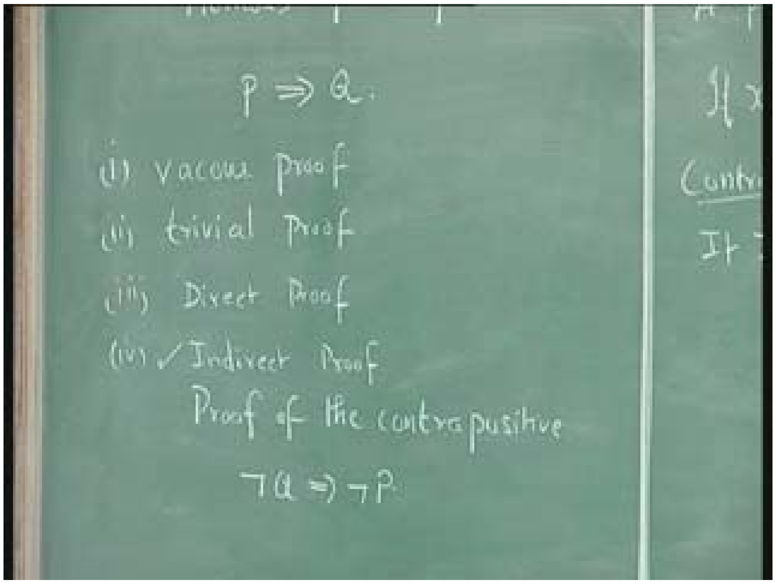
Discrete Mathematical Structures
Dr. Kamala Krithivasan
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture # 7
Methods of Proof

So we have been considering methods of proof. We found that if the form of the theorem is in the form of a tautology then by its nature it will be a theorem. Then we considered statements of the form P implies Q. hence, for this we considered that, if the statement of the theorem is of the form P implies Q then several types of proofs are considered, one is vacous proof where if you show that P is false because the implication will be true when the premise is false.

If you show P is false then P implies Q will be true, that sort of a proof is called a vacous proof. Then you have what is called trivial proof, the implication is true if Q is true, so if you just prove Q is true then because it is an implication P implies Q will be true. Such a proof is called a trivial proof. Then you can assume that P is true and prove that Q is true which is called direct proof. We saw an example of a direct proof. The last one is in direct proof or proof of the contrapositive. This also you can call as proof of the contrapositive.

(Refer Slide Time: 03:12)

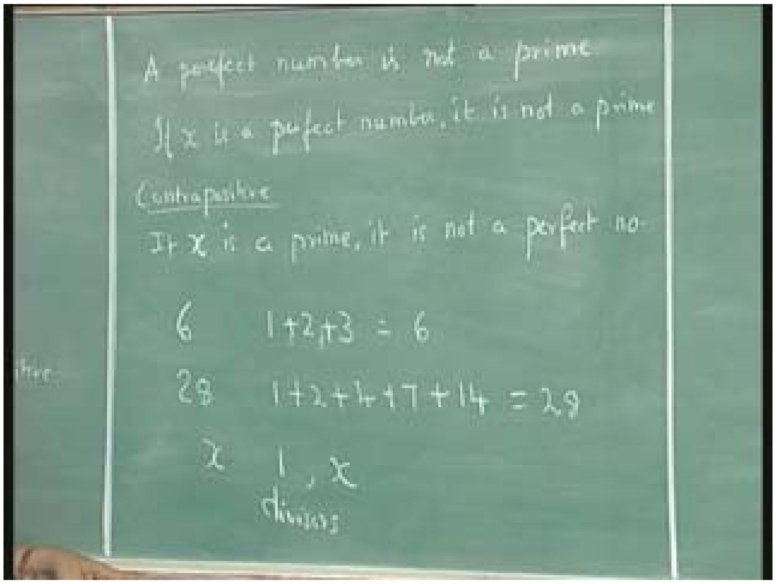


$P \Rightarrow Q$.

(i) vacous proof
(ii) trivial proof
(iii) Direct Proof
(iv) ✓ Indirect Proof
Proof of the contrapositive
$\neg Q \Rightarrow \neg P$

We know that the contrapositive of P implies Q is NOT Q implies NOT P. So, instead of assuming P and then trying to prove Q we can assume NOT Q and try to prove NOT P. Such a proof is called an indirect proof. We shall see an example of indirect proof now.

(Refer Slide Time: 05:11)



A perfect number is not a prime

If x is a perfect number, it is not a prime

Contrapositive

If x is a prime, it is not a perfect no.

6 1+2+3 = 6

28 1+2+4+7+14 = 28

x 1, x

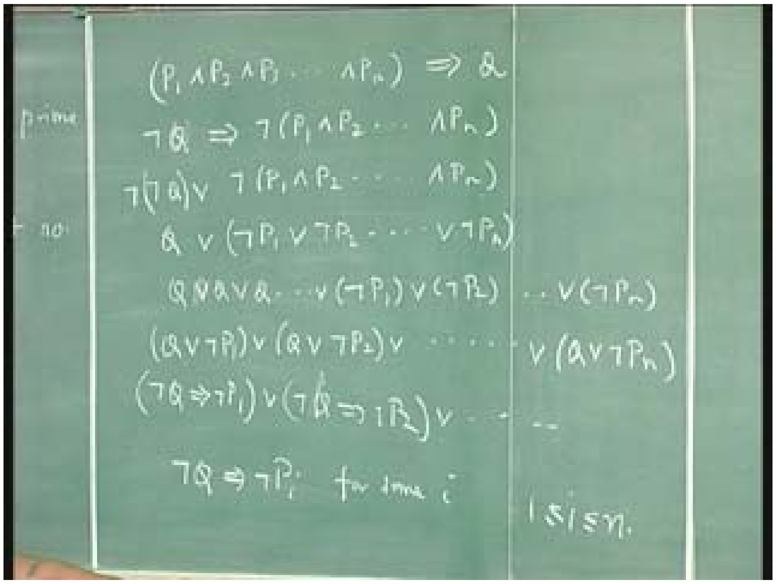
divisors

Consider this statement; a perfect number is not a prime. Now, what is a perfect number? A perfect number is one where if you sum up the divisors it should be equal to the number. For example, 6 is a perfect number, the divisors of it are 1 2 and 3 and if you sum them up you will get 6. That is, 1 plus 2 plus 3 is equal to 6. Then 28 is a perfect number because the divisors include 1 plus 2 plus 4 plus 7 plus 14 so 1 2 4 7 14 are divisors of 28 except itself and if you sum them up it is equal to 28. So the theorem states that a perfect number is not a prime.

Now the contrapositive statement, actually you can look this theorem like this; if x is a perfect number it is not a prime. The contrapositive statement will be if x is a prime it is not a perfect number. So we can easily prove this statement, what is a prime? If x is a prime then it has got only divisors 1, x. The divisors of x are 1, x and leaving out x you have only 1. So the sum cannot be equal to x so it is not a perfect number. This is the sort of proof you give and such a proof is called an indirect proof.

Later on we will see one more example at a different stage. Now, instead of having one statement P implies Q you may have something like that P_1 AND P_2 AND P_3 AND P_n implies Q. the statement of the theorem can be something like this. In that case how would you prove it? Now you can write this as NOT of, the contrapositive will be NOT Q implies NOT of P_1 AND P_2 AND P_n OR NOT of NOT Q OR NOT of P_1 AND P_2 AND P_n. Using De Morgan’s law this will be NOT of P_1 OR NOT of P_2 OR NOT of $P_{n.}$

(Refer Slide Time: 09:00)



$$(P_1 \wedge P_2 \wedge P_3 \cdots \wedge P_n) \Rightarrow Q$$

$$\neg Q \Rightarrow \neg(P_1 \wedge P_2 \cdots \wedge P_n)$$

$$\neg(\neg Q) \vee \neg(P_1 \wedge P_2 \cdots \wedge P_n)$$

$$Q \vee (\neg P_1 \vee \neg P_2 \cdots \vee \neg P_n)$$

$$Q \vee Q \vee Q \cdots \vee (\neg P_1) \vee (\neg P_2) \cdots \vee (\neg P_n)$$

$$(Q \vee \neg P_1) \vee (Q \vee \neg P_2) \vee \cdots \vee (Q \vee \neg P_n)$$

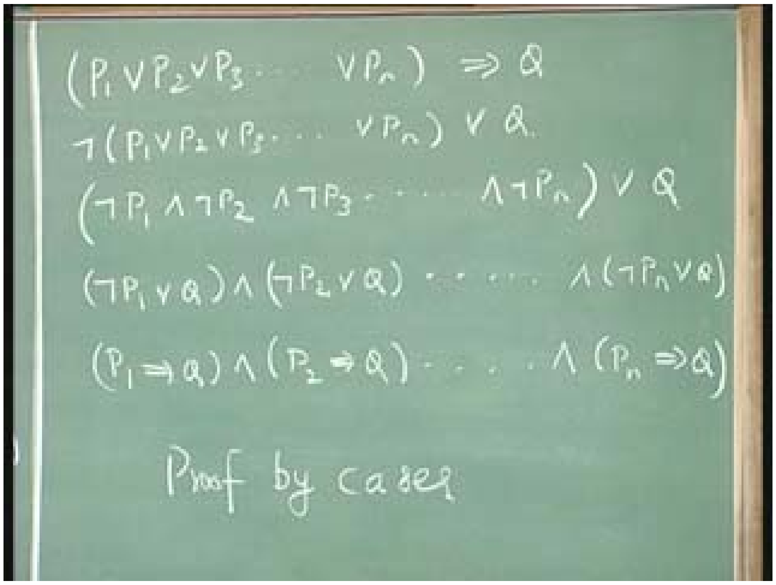
$$(\neg Q \Rightarrow \neg P_1) \vee (\neg Q \Rightarrow \neg P_2) \vee \cdots$$

$$\neg Q \Rightarrow \neg P_i \text{ for some } i \quad 1 \leq i \leq n.$$

And because OR is associative you can write it as Q. And you can also use several Qs Q OR Q OR Q like that or you can use OR NOT of P_1 OR NOT of P_2 OR NOT of P_n. Now you can combine each Q with one P and write as Q OR NOT of P_1 OR Q OR NOT of P_2 OR etc, OR Q OR NOT of P_n. And this will be NOT Q implies NOT P_1 this can be written like this because you can convert the OR again into implication it will be like this OR NOT of Q implies NOT of P_2 OR and so on. So it is just enough if you prove NOT of Q implies NOT of P of i for some i where i is between 1 and n. So if the statement is of the form P_1 and P_2 and P_3 and P_n implies Q then it is enough if you just prove one statement like this. Again we are using some sort of an indirect proof.

Now the statement can be something like this; P_1 OR P_2 OR P_3 OR P_n implies Q. This you can write as NOT of P_1 OR P_2 OR P_3 OR P_n OR Q. And using De Morgan’s laws this will be NOT of P_1 AND NOT of P_2 AND NOT of P_3 AND NOT of P_n OR Q. Using distributivity this will be; NOT of P_1 OR Q AND NOT of P_2 OR Q etc AND NOT of P_n OR Q. This you can write as P_1 implies Q AND P_2 implies Q AND P_n implies Q. So you must prove for every P_i that P_i implies Q. P_1 implies Q you have to prove, you have to prove P_2 implies Q, you have to prove P_3 implies Q and so on. You have to prove each and every statement here and this is called proof by cases. So if the statement of the theorem is of the form P_1 AND P_2 AND P_3 AND P_n implies Q using the contrapositive it is enough if you just prove that NOT of Q implies NOT of P_i for some i. It is enough if you prove just for one i then the result will hold.

(Refer Slide Time: 11:07)



$(P_1 \vee P_2 \vee P_3 \cdots \vee P_n) \Rightarrow Q$

$\neg(P_1 \vee P_2 \vee P_3 \cdots \vee P_n) \vee Q$

$(\neg P_1 \wedge \neg P_2 \wedge \neg P_3 \cdots \wedge \neg P_n) \vee Q$

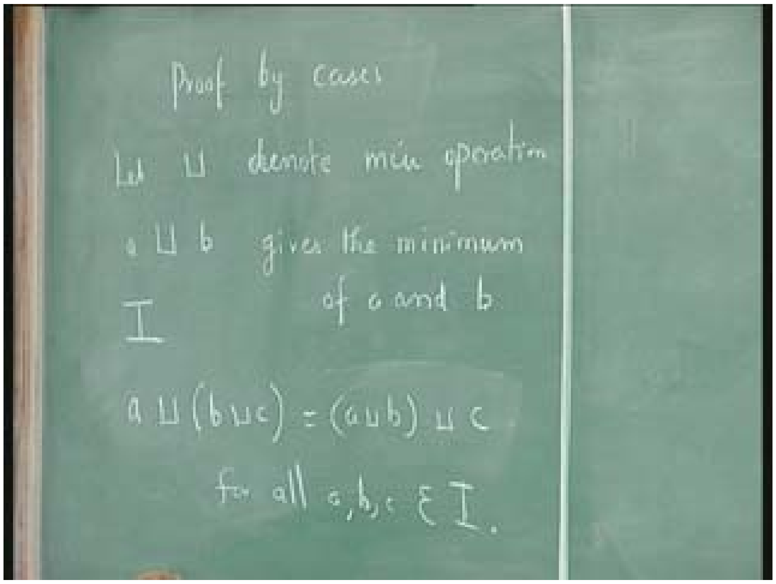
$(\neg P_1 \vee Q) \wedge (\neg P_2 \vee Q) \cdots \wedge (\neg P_n \vee Q)$

$(P_1 \Rightarrow Q) \wedge (P_2 \Rightarrow Q) \cdots \wedge (P_n \Rightarrow Q)$

Proof by cases

Whereas if the statement of the theorem is of the form P_1 OR P_2 OR P_3 OR P_n implies Q then you have to prove P_1 implies Q, P_2 implies Q, P_n implies Q and you have to prove every statement one by one. So let us consider an example of proof by cases.

(Refer Slide Time: 12:57)



Proof by cases

Let $\sqcup$ denote min operation

$a \sqcup b$ gives the minimum of a and b

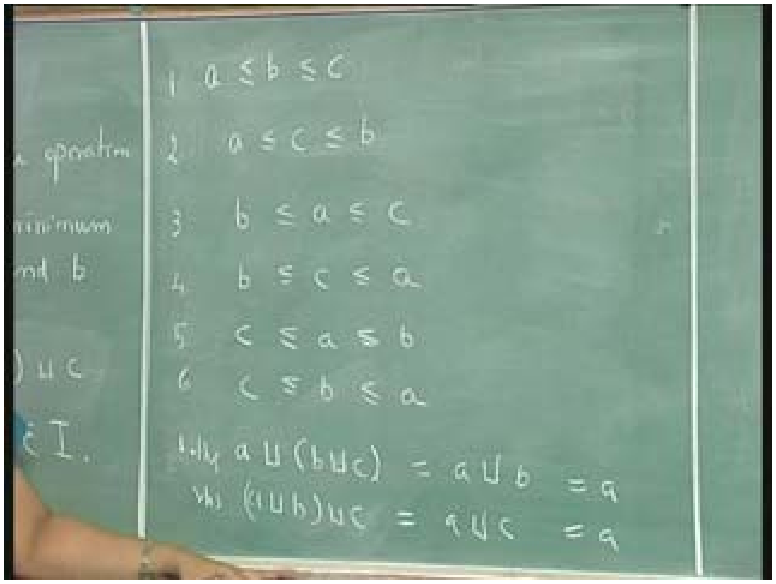
I

$a \sqcup (b \sqcup c) = (a \sqcup b) \sqcup c$

for all $a, b, c \in I$.

Let this denote the min operation that is a so this b gives the minimum of a and b. Let us restrict ourselves to the set of integers. So we consider the set of integers. Then we want to show that this operator is associative. That is we want to prove that a for all a, b, c belonging to I. For all a, b, c belonging to I the operator min is associative. Now how do you prove? You can use proof by cases here. If you take three integers a b and c what are the possibilities? The possibilities are that a can be less than or equal to b less than or equal to c or you can have a less than or equal to c less than or equal to b. Or you can have b less than or equal to a less than or equal to c. Or you can have b less than or equal to c less than or equal to a. Or you can have c less than or equal to a less than or equal to b or c less than or equal to b less than or equal to a. These are the six possibilities, there are no other possibilities.

(Refer Slide Time: 15:00)



1 $a \leq b \leq c$

2 $a \leq c \leq b$

3 $b \leq a \leq c$

4 $b \leq c \leq a$

5 $c \leq a \leq b$

6 $c \leq b \leq a$

l.h.s $a \sqcup (b \sqcup c) = a \sqcup b = a$

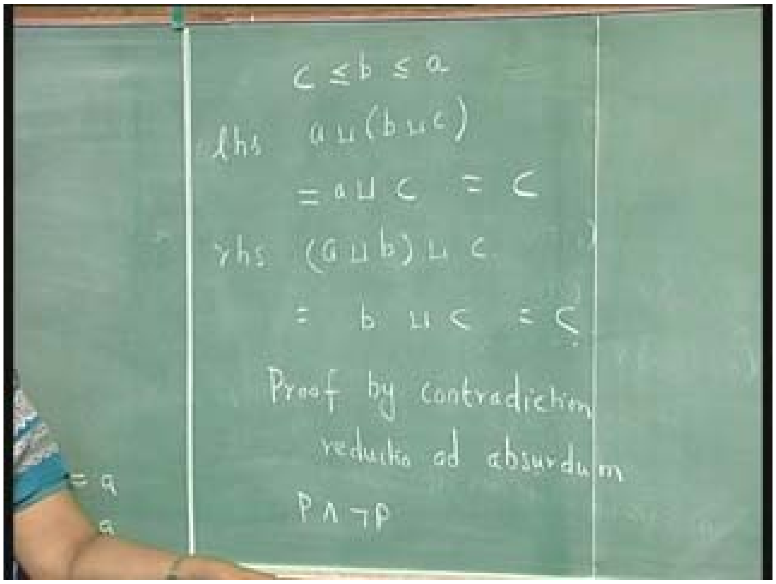
r.h.s $(a \sqcup b) \sqcup c = a \sqcup c = a$

Now for each one of this if you prove that the associative property holds then you are proving the result. This is called proof by cases. There are six cases and for each one of the case you prove that the result is true. Let us take a less than or equal to b less than or equal to c then what will you get if you take a minimum b union c, this will be a minimum then minimum of b and c that is because b is less than or equal to c you will get b and the minimum of this will be a.

On the other hand, if you take the right hand side a min b min c you have to take the minimum of a and b that is a then you have to take the minimum of a and c that is also a. So you prove that the left hand side is equal to the right hand side right. So the associative property holds. Like that you have to prove for each and every case. Let us take one more case, take the sixth case.

So let us consider the case when c is less than or equal to b less than or equal to a. In that case the left hand side is a min b min c which is equal to, what is min of b and c? That is c and what is min of a and c? That is c. And on the right hand side you have a min b min c. What is the min minimum of a and b? That is b. Then what is the minimum of b and c? That is c, so left hand side is equal to the right hand side. So we have to prove for each one of the six cases and this sort of a proof is called proof by cases.

(Refer Slide Time: 16:48)



$c \leq b \leq a$

lhs $a \sqcup (b \sqcup c)$

$= a \sqcup c = c$

rhs $(a \sqcup b) \sqcup c$

$= b \sqcup c = c$

Proof by contradiction

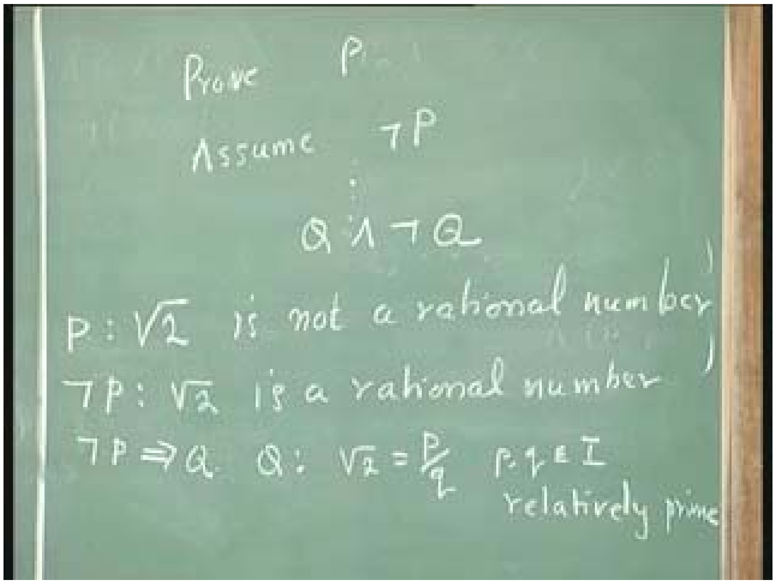
reductio ad absurdum

$P \wedge \neg P$

Then we come to the important type of proof which you would have learned in schools also that is called proof by contradiction. Or reductio ad absurdum we have already seen what a tautology is, what a contradiction is and what is a contingency. A contradiction is a propositional form which is always false something like P AND NOT P this will always be false. So in this type of proof supposing we want to prove some statement P. You want to prove some statement P. Then you assume that P is NOT true. Then from this by some argument you arrive at a contradiction something like that Q AND NOT Q. That is assuming P is NOT true or assuming NOT P you arrive at a statement of the form Q AND NOT Q that means your assumption is not correct that is NOT P is not true that is P is true. This sort of an argument is called proof by contradiction.

Let us take an example. Root 2 is not a rational number. We want to prove that root 2 is not a rational number, this is statement P. So you assume NOT P, what is NOT P? NOT P is root 2 is a rational number. Now, from this you show that NOT P implies another statement Q where Q denotes root 2 is P by Q where P and Q are integers and also relatively prime.

(Refer Slide Time: 19:16)



Prove P

Assume $\neg P$

$\vdots$

$Q \wedge \neg Q$

$P : \sqrt{2}$ is not a rational number

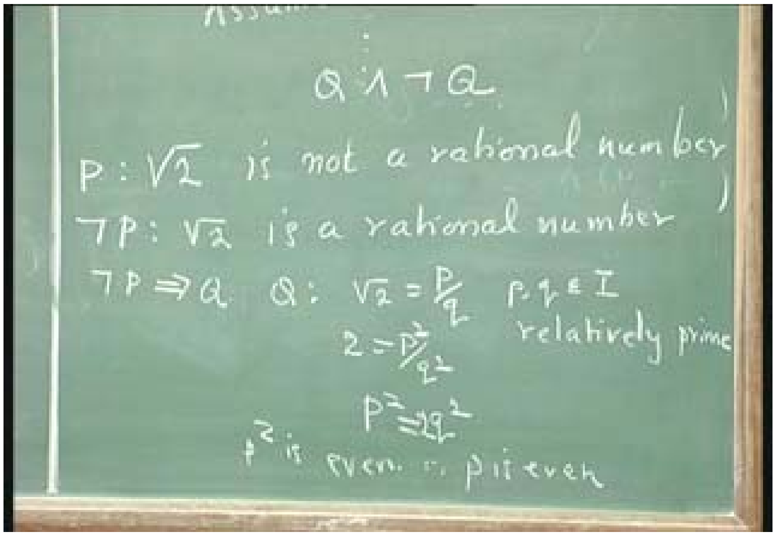
$\neg P : \sqrt{2}$ is a rational number

$\neg P \Rightarrow Q \quad Q : \sqrt{2} = \frac{p}{q} \quad p, q \in I$

relatively prime

What do you mean by relatively prime? P and Q do not have common factor. So you assume that root of 2 is equal to P by Q where P and Q do not have any common factor. So if root 2 is a rational number obviously it can be written in the form of an integer by an integer and so you get this statement NOT P implies Q. So what do you get? You get root 2 is equal to P by Q now squaring you will get 2 is equal to P square by Q square or P square is equal to 2Q square so P square is even which would mean P is even.

(Refer Slide Time: 20:27)



$\vdots$

$Q \wedge \neg Q$

$P : \sqrt{2}$ is not a rational number

$\neg P : \sqrt{2}$ is a rational number

$\neg P \Rightarrow Q \quad Q : \sqrt{2} = \frac{p}{q} \quad p, q \in I$

relatively prime

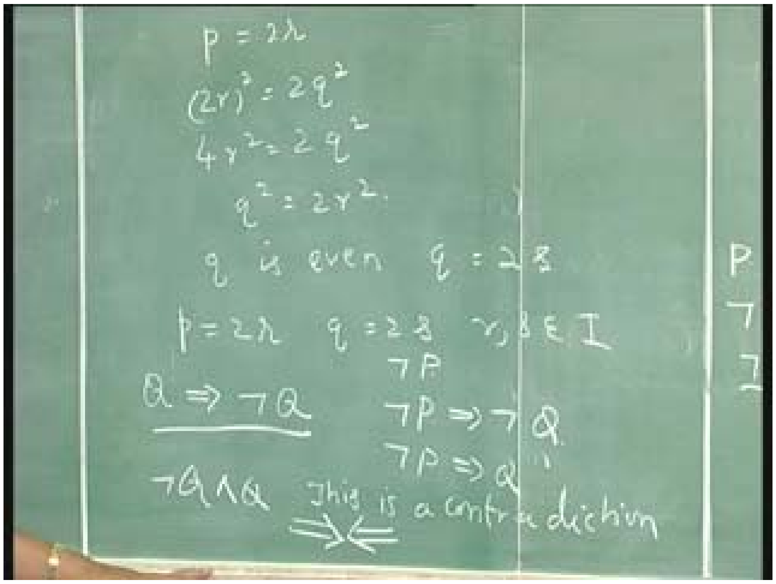
$2 = p^2/q^2$

$p^2 = 2q^2$

p^2 is even $\therefore$ p is even

So p is even means you can write it as 2r. So what do you get? You get (2r) square is equal to 2q square or 4r square is equal to 2q square or q square is equal to 2r square which would mean q is even.

(Refer Slide Time: 23:28)



So q square is even so q is even so q is of the form 2 of s. So p is of the form 2 of r and q is of the form 2 of s where r and s belong to a set of integers. That means p and q have a common factor 2 and not relatively prime. So from this actually starting with q, q is root 2 can be expressed in the form p by q where p and q are integers and relatively prime. From this you come to the conclusion that NOT Q that is you do not express root 2 as p and q where they do not have a common factor. They have a common factor you assume like that and you will come to the conclusion that they have a common factor and not relatively prime. So, from this and this using Hypothetical syllogism you will get NOT P implies NOT Q and from NOT P implies Q and Q implies NOT Q you will get NOT P implies NOT Q. So you have both NOT P implies Q and you also have NOT P implies NOT Q and you started with NOT P.

Thus, using Modes ponens to this you will get NOT Q and using Modes ponens to this you will get Q. And by addition you get NOT Q AND Q and this is a contradiction. Usually you denote it like this, you use this symbol for contradiction to show that you have arrived at a contradiction. So assuming NOT P you have arrived at a contradiction so the assumption is wrong or false NOT P is false so P is true. This is the way therefore P is true. This is the way you prove proof by contradiction. This is a method which is very commonly used, proof by contradiction is a very famous or very widely used method for proving results.

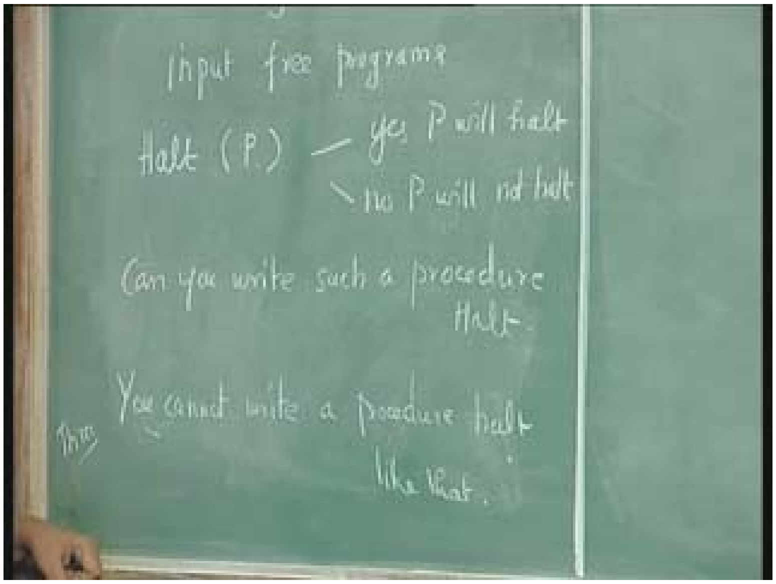
We shall take one more example. Let us consider one more example, this is a well known problem known as the halting problem. You know that if you do not have any restriction on the memory of the computer or for the time which a program can run, if you allow these two conditions then given a program ultimately will it halt or not? This is known as the halting problem. That is, if you have a computer program can you say whether it will halt or whether it will not halt.

We are not putting any restriction on the memory size, we are not putting any restriction on the time for which the program can run. So under those circumstances can you show whether it will halt or not. Or can you design an algorithm which will tell you whether it will halt or not.

Now whether a program will halt or not will depend upon the input it gets. So, for some input it may get into a loop or some input it may halt. So, testing a program whether will halt or not will depend on the input. So for simplicity we shall consider only input free programs. We will consider programs which are input free. So we are even restricting the problem a little bit. We are considering only input free programs and asking the question whether the program will halt or get into a loop.

Can you write a program halt that is in essence it is like this, can you write a program or a procedure halt for which the input is a computer program P. And this will tell you yes P will halt or it will tell you no P will not halt. So it can give you two answers yes P will halt or no P will not halt.

(Refer Slide Time: 27:35)



input free programs

Halt (P) — yes P will halt

\ no P will not halt

Can you write such a procedure

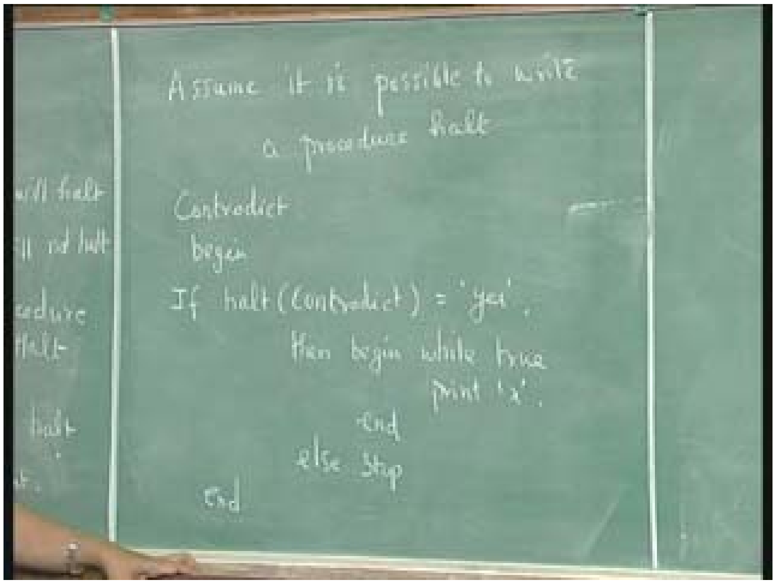
Halt.

Thm/ You cannot write a procedure halt

like that.

The question is can you write such a program such a procedure halt. The thing is you cannot write such a program halt that is what we want to prove. You cannot write a procedure halt like that, this is the theorem you want to prove. Now how would you again use proof by contradiction? Assume it is possible to write a procedure halt. Then you have to show that you are arriving at a contradiction. In that case you can write a program contradict as follows:

(Refer Slide Time: 29:08)

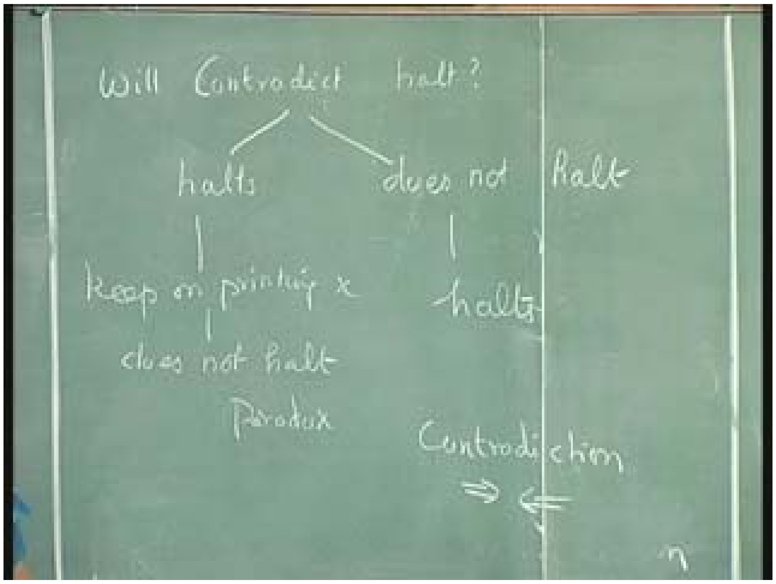


```
Assume it is possible to write
        a procedure halt

Contradict
 begin
If halt (Contradict) = 'yes'
        then begin while true
                       print 'x'.
                 end
        else stop
   end
```

You can write the procedure contradict like this; begin if halt of contradict equal to yes then begin while true print x end else stop then an end for this begin. You can write a procedure contradict in this manner. It says if halt of contradict is yes then begin while true print x else stop. Now you can note that contradict is an input free program there is no input for that it is straight away a straight program. Now will contradict halt or not. The question is will contradict halt or not? Will these procedures contradict halt? Look at this, if it halts there are two possibilities it halts or it does not halt. Let us see what happens here.

(Refer Slide Time: 32:20)



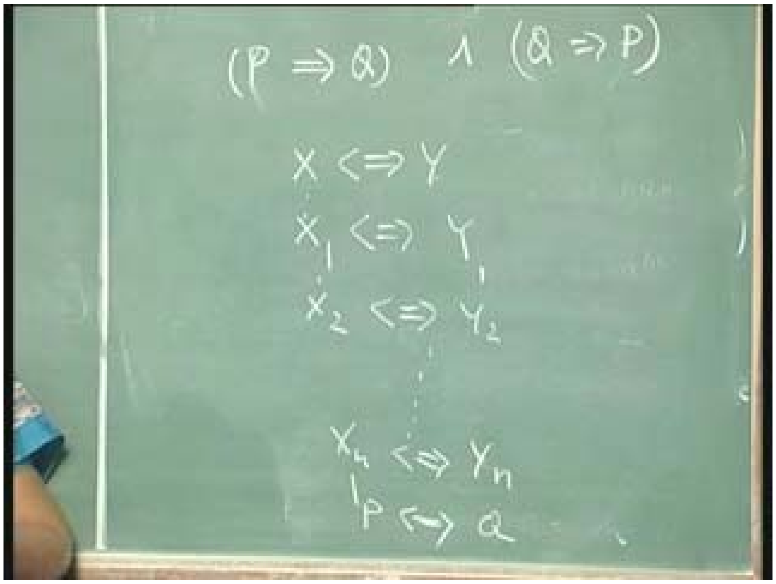
Supposing it halts then in this run this program halt of contradict should give you the yes answer because contradict halts. And in that case what happens this portion of the program will get executed that is you keep on printing x x x x like that it will never end. So keep on printing x that is it does not halt.

On the other hand, if contradict does not halt halt of contradict is not yes it is no so the else portion should be executed so it will stop, stop means it will halt. So if it does not halt you come to the conclusion that it halts. So the argument goes like this; if it is possible to write a procedure halt then it is possible to write a procedure contradict like this and what happens to contradict if it halts it does not halt if it does not halt it halts. This is something like a paradox, it is a paradox or it is a contradiction.

So assuming that it is possible to write a program like halt then you come to the contradiction and so the assumption that it is possible to write a program like halt is not correct it is not possible to write a program like halt or halt does not exist, a procedure like halt does not exist. This is the famous halting problem but instead of programs Turing defined it for Turing machines and even in 1936 he was able to see what is possible by a computer and what is not possible by a computer. And this halting problem or this is called undecidability. This undecidability of the halting problem is a major break through in Mathematics and theoretical Computer Science.

Once people came to know about this many other problems for which they did not have algorithms they realized that they are undecidable problems. An undecidable problem cannot have an algorithm. You cannot write an algorithm for an undecidable problem. It is a very strong statement it is not that you do not know how to write the program or somebody else does not know how to write the program, how to solve the algorithm or how to solve the problem or how to give an algorithm for the problem. It is just that you cannot have an algorithm for such undecidable problems. So let us continue with other types of proofs.

(Refer Slide Time: 35:45)



$(P \Rightarrow Q) \wedge (Q \Rightarrow P)$

$X \Leftrightarrow Y$

$X_1 \Leftrightarrow Y_1$

$X_2 \Leftrightarrow Y_2$

$X_n \Leftrightarrow Y_n$

$P \Leftrightarrow Q$

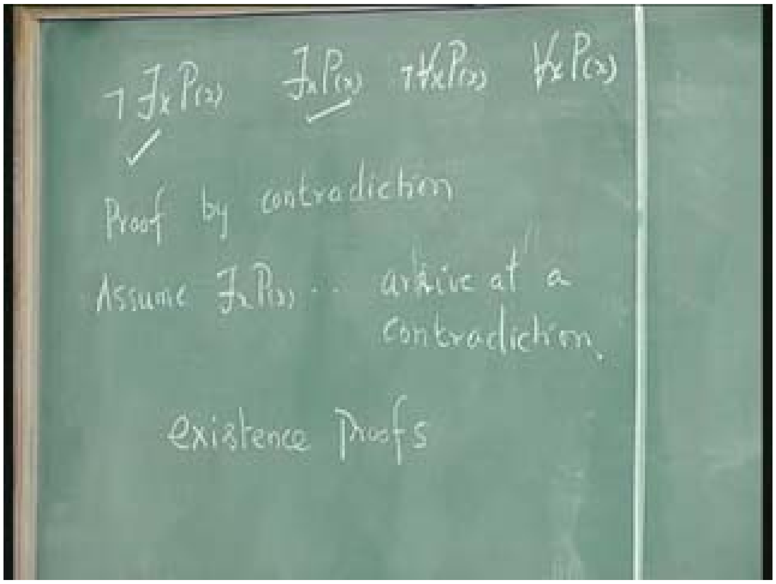
Now we have considered implications we also have statements of the form P is equivalent to Q. So P if and only if Q or P is a necessary and sufficient condition for Q. We will come across several statements of theorems which are of this form. Generally it is proved in this manner you prove it in two parts P implies Q AND Q implies P you prove it in two parts like this.

On rare occasions you also have another method of proving this, that is you start with some X is equivalent to Y some well known statement X is equivalent to Y and from this may be you write an equivalent statement here and an equivalent statement for Y here then another equivalent statement X_2 here another equivalent statement Y_2 here and like that finally after a few steps say n steps you get something like this X_n is equivalent to Y_n where this will represent P and this will represent Q. This is a rarely used method but it is also used in some cases. But the general way you prove an equivalent statement like this P is equivalent to Q or P if and only if Q is by proving P implies Q AND Q implies P. Or if you want to prove that P is a necessary and sufficient condition for Q, first you prove P is a sufficient condition and then you prove it is a necessary condition or the other way around.

So far we have considered statements which did not use any quantifiers. Next we shall consider the theorems, how to prove theorems which have quantifiers in their statements. When we use quantifiers the statement can be of this form; NOT there exist x P(x) or there exist x P(x) NOT for all of x P(x) or for all of x P(x) the statement of the theorem can be in any one of the four forms. Let us consider one by one. Now if you consider this statement to be of the form NOT there exist x P(x) usually for such theorems the proof is given by contradiction. So you assume there exist x P(x) and show that you arrive at a contradiction.

Now, if you consider the statements of form there exist x P(x) such proofs are existence proofs. You show the existence of something so they are very famous, the existence proofs are famous in several branches of Mathematics like differential equations and so on.

(Refer Slide Time: 38:10)



$\neg \exists_x P(x)$ ✓ $\exists_x P(x)$ $\neg \forall_x P(x)$ $\forall_x P(x)$

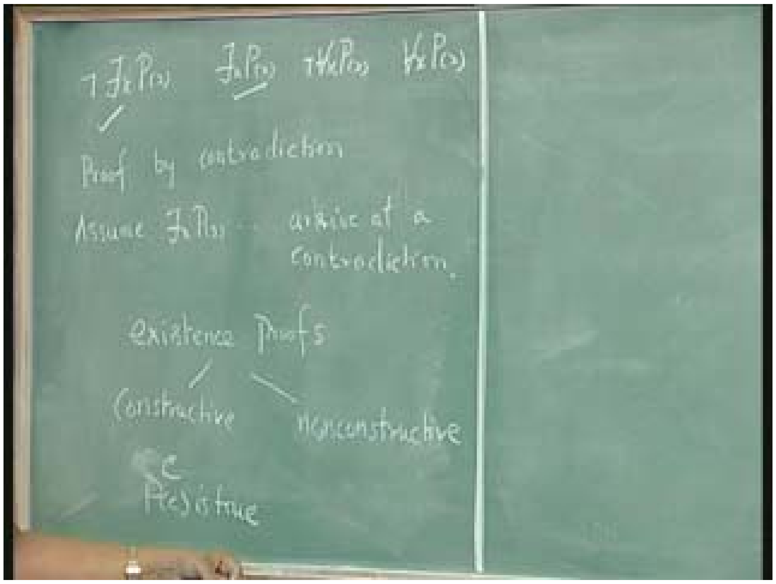
Proof by contradiction

Assume $\exists_x P(x)$ . . arrive at a contradiction.

existence Proofs

Here there are two types of proofs constructive and nonconstructive. A constructive proof gives you the element x. It tells you what is the element x or some element c for which P(c) is true. Sometimes it gives you the exact element c or sometimes it gives you an algorithm to find that element c or how to find that element c. Such proofs are called constructive proofs. In the nonconstructive proof no proof is given as to which c will satisfy the P(x) or which c will give you the value P(c). But by some other argument starting from something else you proof and so on. Such proofs are called nonconstructive proofs.

(Refer Slide Time: 39:35)

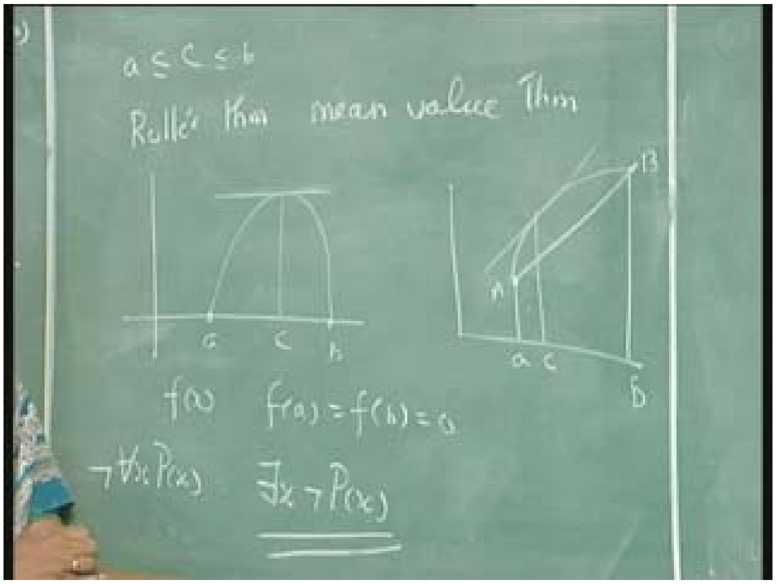

¬∃x P(x)   ∃x P(x)   ¬∀x P(x)   ∀x P(x)
Proof by contradiction
Assume ∃x P(x) ... arrive at a contradiction.
existence Proofs
Constructive
nonconstructive
c
P(c) is true

And usually there are some cases where you will not get the exact element c but some range in which you will get the element c. That is it will tell you that you can get c between some values a and b though it may not exactly give you the value c. Such proofs are very common. You would have know about Rolle’s theorem and Mean value theorem. I will not state the theorems but give a rare graphical expression.

Rolle’s theorem states that if you have a function f(x) and at point a and b that is f of a is equal to f(b) is equal to 0 and the function is continuous and differentiable then at some point you will get the tangent parallel to the x axis or at some point you will get a maxima or a minima point. This value c is not really given but it is between a and b that sort of a range is given.

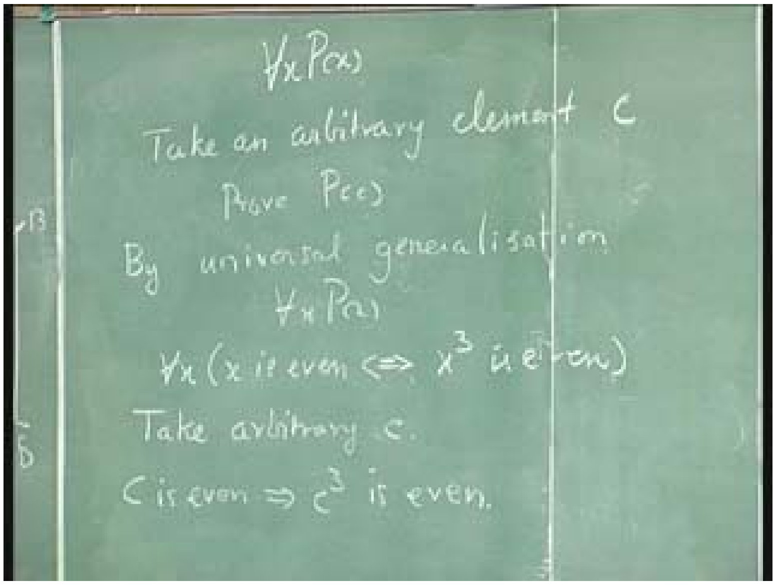
Similarly, Mean value theorem is graphically like this; if you have a point a and point b and the function is continuous and differentiable in this range then at some point c between a and b the tangent will be parallel to the line which is this line of the function value. I am just giving you the graphical interpretation of Rolle’s theorem and Mean value theorem. So these are examples of existence proofs where you make use of the statement of the form there exist x P(x). And if you have statements like NOT for all P(x) you know that NOT for all of x P(x) can be written in the form there exist x NOT of P(x) so if you bring the NOT inside for all will become there exist so this is equivalent to saying like this. And again now the statement is of the form there exist x P(x). So both constructive and nonconstructive proofs can be given.

(Refer Slide Time: 42:29)



Now, if you have statement of the theorem as for all of x P(x) usually what you take is, take an arbitrary element C of the universe and prove P(c). Then by Universal generalization you get for all of x P(x). Take an example, consider for all of x x is even is equivalent to saying x cubed is even. Now how do you prove that? Take arbitrary C and show that in two parts. This is equivalence so you have to show it in two parts. C is even implies C cube is even this is one part how will you prove that? You use direct proof C is equal to 2K if C is even you can write C is equal to 2K so C cube will be 8K cube where K is an integer so C cube is even. The other way around you have to prove C cube is even implies C is even.

(Refer Slide Time: 44:33)



$\forall x P(x)$

Take an arbitrary element c

Prove P(c)

By universal generalisation

$\forall x P(x)$

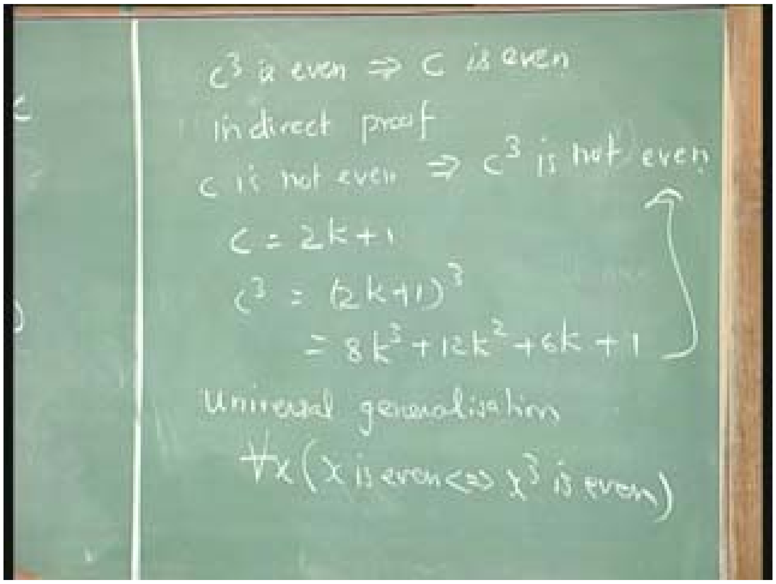
$\forall x$ (x is even $\Leftrightarrow$ x^3 is even)

Take arbitrary c.

c is even $\Rightarrow$ c^3 is even.

Now you use indirect proof here that is instead of showing this you show that C is not even implies C cube is not even that is C is equal to 2K plus 1 it is odd like this.

(Refer Slide Time: 46:03)



c^3 is even $\Rightarrow$ c is even

indirect proof

c is not even $\Rightarrow$ c^3 is not even

$c = 2k+1$

$c^3 = (2k+1)^3$

$= 8k^3 + 12k^2 + 6k + 1$

Universal generalisation

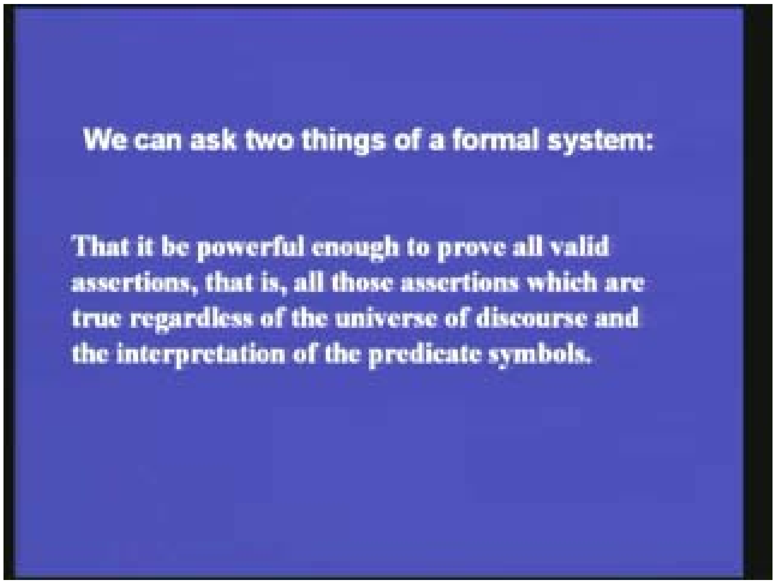
$\forall x$ (x is even $\Leftrightarrow$ x^3 is even)

So C cubed will be (2K plus 1) to the power 3 is equal to 8K to the power 3 plus 12K square plus 6K plus 1. So you can see that this is even this is even this is even so C cube is odd with that one so C^3 is not even. So you have proved that C cube is not even. Now, use Universal generalization because C is arbitrary you get for all of x x is even is equivalent to saying x cubed is even. This is the way we prove using quantifiers.

So we consider some methods of proofs, there are some other methods like proof by induction. And induction itself you have two types of induction weak induction and strong induction. We shall learn about that after we learn sets and how to define a set inductively. Some more methods are also there but these are the main methods of proofs and we have considered them.

Now when you have a formal system we ask two things about a formal system that it be powerful enough to prove all valid assertions, that is, all those assertions which are true regardless of the universe of discourse or the interpretation of the predicate symbols.

(Refer Slide Time: 46.44)



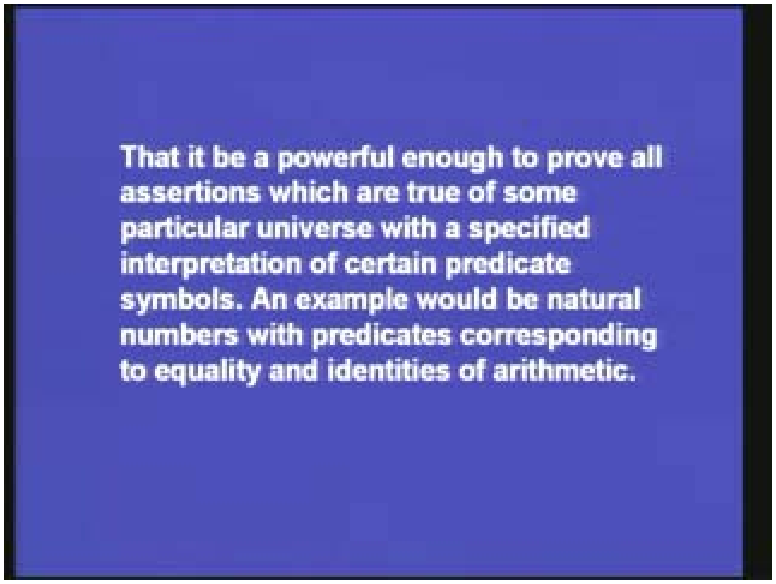
We can ask two things of a formal system:

That it be powerful enough to prove all valid assertions, that is, all those assertions which are true regardless of the universe of discourse and the interpretation of the predicate symbols.

We ask two questions; one is this; is it powerful enough to prove all valid assertions, all those assertions which are true regardless of the universe of discourse and the interpretation for the predicate symbols.

The second question is this; is the system powerful enough to prove all assertions which are true of some particular universe with the specified interpretation of certain predicate symbols. As an example you can consider natural numbers with predicate corresponding to equality and identities of arithmetic.

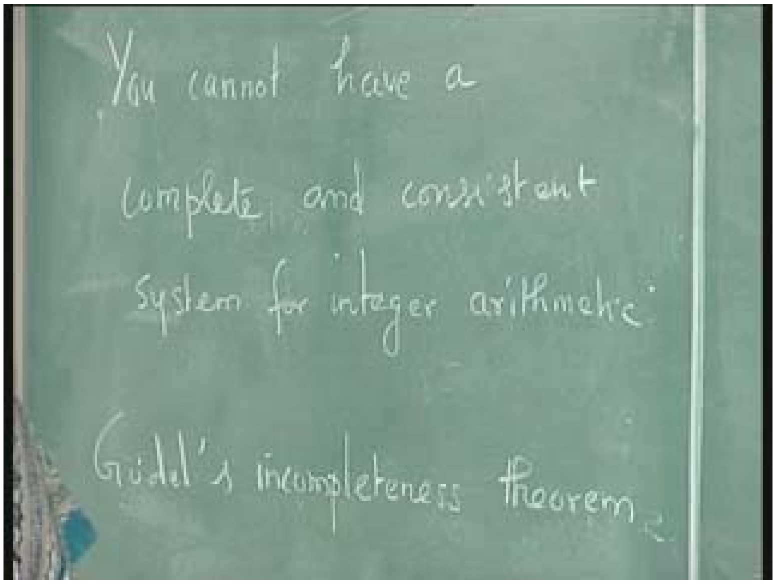
(Refer Slide Time: 47:43)



That it be a powerful enough to prove all assertions which are true of some particular universe with a specified interpretation of certain predicate symbols. An example would be natural numbers with predicates corresponding to equality and identities of arithmetic.

We have to say that logicians and mathematicians have been successful with regard to the first question or the first type of problem which is called usually you say where is a predicate or a well formed formula of predicate calculus is valid or not. In the case of propositional logic it can be decided in the case of predicate first order logic it is partially desirable and you can give an algorithm. Whereas mathematicians have not been very successful in dealing with problems of type two. And Gödel showed that this not being successful is rather inherent in the system itself. That is, you cannot be successful in dealing with such types of problems. What he proved is you cannot have a complete and consistent system for integer arithmetic. This is called Gödel's incompleteness theorem.

(Refer Slide Time: 50:10)



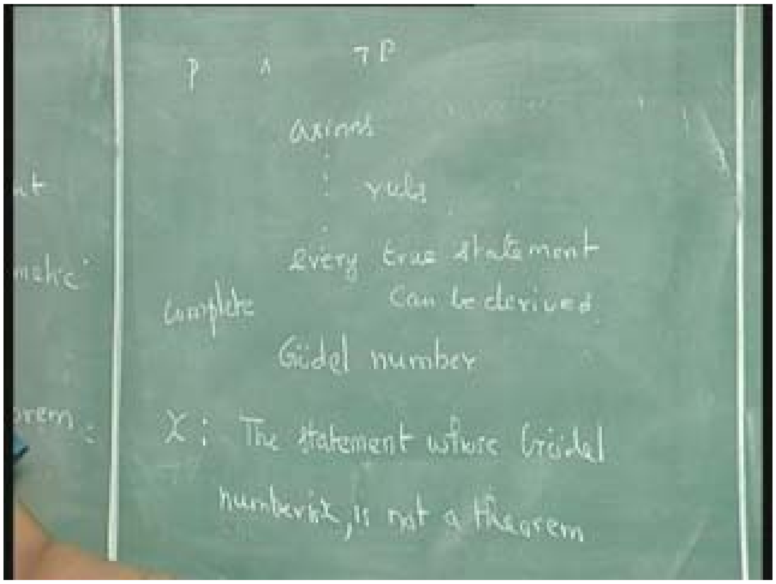
You cannot have a complete and consistent system for integer arithmetic:

Gödel's incompleteness theorem.

A consistent system is one in which you cannot prove both P and NOT P. In a system you prove both P and NOT P the system is inconsistent. A complete system is one in which you can prove every true statement. What you mean by proving? Starting from axioms using rules every true statement can be derived, such a system is called a complete system. What Gödel proved is, if a system is consistent it cannot be complete for integer arithmetic. For that he used what is known as a Gödel numbering. Each sentence can be given a number that number is a very large number and he used something like this; x, x is a number, it is a Gödel number, he used what is called a Gödel number. The statement of the theorem is like this; the statement whose Gödel number x is not a theorem. I am roughly writing this, the statement of the theorem was something like this.

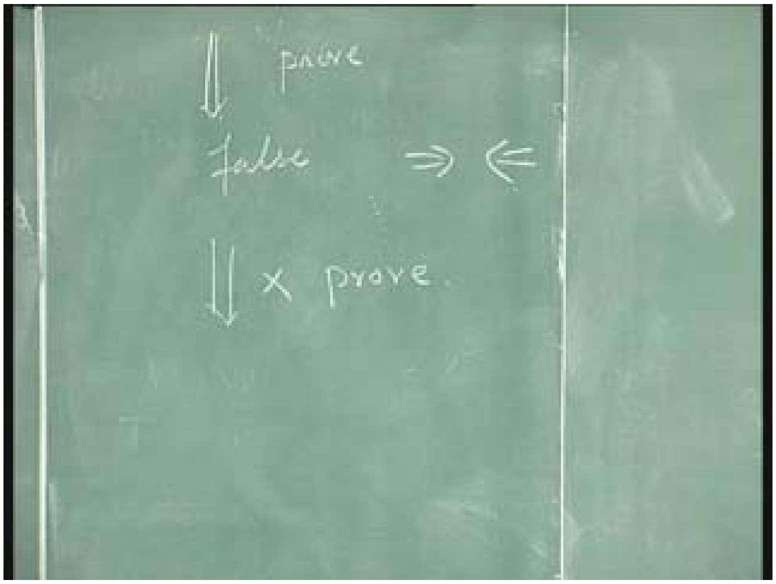
It said that the statement whose Gödel number is x is not a theorem.

(Refer Slide Time: 52:17)



What happens if starting from axiom if you are able to derive this particular statement then it is false saying that you cannot prove that it is false, if you are able to prove starting from axioms, this theorem this statement then it says that you cannot prove it, it is not a theorem, you cannot prove it starting from the axiom then it becomes false so you arrive at a contradiction. Now, suppose you are not able to prove this, starting from the axiom if you are not able to prove this statement then it is a true statement about the system which you cannot prove. So the system is incomplete.

(Refer Slide Time: 53:02)



It is a true statement but you will not be able to prove that statement so it is incomplete the system is incomplete. This is the essence of Gödel incompleteness theorem. Of course I am not going into the details of the way in which the Gödel number is chosen and how that particular statement of the theorem is written and so on. So Mathematicians have been successful in dealing with problems of type one and they have not been successful of problems of this type. And this is rather because of the inherent difficulty or this lack of success is inherent in the system itself.

We have considered some methods of proof which you already have studied in schools and so on. This is just to tell you how a formal method can be used to prove theorems and how you look at the formal way of proving theorems when the statements of the theorems are in particular forms.