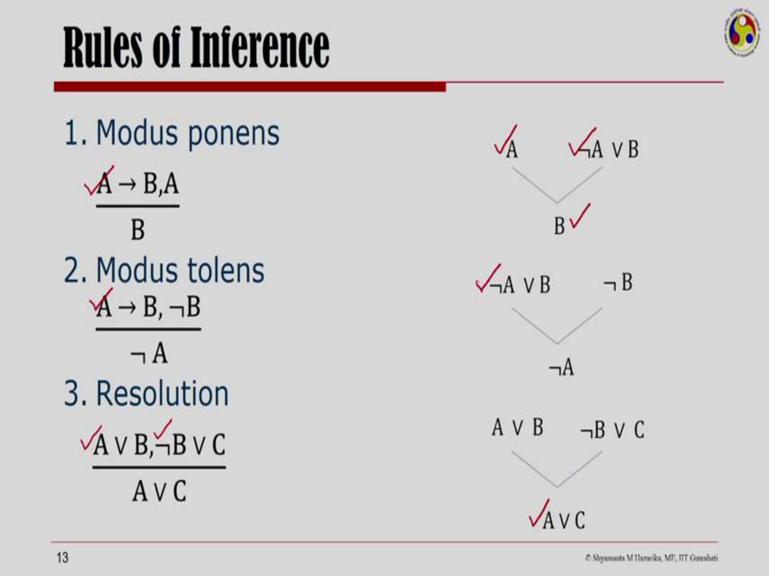
(Refer Slide Time: 13:37)



This is interesting for if I am told that it is snowing outside implies it is cold outside and now I am told that it is cold outside, note that, it is not a sound rule of inference to say it is snowing outside. This is precisely because there could be other reasons why it is cold outside. There are ways of reasoning where you would try to fill up the reason here given the premise and the fact that you observe. This sort of reasoning is called abduction but we would not cover abduction in our discussion here.

So, for us if we are told it is cold outside it is not necessarily the case that it is snowing outside.

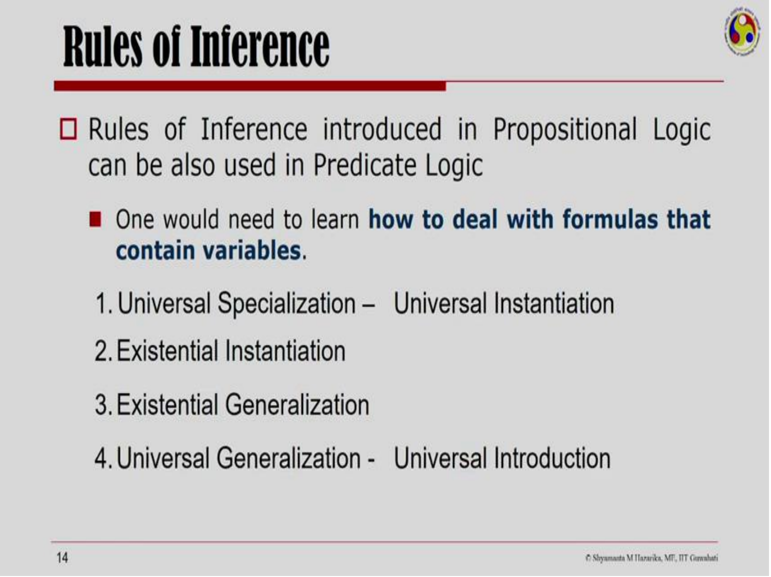
(Refer Slide Time: 14:42)



Then we focus on one of the most interesting rules of inference, I refer to as interesting because it is that makes it so easy to automate theorem proving in first order logic because of these very rule of inference called resolution. So, what we have here is A or B and another called not A or C from these two I resolve that A or C is true. Now it is interesting to see that both modus ponens and modus tolens could be looked at in terms of resolution.

I could write the implication here A implies B as not A or B and once I have an A in my premise I could resolve and get a B. Similarly, I could have modus tolens which is again writing this implicit as not A or B and having not B in the premise I could get to a not A.

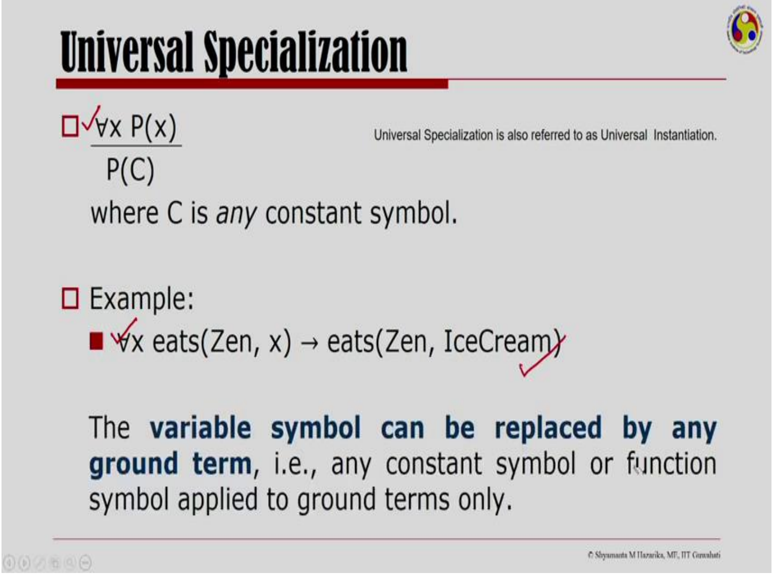
(Refer Slide Time: 16:13)



Now rules of inference that have been introduced in propositional logic as I told you can also be used in predicate logic. But then if you remember the only thing that makes the difference here is the quantifiers. So, one need to learn how to deal with formulas that contains variables and quantifiers to use the rules of inference in first order logic. We will focus our attention on four different methods to deal with quantified sentences in first order logic.

The first of this is called Universal specialization also referred to as Universal instantiation. The second is existential instantiation. Then we have existential generalization and the fourth Universal generalization also referred to as universal introduction. Let us look at each of them one by one.

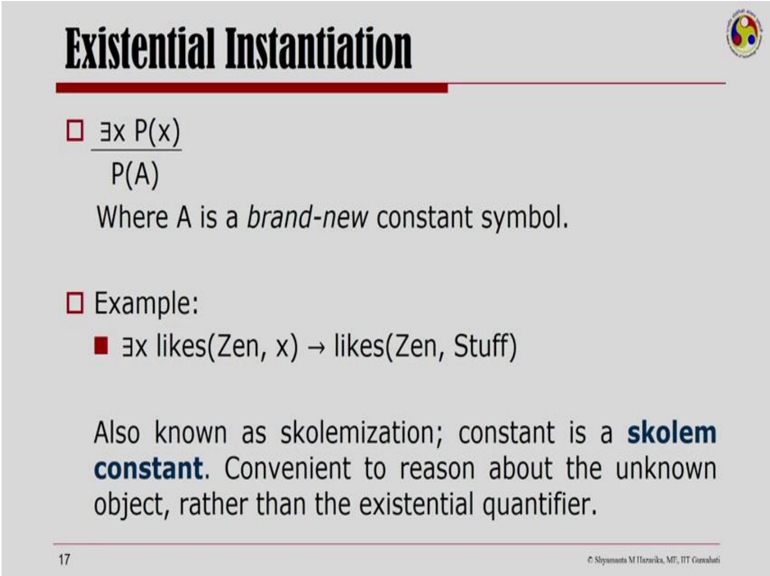
(Refer Slide Time: 17:33)



So, Universal specialization it is about having a universally quantified statement saying for all x, P(x) is true. I could then very well say that P(C) for C being any constant symbol is true. Here is an example if I know that zen eats everything, that is what is being said here. For all x, zen eats x. So this part of the statement here is saying that zen eats everything. So, given this zen eats everything I can very well infer that zen eats ice cream.

So, the variable symbol can be replaced by any ground term that is any constant symbol or function symbol applied to ground terms only. That is Universal specialization or also referred to Universal instantiation because a universally quantified variable x has been instantiated to a constant symbol C.

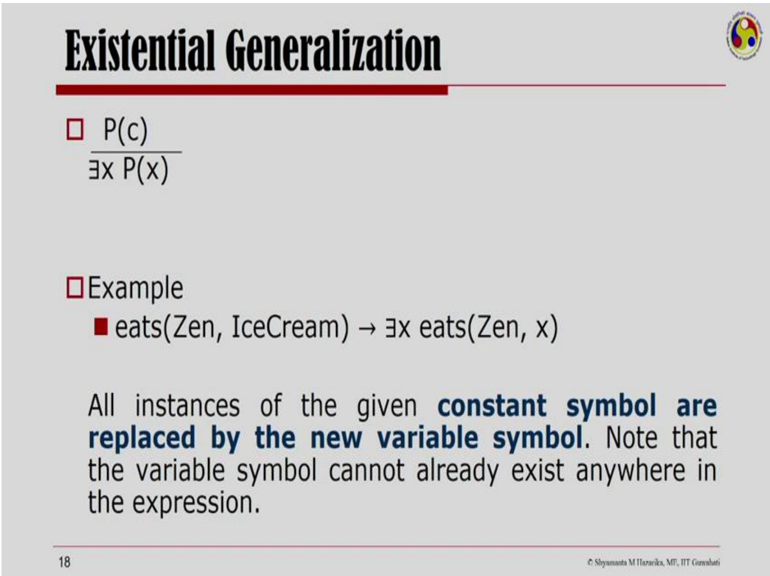
(Refer Slide Time: 18:53)



The next is existential instantiation. So if I have an existentially quantified statement that there is an x for which P(x) is true, then I can say that that x is A, a brand new constant symbol and therefore I could say P(A). So, here is the example if I have a statement that says that zen likes x, there is some x which zen likes, then I can very well imagine that x to be some stuff. So, in existential instantiation a variable is replaced by a brand new constant and that constant should not occur in this or any other sentence in the knowledge base.

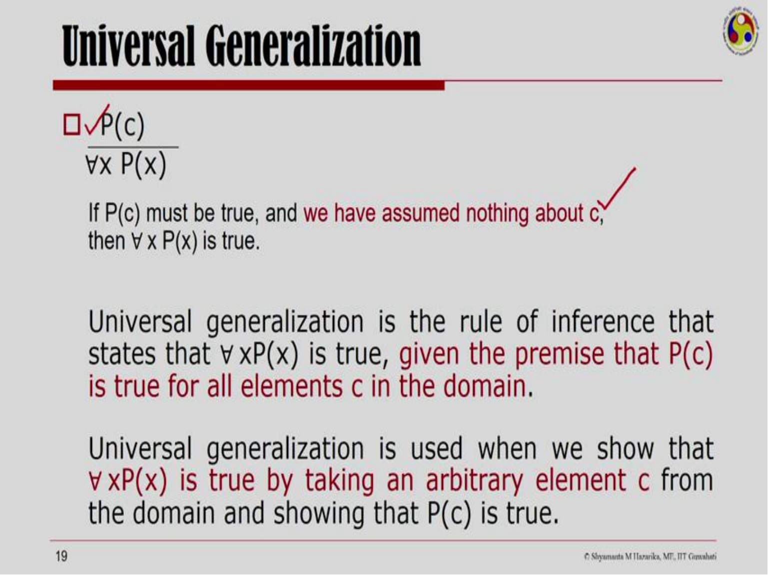
This is because I am talking of something existing which should not have existed in the knowledge base. So, this is also known as skolemization and the constant is called as skolem constant. It is convenient to reason about the unknown object rather than the existing shell quantifier.

(Refer Slide Time: 20:26)



The next is called existential generalization which is that if I know specifically that P(C) is true where C is an object constant, then I can definitely say that there is an x for which P is true. Like if I know that zen likes to eat ice cream I can very well say that there is an x which zen eats. All instances of the given constant symbol are replaced by the new variable. Note that the variable symbol cannot already exist anywhere in the expression.

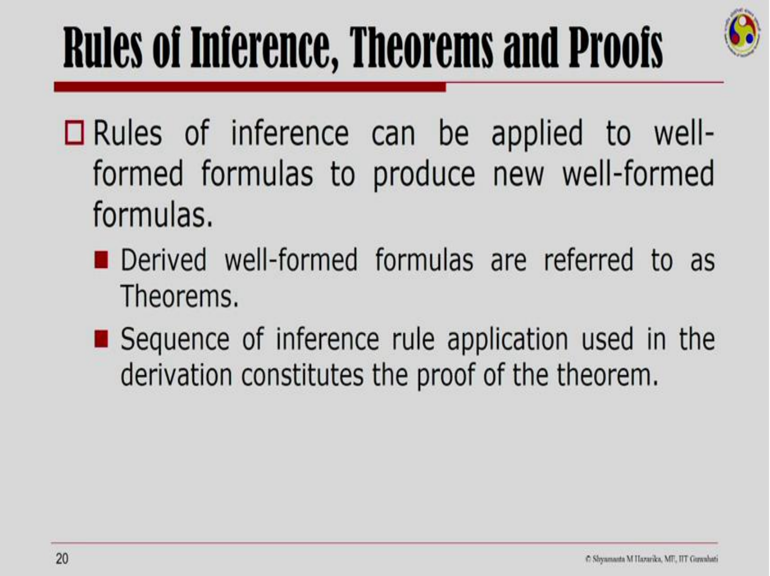
(Refer Slide Time: 21:07)



Now the last of these treatments of quantified sentences the universal generalization is counterintuitive. Here we are saying that if I have a statement that says P(C), I could generalize this to say for all x P(x) but then one needs to deal with this very, very carefully. One needs to remember that if P(c) must be true then we have assumed nothing about C. Only under that state I could generalize it to say that for all x P(x) is true.

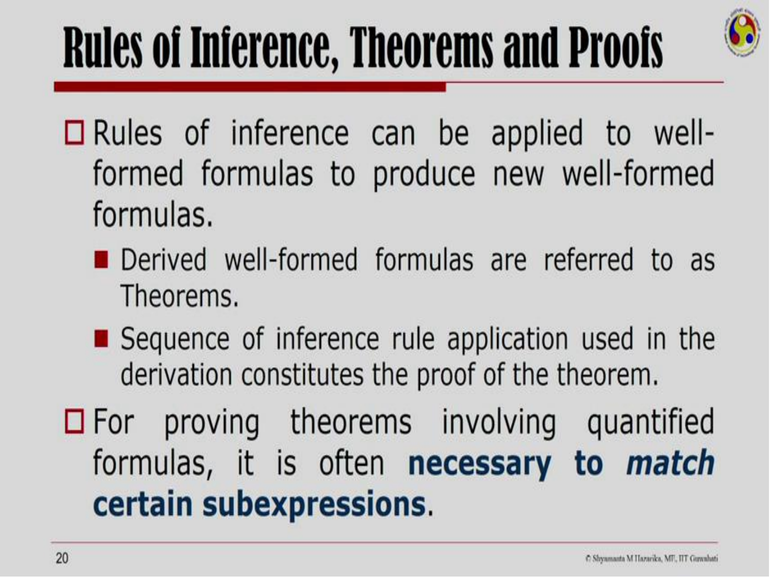
Universal generalization is under the premise that the PC statement is true for all elements of C in the domain. Therefore, you do not commit to anything about C. We assume nothing about C. And used when we show that for all x P(x) is true by taking an arbitrary element C for which we do not assume anything.

(Refer Slide Time: 22:28)



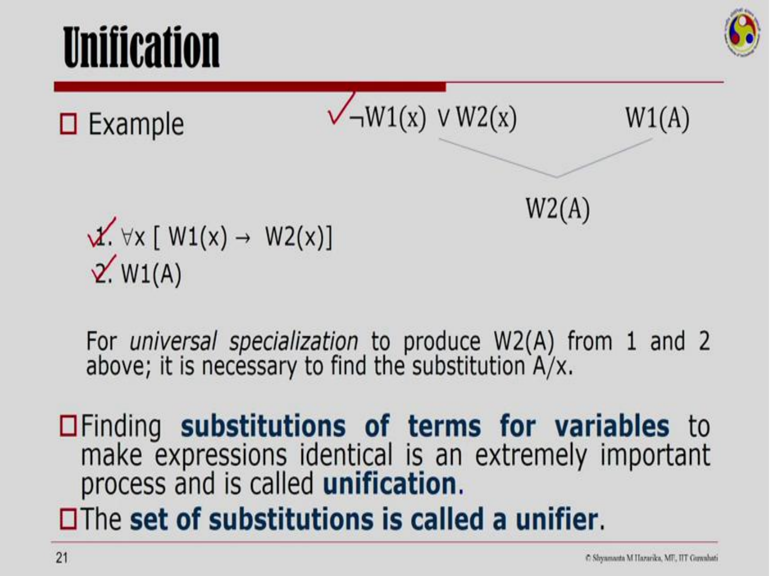
Now the rules of inference that we were discussing can be applied to produce new well form formulas. And as I have been emphasizing the new derived well-formed formula keeping in view the parlays of mathematical proofs that we use, the derived well-formed formula are referred to as theorems. And the sequence of the rule application using the derivation is what constitutes the proof of the theorem.

(Refer Slide Time: 23:08)



For proving theorems involving quantified formulas it is often necessary to match certain sub expressions.

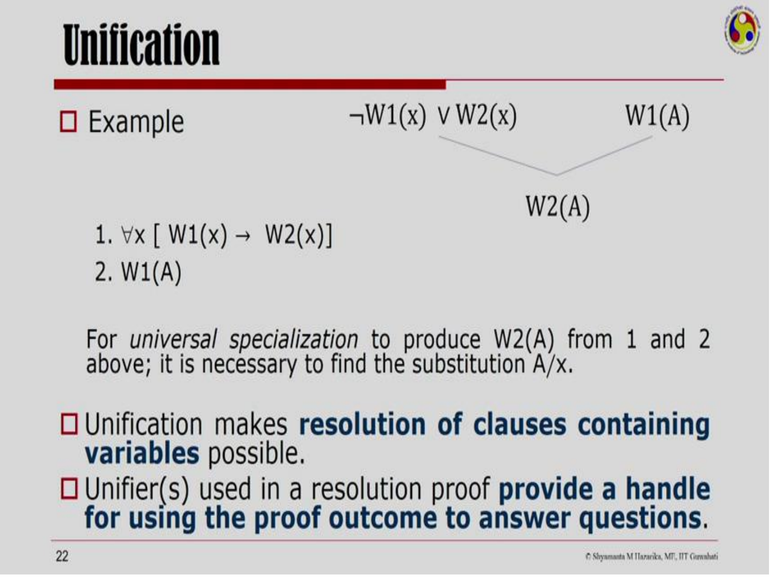
(Refer Slide Time: 23:19)



Let us see what we mean by that here is an example where I have a statement which says for all x W1 of x implies W2 of x and I am also told W1 of A. So given these two statements I could convert the implication to not W1 or W2. And I have my second statement W A now instead of x here if I would have had an A I could have immediately resolved them to derive W2 of A. But how do you produce W2 of A here from 1 and 2.

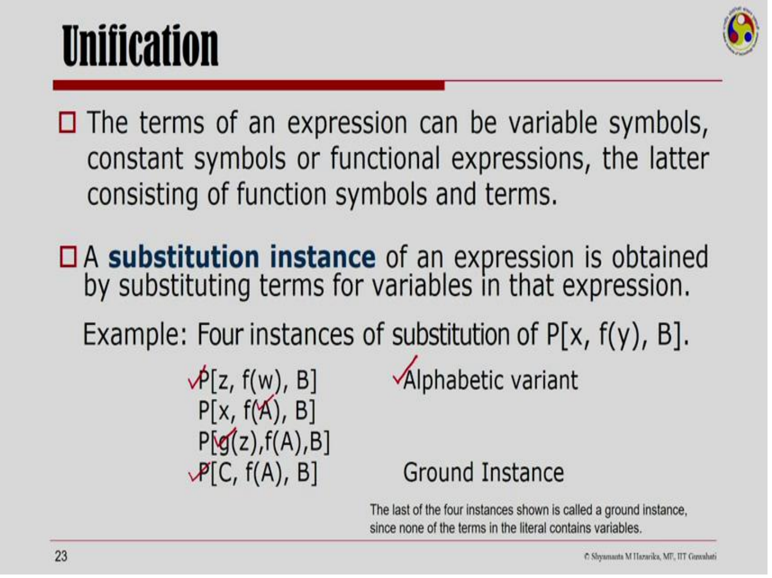
One needs to realize that in order to do that I have to have a substitution A for x in here to actually get this statement rewritten as not of W 1 A or W 2 A and in which case it resolves with W 1 A to give me W 2 A. This idea of finding substitutions of terms for variables to make expressions identical is an extremely important process in going towards first order proofs and is called unification. The set of substitution that is being used is called the unifier.

(Refer Slide Time: 25:07)



So, unification makes resolution of clauses containing variables possible. And the unifiers using a resolution proof actually provide a handle for using the proof outcome to answer questions because if I was using some unifier and it had given a value for an initial existential variable I know I am looking for an answer to the question that was posed as an existential formula.

(Refer Slide Time: 25:45)



The terms of an expression that I am trying to substitute can be variable symbols constant symbols are functional expressions. The later consisting of function symbols and terms. So, here is a substitution instance of an expression obtained by substituting terms for variables. Of the four instances if you focus on the first instance here I have substituted variable x in the expression with a new variable z, variable y with a new variable w and the constant B has been kept intact.

So, the first of the four instances of substitution for the expression is actually an alphabetic variant except for replacing variables with newer variables I have not really gone ahead and did a real substitution. In the second instance shown here I have substituted an A for y. In the third instance the x has been substituted by a function of z. And this is interesting we will focus on this very soon. Here what this means that the variable x is some function of z and therefore I have substituted it as a function of z.

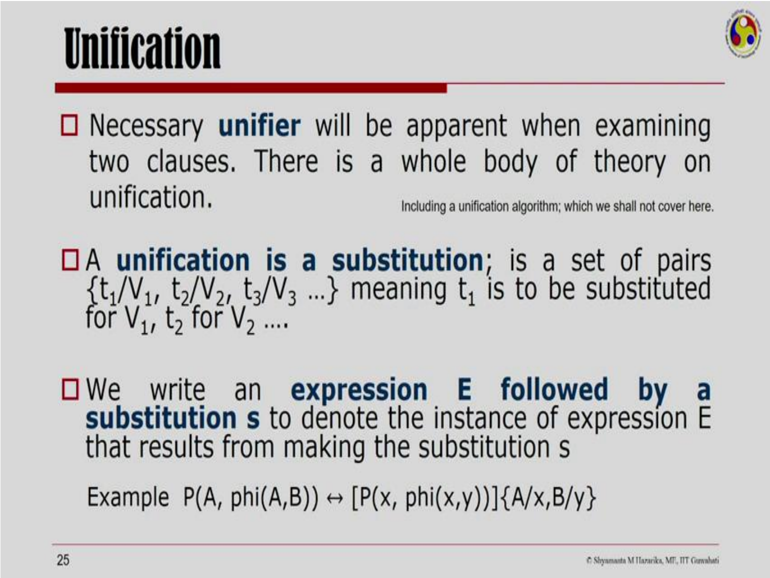
The last of the four instances shown here if you take note does not have a variable in it. All of the variables have been replaced by constants. This is called a ground instance. A ground instance is when none of the terms in the literal contains variables.

(Refer Slide Time: 27:57)

Now let us look at a couple of properties of substitution that we were trying to do for unification. For two formulas Phi and Psi at least one of which contains variable I can talk of a substitution U that makes them identical. So, that substitution is called a unifier for Phi and Psi. So, here is an example I have two sentences P (A, x) and P (y, z) now I use a substitution U A for y and x for z one needs to realize that often we have more than one unifier for a pair of formulas.

For example, in the above case I could go with a substitution which could be A for y B for x and B for z this is another unifier for these two sentences. Variables or term containing variables can also be used for another variables as you must have realized when I was showing you the example substitutions.

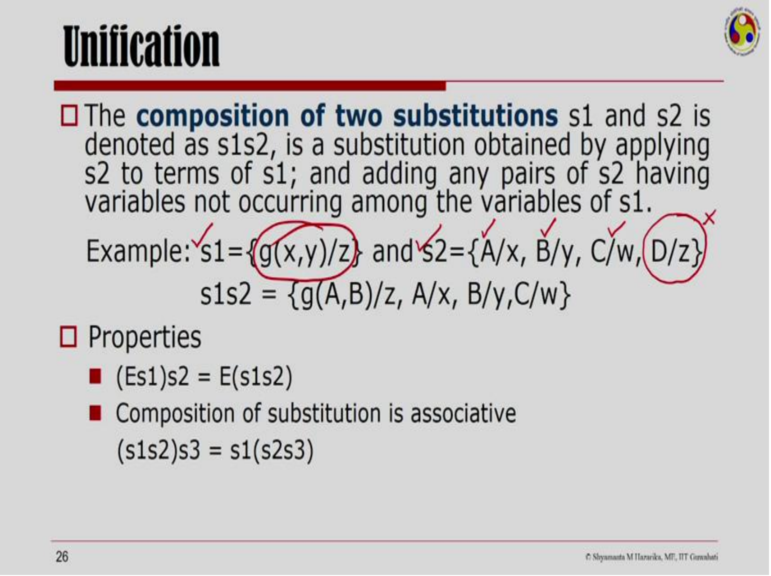
(Refer Slide Time: 29:36)



Necessary unifier will be apparent when you examine two classes. There is a whole body of theory on unification including unification algorithm which we will not cover here. Having said that let us look at unification a little bit more closely. So, our unification is a substitution and is a set of pairs t1 for V1, t2 for V2, t3 for V3 so on and so forth. Meaning that t1 is to be substituted for V1, t2 is to be substituted for V2. We write an expression E followed by a substitution s to actually denote the instance of expression E that results from making the substitution.

Like in the example here I have our expression PA phi AB let us say I use a substitution A for x and B for y then this instance of expression E that results from making the substitution I could show by writing E followed by the substitution s.

(Refer Slide Time: 29:36)



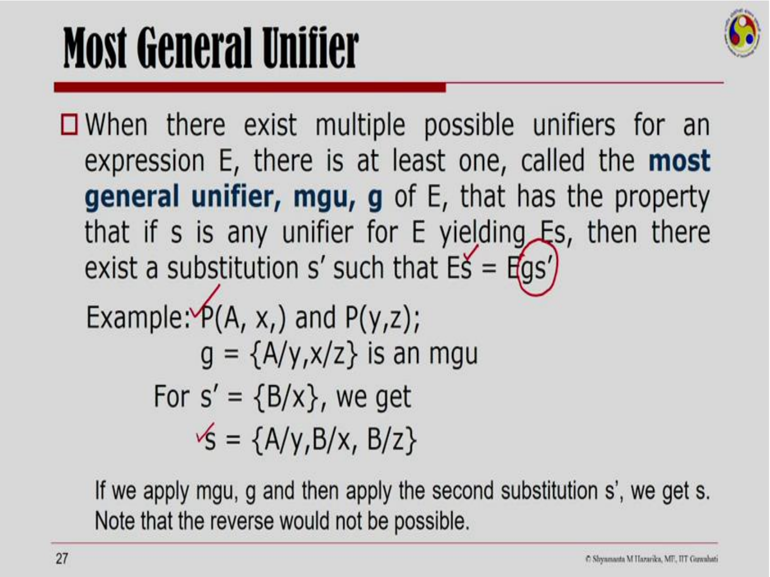
The composition of two substitutions s1 and s2 is denoted as s1 s2 and is a substitution obtained by applying s2 to terms of s1 and adding any pairs of s2 having variables not occurring among the variables of s1. So, let us look at an example here I have a substitution s1 which says that the function g x, y is substituted for z and the second substitution s2 is a substitution that substitutes A for x, B for y, C for w and D for z.

Now the composition of these two substitutions s1 s2 would be obtained by applying the second substitution onto the first substitution. So, when I apply s2 the substitution in s1 you should realize that this x need to be replaced by an A as shown here because A is for x and this y needs to be substituted by B because B is for y in substitution s2. So, here I have g A, B for z and then because these pairs for x, y and w did not occur in my original substitution s1. So, I have to add them into my substitution s2, so I added the pairs of s2 having variables which did not occur among the variables of s1, so I had to add for x, y and w. But for z, s2 had a substitution saying D for z which s1 already stated as z of x, y for z and therefore this does not get included into the list of the composition of the two substitutions.

Now let us look at a couple of properties of substitutions the first says that if I have an expression and a substitution s1 it is followed by a substitution s2. It is same as the expression being followed by the composition of the two substitutions. The second property is about composition of substitution being associative. So, if I have s1, s2 then s3 this is same as s1 and composition of s2 s3.

However, substitutions in general are not commutative. So, composition of substitution s1 s2 is not equal to s2 s1.

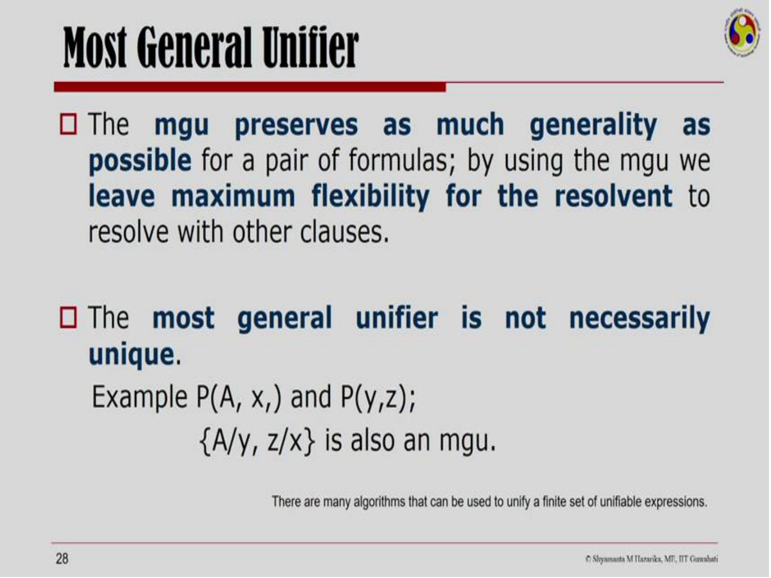
(Refer Slide Time: 35:08)



Having said that we could have multiple unifiers for an expression I can always find out at least one which is called the most general unifier which has the property that if s is any unifier for E yielding Es then I could always find a substitution s prime, such that the composition of the most general unifier with s prime will give me the same substitution as s. So, let us look at an example and try to understand what is the most general unifier.

So, here is our substitution A for y and x for z, given the two statements above P(A,x) and P(y, z). I can now think of a substitution s prime which is B for x so we get a substitution s which is a composition of g and s prime. And this substitution says that A for y B for x B for z is same as the substitution which is a composition of these two. Now if we apply the most general unifier and then apply the second substitution s prime we get s. Note that the reverse would not be possible.

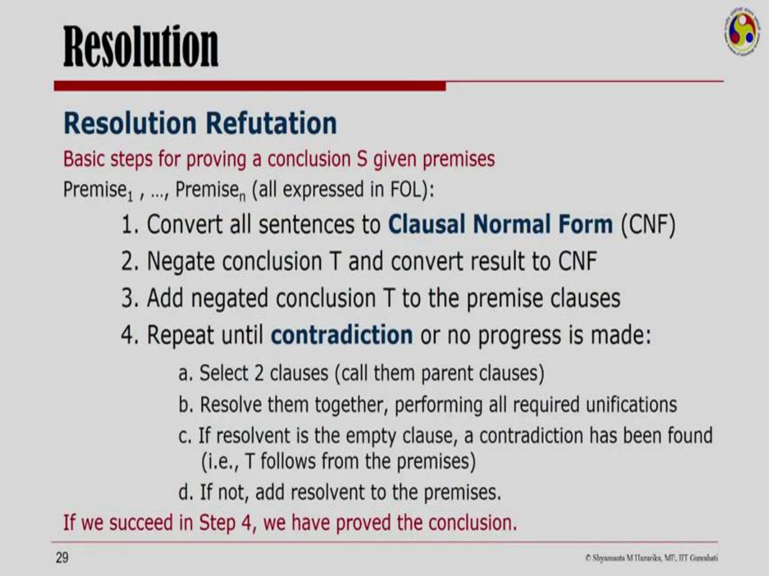
(Refer Slide Time: 37:07)



The most general unifier is important for it preserves as much generality as possible for a pair of formulas. And by using the most general unifier we are actually leaving maximum flexibility for the resolvent to resolve with other clauses. The most general unifier is not necessarily unique. I could have more than one most general unifier for a group of sentences. So, here is my pair of formulas P of A, x and P of y, z and you could see that A for y and z for x is also a most general unifier.

There are many algorithms that can be used to unify a finite set of unifier expressions we shall be talking of them during the course of our discussion.

(Refer Slide Time: 38:21)

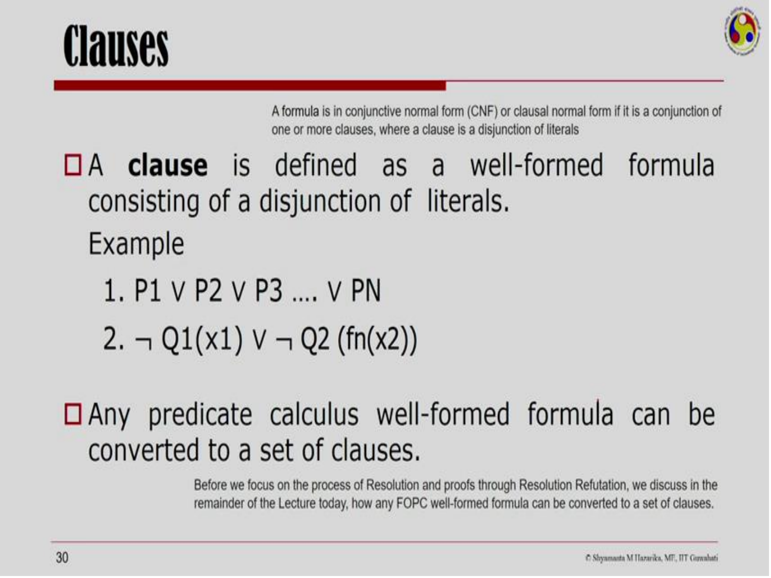


Now for completeness let us quickly look at a concept that we will cover in more depth in our subsequent lecture. So, this is the concept of resolution refutation which is about getting to a proof for a given statement by adding the negated concretion to the premise classes. So, what we do is we have a set of satisfiable clauses given to us. And a conclusion T to be proved the conclusion is negated and added to the original set of clauses.

The whole set of clauses if it now resolves to an empty clause a contradiction has been found. If we have started with a non contradictory set of clauses and now a contradiction has been found the only reason the contradiction has come about is because of the negated conclusion, that means, the negated conclusion is not true. And it means that the conclusion follows from the premise. So, this is what we do we convert all sentences to clausal normal form.

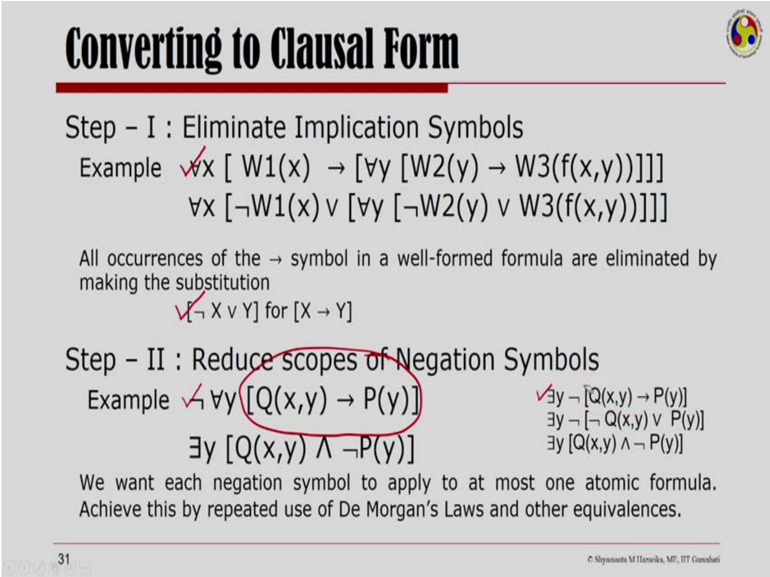
We negate the conclusion and convert result to the clause and normal form. Add the negated conclusion to the premised clauses and we repeat these four steps until contradiction or progress is made. If we succeed in step four in getting a contradiction, we have proved the conclusion. We will come back to this in more depth in subsequent lecture but what I want to emphasize here is that the first step in a resolution refutation proof is to convert all sentences to the clausal normal form. And this is what we will try to understand in the remainder of the lecture today.

(Refer Slide Time: 40:41)



A formula is in conjunctive normal form or clausal normal form if it is conjunction of one or more clauses where a clause is a disjunction of literals. And any predicate calculus well form formula can be converted to a set of clauses. Before we focus on the process of resolution and proof through a resolution refutation we will discuss how a first-order predicate calculus well-formed formula can be converted to a set of clauses.

(Refer Slide Time: 41:20)



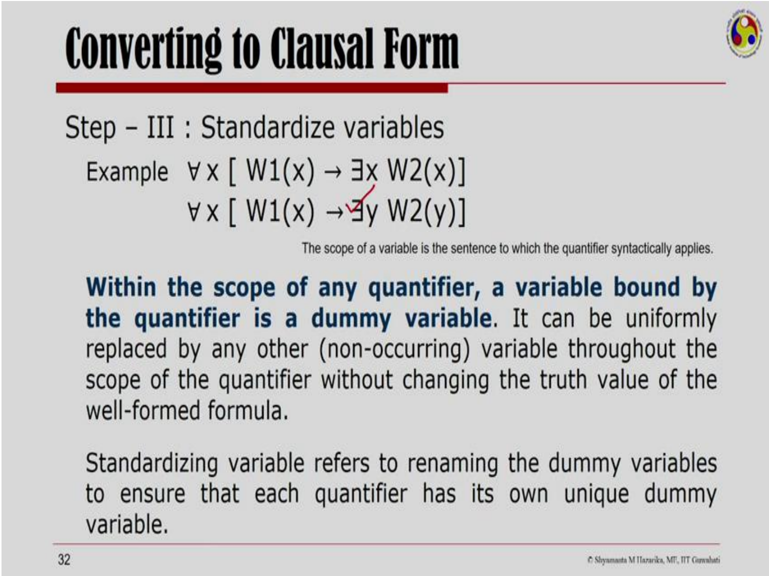
So, the first step in converting to clausal form is to eliminate the implication symbol. So, here is an example I have a statement that says for all x W1x implies for all y W2y implies W3f x. So, I could replace this by not of w1 or whole of this sentence. And for the inside implication I could again bring not of W2 or W3. So, all occurrences of the implication symbol in a well-formed formula are eliminated by making the substitution not x or y for any implication x implies y.

The second step is to reduce the scope of the negation symbols. For example, here the negation symbol applies to the whole of the formula we want each negation symbol to apply to at most one atomic formula and we achieve this by repeated use of the De Morgan laws and other equivalences. Like here given this statement not for all y this portion which is an implication I could first push in the negation and write there exists y not and this implication.

There after this implication could be replaced by not of Q or P and now we have discussed this while discussing propositional logic that when I have a statement that says not of P or Q, I could have not of P or Q being replace by not of P and not of Q. So, this statement already having a not there. Another not coming in here will give me a Q and here I will have not of Py and that is a conjunction.

So what we have done by repeated application of the De Morgan laws is that we have reduced the scope of the negation symbol from the whole formula to an atomic formula here.

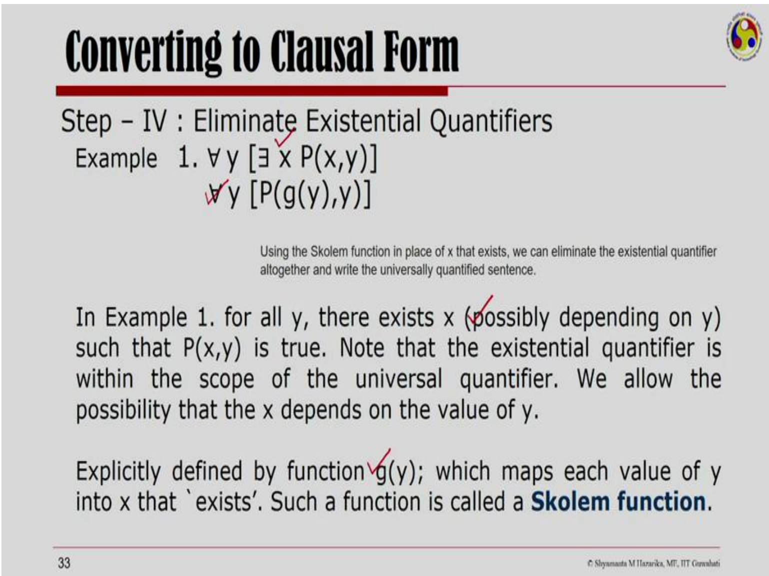
(Refer Slide Time: 44:12)



The third step in converting to clausal form is about standardizing variables. So, the scope of a variable is the sentence to which the quantifier syntactically applies and in here we could see that for all x this variable applies to the whole of the statement. Whereas there exists x this existential x applies to only W2 within the scope the variable is bound by the quantifier. And this bonding is to a dummy variable.

So what we can do is we can uniformly replace any other non occurring variable throughout the scope of the quantifier without changing the truth value of the well form formula. That is given this x and this x here I could very well write this as y without any change in the truth value of the well form formula. Standardizing variables refers to renaming the dummy variables to ensure that each quantifier has its own unique dummy variable.

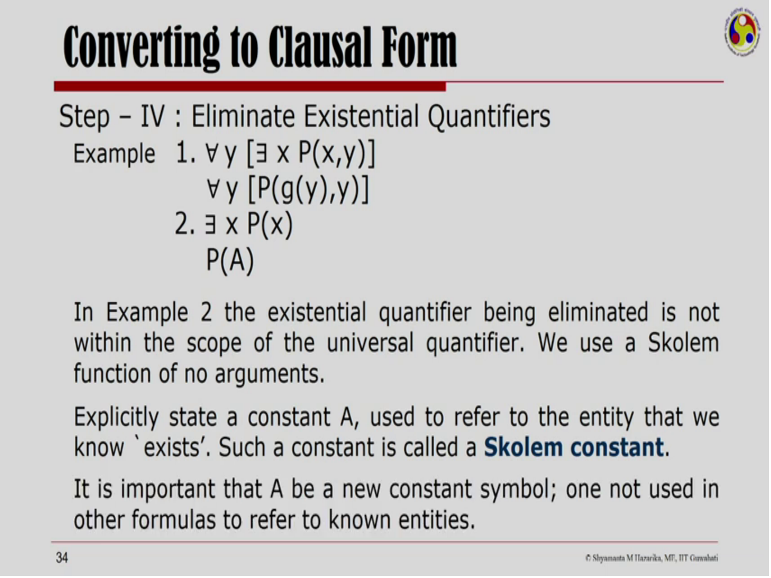
(Refer Slide Time: 45:30)



So, we have this fourth step on converting to clausal form which is about eliminating existential quantifiers and this is very vital for one to understand this step very carefully. So, here is an example we say for all y there exists an x, P of x, y. Now one should realize that this x that exists is dependent on the y. So, we would use a function g of y to remove this existential x. So, we can eliminate the existential quantifier altogether and write the universally quantified statement.

For all y there exists x now this existence we should understand is possibly depending on y such that P of x, y is true. So, we allow an explicit function g of y which maps each value of y into x that exists and such a function is called a skolem function.

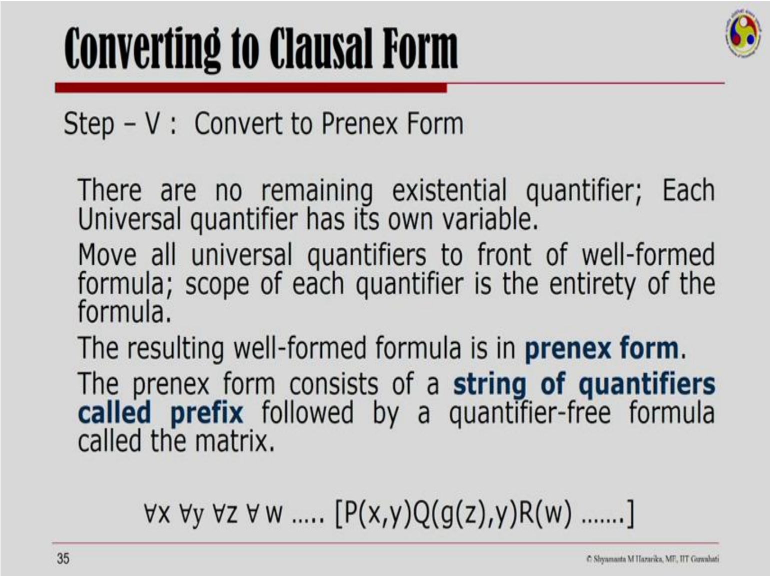
(Refer Slide Time: 46:57)



We will now look at a second example where we again have an existential quantifier there exists x P of x. But now if you look closely this existential quantifier there is no preceding universal quantifier. So, in example 2 the existential quantifier being eliminated is not within the scope of a universal quantifier. So, we can use a skolem function of no arguments which is finally a skolem constant and we can explicitly state the constant.

Now this constant is referring to an entity that we know exists and this constant is called the skolem constant. It is important that the skolem constant that we talk of be a new constant symbol one that I have not used in other formulas to refer to known entities.

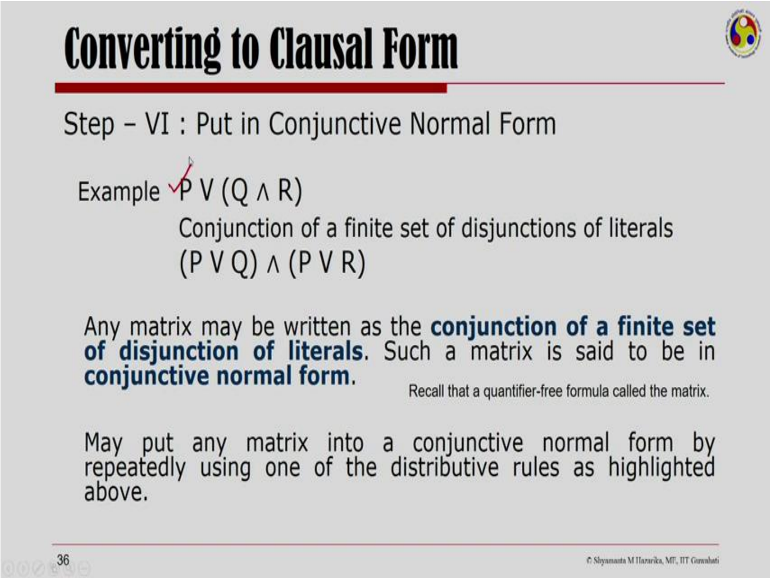
(Refer Slide Time: 48:01)



The fifth step in converting to the clausal form is about getting to the prenex form. One should realize that we have eliminated the existential quantifiers there are no remaining existential quantifier. And each universal quantifier has its own variable. So, what we can do is move all universal quantifiers to front of the well form formula. The scope of each quantifier is the entirety of the formula.

Now and the resulting well form formula is said to be in prenex form. So, the premix form actually consists of a string of quantifiers which is called the prefix followed by a quantifier free formula called a matrix.

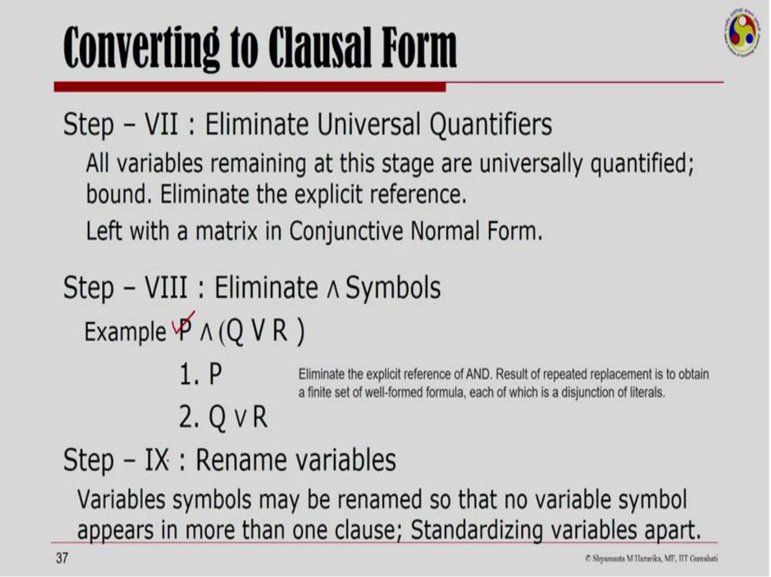
(Refer Slide Time: 48:58)



The next step is to put this in the conjunctive normal form. So, the matrix that we have got may be written as the conjunction of a finite set of disjunction of literals such a matrix is said to be in conjunctive normal form. Now recall that when I say matrix here what I am referring to is a quantifier free formula. So, here is an example that highlights the way the conjunctive normal form is arrived at by repeatedly using one of the distributive laws.

Like if I have P or Q and R I could write it as a conjunction of a finite set of disjunction of literals P or Q and P or R.

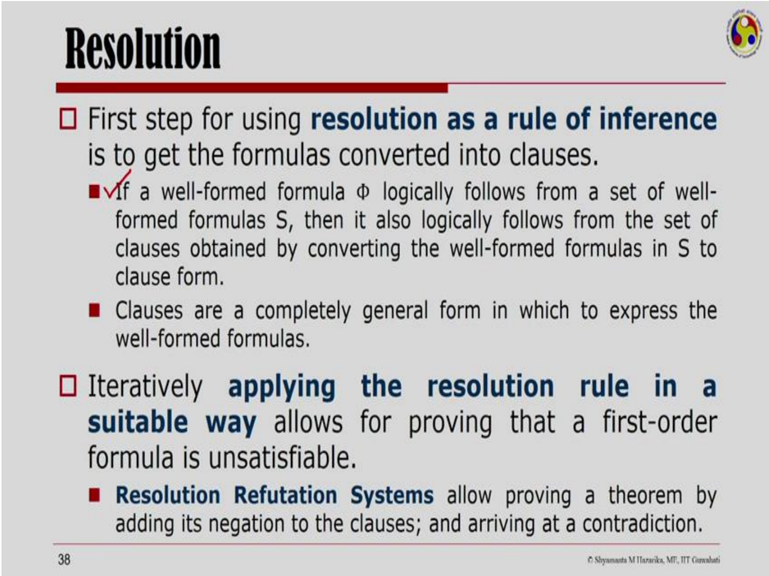
(Refer Slide Time: 49:58)



Finally, I am having all the variables universally quantified bound and I could now eliminate the explicit reference to the universal quantifiers. And what I will be left with is a matrix which is in conjunctive normal form. Finally, I can eliminate the conjunction symbols. I eliminate the explicit reference to the conjunction symbols by breaking each of them as a clause. So, if I have a statement P and Q or R I could write as P and I could write s Q or R.

Once I have written these statements in as individual statements the final step is about standardizing the variables apart. What it literally means is that the variable symbols may be renamed so that no variable symbol appears in more than one Clause. So, we have taken a 9-step process of converting a first-order predicate calculus statement to clausal form.

(Refer Slide Time: 51:24)



One needs to realize that when I am using resolution as a rule of inference getting to the clausal normal form is the first step. Now, one thing to note here is interesting result that if, I have a well-formed formula Phi which logically follows from a set of well-formed formulas S, then it also logically follows from the set of clauses obtained by converting the well-formed formulas S to the clausal form.

And therefore I could work on the clausal form on resolution and still assure that what I am getting is what would have logically followed. Clauses are a completely general form in which to express the well-formed formulas. Iteratively if we apply the resolution rule in a suitable way, we would be able to prove that a first order formula is unsatisfiable. Now resolution refutation systems allow proving a theorem by adding its negation to the clauses as I was discussing.

And arriving at a contradiction this we would take up for discussion in our next class thank you very much.