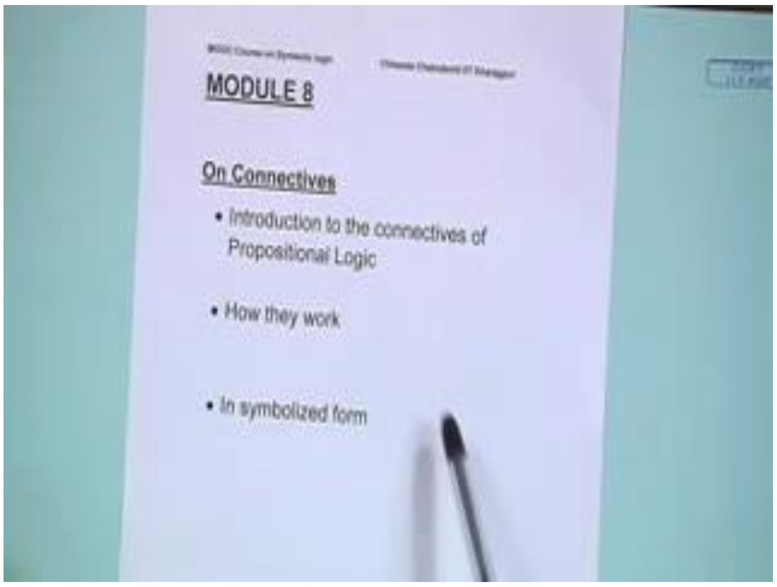
Symbolic Logic

Prof. Chhanda Chakraborti
Department of Humanities and Social Sciences
Indian Institute of Technology, Kharagpur

Lecture – 08
Connectives
Scope of Connectives

Hello! This is module 8 and we are going to talk about the connectives in propositional logic syntax as part of propositional logic syntax.

(Refer Slide Time: 00:26)



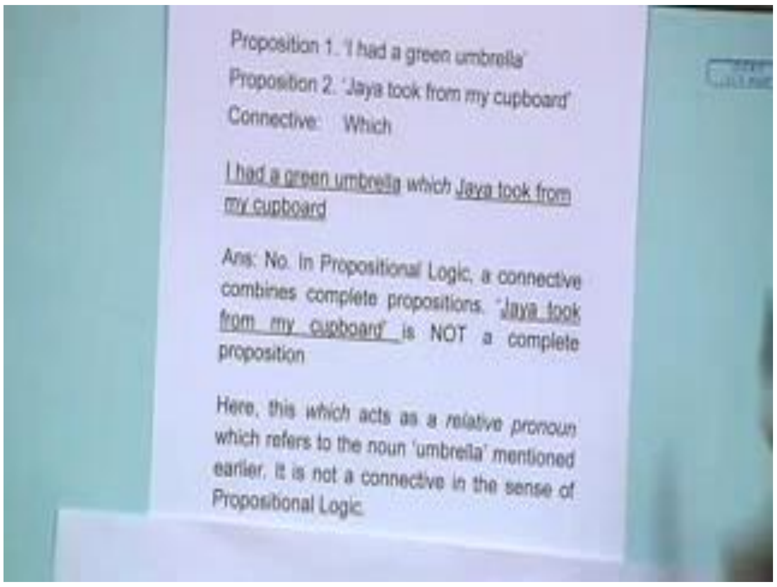
MODULE 8

On Connectives

- Introduction to the connectives of Propositional Logic
- How they work
- In symbolized form

So, we will be introduced to the connectives that they use, and specifically how they work and we will start the symbolization also in the symbolized form, how the connectives are to be represented in the system.

(Refer Slide Time: 00:59).



Proposition 1. 'I had a green umbrella'

Proposition 2. 'Jaya took from my cupboard'

Connective: Which

I had a green umbrella *which* Jaya took from my cupboard

Ans: No. In Propositional Logic, a connective combines complete propositions. 'Jaya took from my cupboard' is NOT a complete proposition

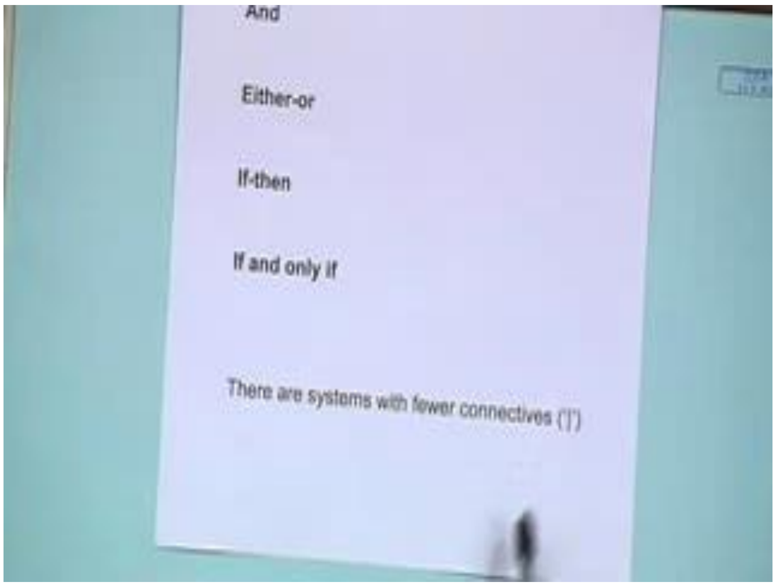
Here, this *which* acts as a *relative pronoun* which refers to the noun 'umbrella' mentioned earlier. It is not a connective in the sense of Propositional Logic.

First, the idea of a connective. See, when we …I have shown you some examples, but may be the concept needs be clarified further. What qualifies as a connective in propositional logic system? So let us consider this statement: 'I had a green umbrella which Jaya took from my cupboard'. This looks like a compound statement, where you might think that this is a connective. 'I had a green umbrella *which* Jaya took from my cupboard' and you will say there are two components. So is the "which" connective here? That's what we are trying to find out. Can we read the whole sentence, say , as a combination of two propositions, where the 'which' is acting as the connective? The answer is : No. If you are thinking in this kind of words, that we can probably analyze the whole sentence like so, and 'which' is a connective, the answer is clearly no. Why not?

First of all, understand that in propositional logic, a connective is supposed to combine complete propositions. So stand-alone propositions. And if you now look into the sentence itself, then you will find there one of the component is not even a complete proposition. 'Jaya took from my cupboard'… What Jaya took? 'Jaya took from my cupboard' is not even a complete proposition. So the way the connective works in propositional logic is that a complete proposition, a complete proposition and then the combination of these. So that's one of the reasons why 'which' here is not a connective.

The further thing that you then… you will say then what is this ‘which’? What is its function? Its function is as a relative pronoun. It’s a relative pronoun which refers to a noun, that has been previously used in the sentence, which in this case the umbrella. So ‘I had a green umbrella which Jaya took from my cupboard’. So the noun ‘umbrella’ is mentioned by, referred to by this ‘which’. Whatever it is, my point is that it is not a connective in the sense in which propositional logic connectives work, right? So with that introduction, let’s go into the proper discussion of the propositional logic connectives.

(Refer Slide Time: 03:51)



And

Either-or

If-then

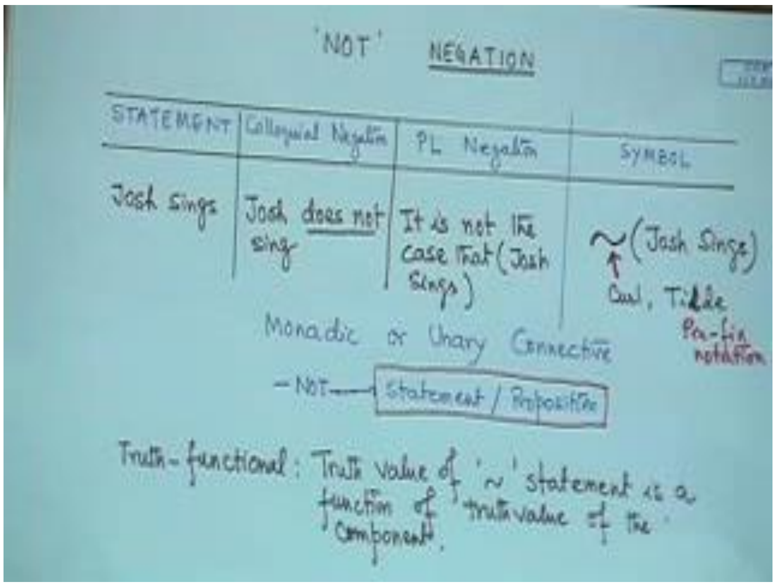
If and only if

There are systems with fewer connectives ('|')

As already discussed, propositional logic has five truth functional connectives. What are they? It’s important to also know that every logical system does not have to have five connectives, but this system does. This system that we are calling the PL, it has five truth functional connectives. What are those five truth functional connectives? Well, the first one we have already probably seen is ‘not’. And it is the only unary connective amongst these. ‘NOT’, that word. The other four are all binary or dyadic connectives. The other four are: ‘And’, then, ‘either-or’ and then, ‘if-then’ and also ‘if and only if’. So that is the complete set of five connectives that PL uses and each of them is a truth functional connective.

There can be, as I was trying to say, that there can be systems which have fewer connectives. Every system doesn’t have to have five connectives. For example, some systems have only one : That is stroke function. This is just to give you an idea about the diversity that exists among logical systems. But PL has these five, the ‘not’, the ‘and’, the ‘either- or’, the ‘if- then’ and ‘if and only if’. And we are going to now look closely into each of this and try to learn how they are going to be used in the system.

(Refer Slide Time: 05:48)



‘NOT’ <u>NEGATION</u>

STATEMENT	Colloquial Negation	PL Negation	SYMBOL
Josh sings	Josh does not sing	It is not the case that (Josh Sings)	~(Josh Sings)

Curl, Tilde

Pre-fix notation

Monadic or Unary Connective

– NOT ⟶ $\boxed{\text{Statement / Proposition}}$

Truth-functional : Truth value of ‘~’ statement is a function of truth value of the component.

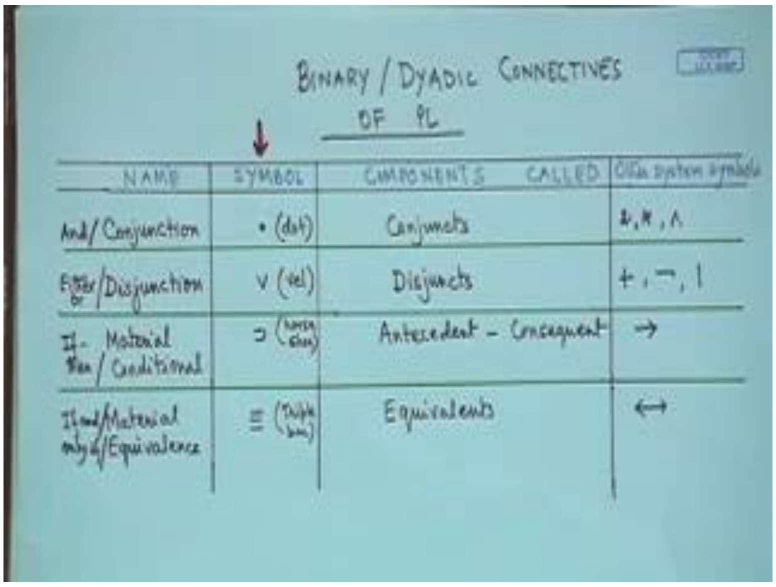
The first one is the NOT. The NOT that we are going to call negation also. So this NOT, how does it work? Let me give you an idea about that. See, this is a proposition ‘Josh Sings’. Colloquially, if we attach the ‘not’, we attach it with the verb. ‘Josh does not sing’. If this is the proposition or statement ‘Josh Sings’, its negation is ‘Josh does not sing’. Watch where the NOT is being attached colloquially. In ordinary statements, it attaches with the verb. That’s not how the PL negation works. So first thing to note is that, that PL negation works rather like this. It attaches itself in the beginning of the sentence or statement. So here is ‘Josh Sings’, and negation is ‘it is not the case that josh sings’. This is what we call the prefix notation. And that’s how it is going to be understood. Any negation is to be understood of the proposition itself.

Also appreciate the fact that this is a compound statement. It does not look like a compound, but if you remember the definition of a compound proposition, you will remember that here is a component. Compound propositions are those which have other propositions as its components. So here is the component, and attachment of the negation makes it into a compound. In symbol form, the symbol of the negation is this sign. What we call the Curl or the Tilde. Curl, or you can call it the Tilde. So here is 'Josh Sings', and to that the negation attaches in the beginning. That's how the syntax of PL works with negation. We called it prefix notation because it is attaching in the beginning of the proposition.

Let me also remind you that it is a monadic or unary connective. Please see that at a time it is picking up *only one* component. So it attaches itself to one component only. This is the kind of structure that you can remember. This is the schema how it works. The 'not' attaches in the beginning, and here comes the proposition. Here we have taken an example where the proposition in itself is simple, to that the negation is attached. But it is not necessary that it the component has to be always simple. It could be a complex proposition or compound proposition also. To that also when the negation attaches, it will attach itself at the beginning. That's how negation works in PL.

And then to remind ourselves that this negation is truth-functional in character. So the truth-value of a negation statement or a tilde statement is going to be a function of truth - value of the component itself. If you ask what is the truth-value of this, you need to know what is the value of this, and then compute the value of the negation. This is how the truth-functional NOT is going to work. So that is our first introduction to one of the very first connective, and its one of a kind because it is the only unary connective among the lot.

(Refer Slide Time: 09:43)



BINARY / DYADIC CONNECTIVES OF PL

NAME	SYMBOL	COMPONENTS CALLED	Other system symbols
And / Conjunction	• (dot)	Conjuncts	&, *, ∧
Either or / Disjunction	∨ (vel)	Disjuncts	+, ¬, |
If-then / Material / Conditional	⊃ (horse shoe)	Antecedent - Consequent	→
If and only if / Material / Equivalence	≡ (Triple bar)	Equivalents	↔

Next we are going to bring in before you the other connectives that we have. And these are, as I told you, they are all binary or dyadic connectives. We will go over them one by one. Binary or dyadic means there is going to be at most two components that we are going to see. The first one is ‘AND’. Its formal name is ‘conjunction’, as you can see *conjunction*. And the symbol by which it is represented that is called ‘dot’. It’s a simple ‘dot’. The components, because remember it is a connective, so there is going to be component, and the components are called *conjuncts*. You are going to have two conjuncts conjoined by this dot.

Now you may have seen this ‘AND’ or conjunction working in other logical systems or other programming language, and people may use various kinds of other symbols, but syntactically if you use any of this symbols in PL, it’s going to be a syntax error, which means that you have to learn, you have to orient yourself to learn to use this ‘dot’ when you want to express conjunction.

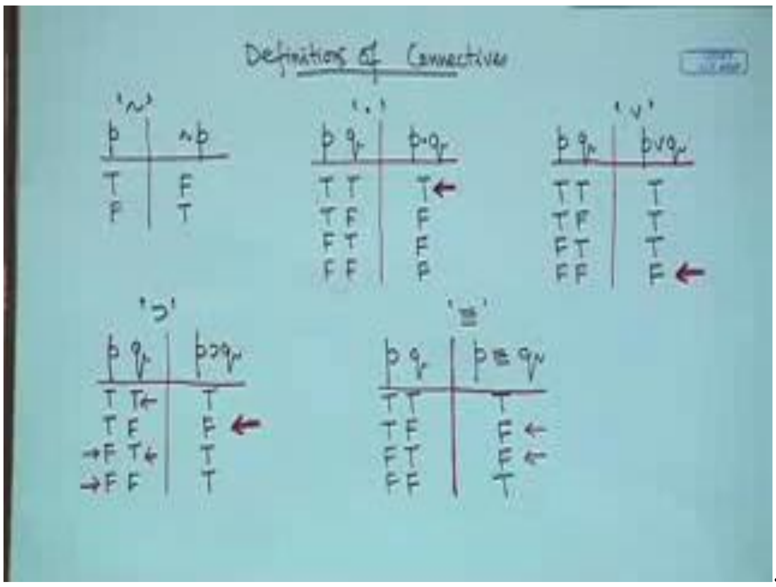
Similarly, we have ‘either-or’. The formal name for that is ‘disjunction’. And the symbol that we are going to use is called the *vel* or the *vee*. It’s a small ‘vee’. See ? This is the small ‘vee’, and the components we are going to be called *disjuncts*. You are going to have two disjuncts and the connective is going to be called the vel or the ‘vee’. Again, whatever other symbols you may have learnt in other systems, you are required to be using this ‘vee’ when you want to express disjunction or ‘either-or’.

Next is the ‘If-then’. The formal name for if-then in PL is ‘material conditional’. Material conditional. And the symbol by which it is represented looks like this ($\supset$). Please do not confuse it with the set symbol. It’s the other way. So it is an inverted C and it’s called *horseshoe*. You know the horses are… earlier days they used to wear iron shoes. So it’s called horseshoe. The components are going to be called ‘antecedent’ and ‘consequent’; which is which we will tell you later , when we go into details about the ‘if-then’. But the components are called… one of them is called ‘antecedent’; the other one is going to be called ‘consequent’. If you have been using so far the arrow symbol for the ‘if-then’, I suggest that you orient yourself to this, using this symbol the horseshoe symbol ($\supset$).

Finally, we come to the fifth connective. It is ‘if and only if’ which the formal name for in the system is *material equivalence*. Material equivalence. And the symbol looks like this. It’s called the triple bar, triple bar, and it is a triple bar. The components are going to be called equivalents and if you have been using double-sided arrow in this fashion again, I will remind you that please get acquainted with this triple bar.

So these are the symbols for the connectives that we are going to use and you need to learn fast to actually start using them when we see the opportunity to express this kind of connectives. Each of them, as… I do not have to remind you, but each of these are actually truth-functional in character and we are going to explain that in our next slide.

(Refer Slide Time: 14:09)



Definitions of Connectives

'~'

p	~p
T	F
F	T

'·'

p q	p·q
T T	T ←
T F	F
F T	F
F F	F

'v'

p q	pvq
T T	T
T F	T
F T	T
F F	F ←

'⊃'

p q	p⊃q
T T ←	T
T F ←	F ←
→ F T ←	T
→ F F	T

'≡'

p q	p≡q
T T	T
T F	F ←
F T	F ←
F F	T

.

We are going to now define the connectives and here you will see how the truth-functional character of these connectives are going to come through. So we will start with the very first one; namely, the ‘negation’. This is the meaning of the negation, where p is any given proposition, and remember there are only two truth-values possible. So p can be either true, or false, and when we attach the tilde to it then it returns the other value in your truth-values. So, if p is true, not-p is false; if p is false, not-p is true. So if you look into this, how do we know what is the value of not-p? The answer is that first you have to know the truth value of the component itself which makes the tilde a truth-functional connective. This is self-explanatory, but I suggest that you remember how the negation works.

Let’s now go into the next one; this is our ‘dot’, which you are going to call conjunction. Conjunction or dot is a dyadic connective, so it’s going to have two components, two conjuncts. And these are the truth values possible. So p has this kind of distribution, q has this kind of distribution, and note that p dot q is true *only when* both the conjuncts are true. Only when both the conjuncts are true. Otherwise in every other case it is false. And that is how the dot works. Once more, it is a truth-functional connective, as you can see, entirely determined by the truth-value of the components. So this is the definition of 'dot'. I strongly suggest that you get acquainted with this kind of meaning, the tabular sort of meaning of the connectives, because we are soon going to use them as we go along and there is… if you don't remember this, it will be very difficult for you to apply them there. So remember, the conjunction, as I have shown, true only when both conjuncts are true.

Next one is disjunction. So here you have again the two disjuncts, and this is the complete table for the vel or the 'vee' of the disjunction. What's happening here is that we find that the 'vee' or the disjunction is false only when both disjuncts are false. Otherwise, it is true in every other possibility. Ok? So normally we would say that 'either-or' works in this way that if I ask you: Will you have tea or coffee? And you are supposed to say : Either tea, or coffee. But here in this table, it seems like we are getting some other kind of picture. So here is tea, or here is coffee, and in both cases the value returned is true. But remember the 'vel' is also true when you say yes to both tea and coffee. So in other words, that was just a stray example, but the I guess the point goes through, is that this disjunction works in this way that it is true when either of the disjuncts is true, or when both the disjuncts also true. And it is false only when both of them are false.

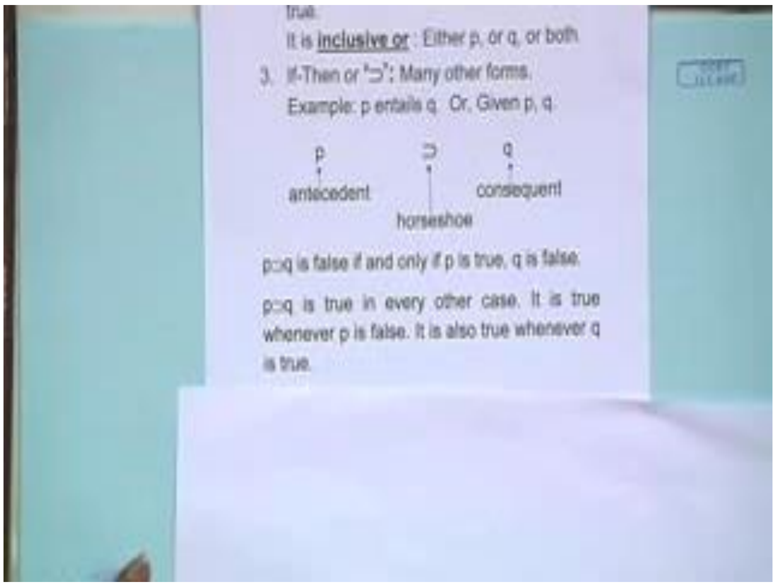
Let's now come to 'if-then'. This is our 'if-then' and the symbol that we were using is the horseshoe ($\supset$). So p, q and this is *if p then q*. And notice again when is it false. The only time p horseshoe is false when p is true and q is false. In every other case it is true. So 'if p then q' is false, when p is true and q is false. Otherwise in every other case it turns out to be true. That is something to remember. That is something very important to remember. Specifically I want to draw your attention to the last two rows. The third and the fourth row. See what is happening. p is false in both cases, and q is true in one case, false in one case. Look at the result, 'p horseshoe q' ($p \supset q$) is true nevertheless. So, from this we can gather that when p is false, it doesn't matter what value q has, 'p horseshoe q' will return the value true. Got it? So the last two rows sort of establish that.

When you have p as false, then it doesn’t matter what value q has, horseshoe will gain the value true. Similarly I would like to draw your attention to the first and the third row. See what is happening. q is true in both cases and p is true in one case, p is false in another. What do we learn from this? That when q is true, it doesn’t matter what value p has. In each case, ‘p horseshoe q’ will return the value true. So, from this, please read this table closely as you go along, but this is an important lesson to learn about the horseshoe how it behaves.

This is our last one and this is triple bar (≡). What is happening? Triple bar means equivalent. So, whenever the values match or equal, then p triple bar q assumes the value true. So when they are true, ‘p triple bar q’ is true. When they are false, ‘p triple bar q’ is true. Only when the value is mismatched; that is, one of them is true, one of them is false, that is when the ‘p triple bar q’ becomes false. Alright?

So this is our quick introduction to all the connectives as we see them.

(Refer Slide Time: 20:48)



true.

It is **inclusive or** : Either p, or q, or both

3. If-Then or ‘⊃’: Many other forms.

Example: p entails q. Or, Given p, q

p
⊃
q
antecedent
horseshoe
consequent

p⊃q is false if and only if p is true, q is false.

p⊃q is true in every other case. It is true whenever p is false. It is also true whenever q is true.

And, as I told you, I suggest that you please remember to use this as often as possible, as a reminder because we are going to soon use this as we go along. So, from this what we have learnt , we will quickly summarize, that first of all that this is our conjunction and something to remember that we have understood how the ‘dot’ works. But this is also to remind you that many words may express the idea of a conjunction without using the ‘AND’. So I remind you to more or less to tell.. call this connective conjunction. The

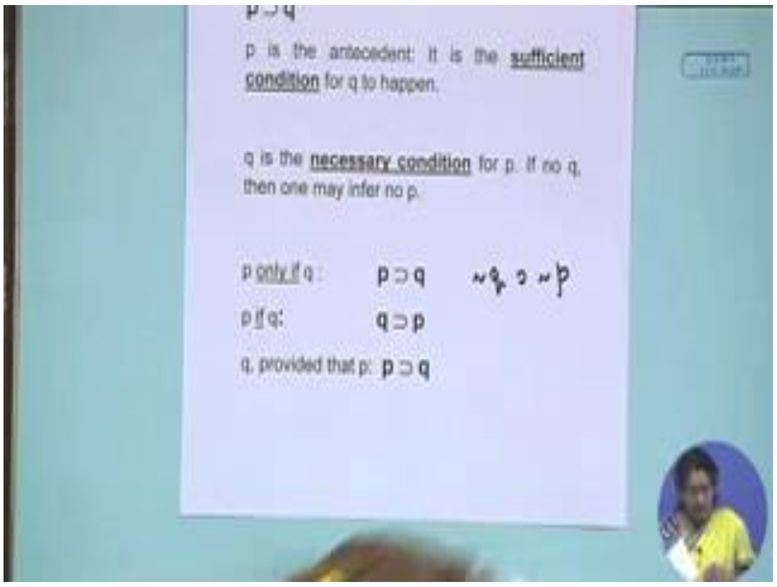
‘AND’ fixation is not necessary. For example, you may have ‘p, but q’. That is a conjunction, but the word is not ‘and’. It expresses the sense of conjunction. Say, ‘in addition to p, q’. That’s a conjunction, but it does not use the word ‘and’.

Regarding the disjunction or the ‘vel’, what we have learnt is that it is not the XOR or the ‘exclusive or’. Why? Because, it is true even when both the disjuncts are true , so that something to also put a flag on. Regarding the ‘if-then’, as I have tried to tell you, there can be many other forms. You may not use the, only the ‘if then’, but you may us, for example, ‘p entails q’. That’s a clear horseshoe. ‘p entails q’, but ‘if-then’ is not used. The word ‘if-then’ is not used. ‘Given p, q’. Again ‘if-then’ is not used. But it’s a conditional statement.

Now, as I was telling, one of them is called antecedent the other one is called consequent. This is if p then q, ‘p horseshoe q’ ($p \supset q$). What comes after the ‘if’ and precedes the “then”, that is what we call the ‘antecedent’. In this case, p is the antecedent and q is called consequent. Something that follows the ‘then’, that is called consequent. And the sign, as you know, is called ‘horseshoe’ .The position is very important. Ok? From now on, I am going to refer to them as antecedent, consequent.

Let’s take a look into various forms. But before that just reminder, that ‘p horseshoe q’ ($p \supset q$) is false only when p is true and q is false. Otherwise, in every other case it is going to be true. As we have already discussed.

(Refer Slide Time: 23:15)



$p \supset q$

p is the antecedent: It is the <u>sufficient condition</u> for q to happen.

q is the <u>necessary condition</u> for p. If no q, then one may infer no p.

p <u>only if</u> q:	$p \supset q$	$\sim q \supset \sim p$
p <u>if</u> q:	$q \supset p$	
q, provided that p:	$p \supset q$	

The symbolization, slowly we are getting into the symbolization, so, here is a simple, sort of a quick way to understand this. That this sign as to be read as 'if p then q'; symbolized as like this $p \supset q$. So, 'if I push the switch then the light will be on'. Ok? So, 'p horseshoe q' $p \supset q$.

When we say that p is the antecedent, the other name for the antecedent is *sufficient condition*, which means that it by itself is sufficient to bring q about. It may not be the only condition under which q happens, but it is sufficient. If you see p, that q is going to be there. The other name of antecedent is sufficient condition.

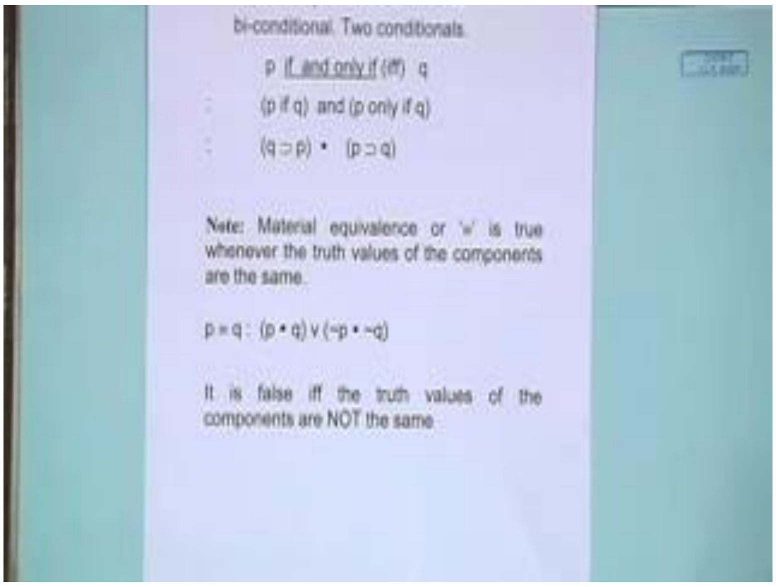
And the consequent is known as *necessary condition*, necessary condition for p. So, the way to understand necessary condition is like this, if there is no q, then one may infer no p either. So remember these two terminologies. That yes, this is antecedent, this is consequent. The other name to know them by that this antecedent works as the sufficient condition for consequent, and the consequent works as the necessary condition for the antecedent.

Now, let's take the various forms, 'p only if q'. 'p only if q'. When we say that, what you are saying is that the q is necessary condition; 'p only if q'. So q does not happen, p does not happen; and the way to represent that would be this 'p horseshoe q' $p \supset q$. You might say that I can also probably write like this: 'not- q horseshoe not- p', $\sim q \supset \sim p$. And you are right! But these two are equivalent. We will learn that when we go along, but right now as I say you all are beginners, so p only if q, remember, this is the way to represent that.

'p if q', which means that q is sufficient condition, ' if' this, this is the antecedent. Therefore, the symbolization will be 'q horseshoe p'.

'q provided that p'; 'provided that p' means this is your antecedent; this is sufficient, 'p horseshoe q'.

(Refer Slide Time: 25:54)



bi-conditional. Two conditionals.

p <u>if and only if</u> (iff) q

: (p if q) and (p only if q)

: $(q \supset p) \cdot (p \supset q)$

Note: Material equivalence or '≡' is true whenever the truth values of the components are the same.

$p \equiv q$: $(p \cdot q) \vee (\sim p \cdot \sim q)$

It is false iff the truth values of the components are NOT the same.

Now these are some things. It is not important to memorize, but more important that you understand this.

And then we come towards the triple bar (≡). The triple bar is known also as 'bi-conditional', so two conditionals are actually packed inside the triple bar. How? Let us take a look. See, p *if and only if* q, right? So if we break that up, then we find what we are seeing is *p if q* and *p only if q*. Once more, material equivalence we express it as 'p if and only if q' , right? And the short form of 'if and only if' is this *I double f* ' iff' . Now if we break it up then it is 'p if q' and 'p only if q'. What is the translation of that? p if q, we just learnt is going to be *q horseshoe p*. And p only if q we just learnt is p horseshoe q. Do you see the two conditionals? So in a way material equivalence is a conjunction of two horseshoe statements, which is why we call it Bi-conditional. We also noted that the material equivalence is true whenever the truth-values are same. So from that, see whether you can read this, from the meaning of triple bar we gain this that 'p triple bar q' either when p and q both are true, or when p and q both are false. That is not-p and not-q are true. So these are the expressive ways to understand 'p triple bar q'. The only time ' p triple bar q' is false when the truth-values do not match. This is how far I will go with this module. We are finished introducing you to the connectives. Please follow these meanings of the connectives because from now on that this is what we are going to use.

Thank you very much.