## LECTURE 7: PROPOSITIONAL LOGIC (1)

Software Engineering Mike Wooldridge

## 1 What is a Logic?

- When most people say 'logic', they mean either *propositional logic* or *first-order predicate logic*.
- However, the precise definition is quite broad, and literally hundreds of logics have been studied by philosophers, computer scientists and mathematicians.
- Any 'formal system' can be considered a logic if it has:
  - a well-defined syntax;
  - a well-defined *semantics*; and
  - a well-defined *proof-theory*.

- The *syntax* of a logic defines the syntactically acceptable objects of the language, which are properly called *well-formed formulae* (wff). (We shall just call them formulae.)
- The *semantics* of a logic associate each formula with a *meaning*.
- The *proof theory* is concerned with manipulating formulae according to certain rules.



- The simplest, and most abstract logic we can study is called *propositional logic*.
- **Definition:** A *proposition* is a statement that can be either *true* or *false;* it must be one or the other, and it cannot be both.
- EXAMPLES. The following are propositions:
  - the reactor is on;
  - the wing-flaps are up;
  - John Major is prime minister.

whereas the following are not:

- are you going out somewhere?
- -2+3

• It is possible to determine whether any given statement is a proposition by prefixing it with:

It is true that ...

and seeing whether the result makes grammatical sense.

- We now define *atomic* propositions. Intuitively, these are the set of smallest propositions.
- **Definition:** An *atomic proposition* is one whose truth or falsity does not depend on the truth or falsity of any other proposition.
- So all the above propositions are atomic.

• Now, rather than write out propositions in full, we will abbreviate them by using *propositional variables*.

• It is standard practice to use the lower-case roman letters

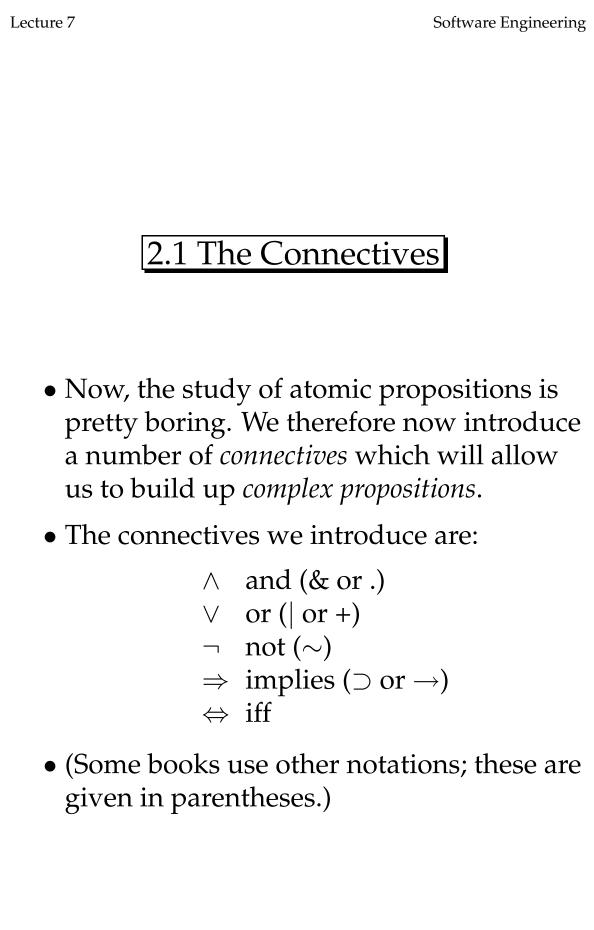
 $p,q,r,\ldots$ 

to stand for propositions.

• If we do this, we must define what we mean by writing something like:

Let *p* be John Major is prime Minister.

• Another alternative is to write something like *reactor\_is\_on*, so that the interpretation of the propositional variable becomes obvious.

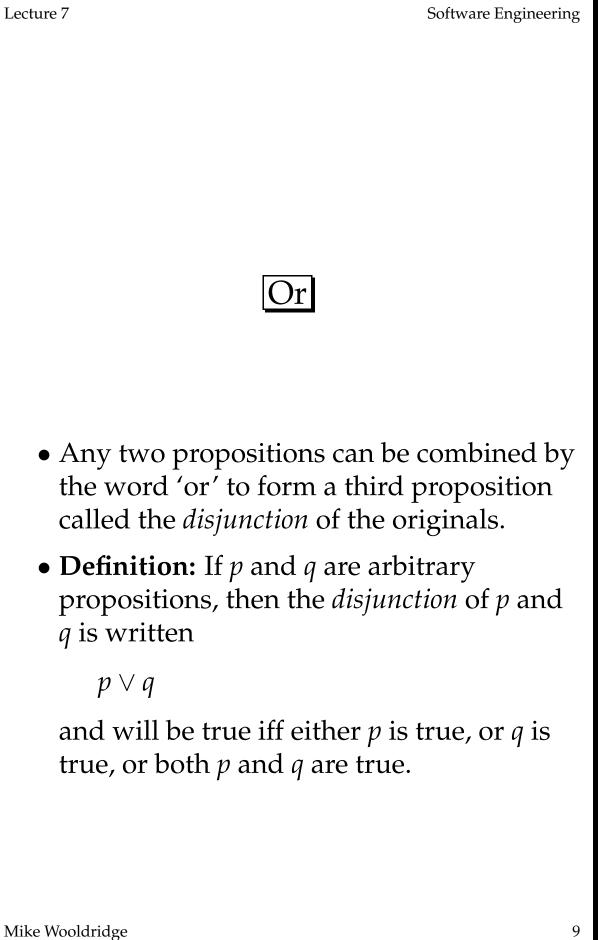


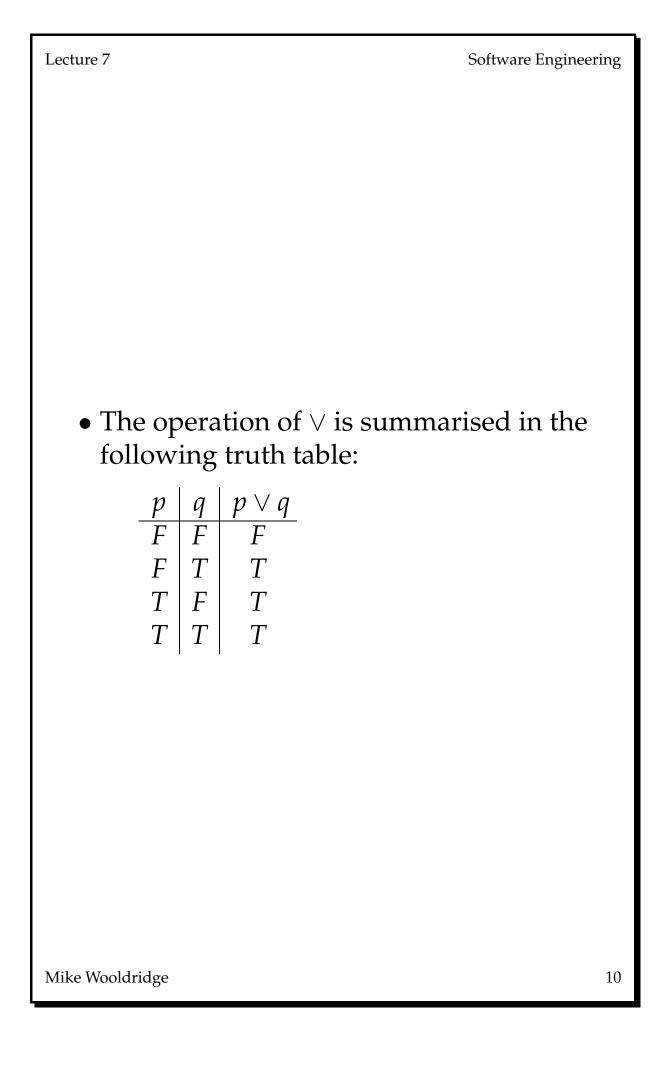
Lecture 7 Software Engineering And • Any two propositions can be combined to form a third proposition called the *conjunction* of the original propositions. • **Definition:** If *p* and *q* are arbitrary propositions, then the *conjunction* of *p* and *q* is written  $p \wedge q$ and will be true iff both *p* and *q* are true.

Lecture 7

- We can summarise the operation of ∧ in a *truth table*. The idea of a truth table for some formula is that it describes the behaviour of a formula under all possible interpretations of the primitive propositions the are included in the formula.
- If there are *n* different atomic propositions in some formula, then there are 2<sup>n</sup> different lines in the truth table for that formula. (This is because each proposition can take one 1 of 2 values *true* or *false*.)
- Let us write *T* for truth, and *F* for falsity. Then the truth table for  $p \land q$  is:

$$\begin{array}{c|c} p & q & p \land q \\ \hline F & F & F \\ F & T & F \\ T & F & F \\ T & T & T \end{array}$$





Lecture 7	Software Engineering
If Then	
• Many statements, partic	ularly in
mathematics, are of the	
<i>if</i> p <i>is true then</i>	q is true.
Another way of saying t write:	the same thing is to
p <i>implies</i> q.	
• In propositional logic, we connective that combined into a new proposition of <i>conditional</i> , or <i>implication</i> that attempts to capture a statement.	es two propositions called the a of the originals,
Mike Wooldridge	11

• **Definition:** If *p* and *q* are arbitrary propositions, then the *conditional* of *p* and *q* is written

 $p \Rightarrow q$ 

Lecture 7

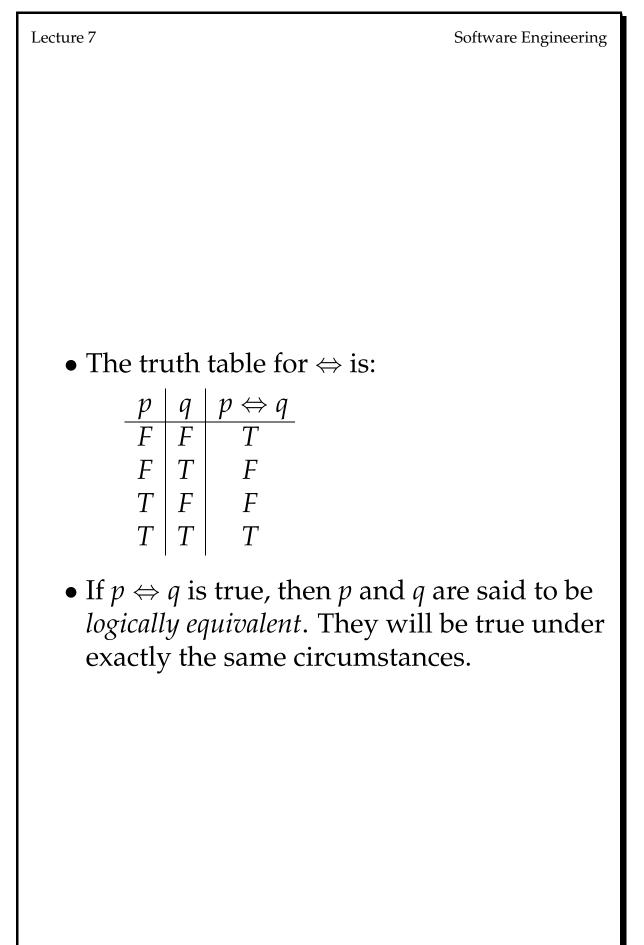
and will be true iff either *p* is false or *q* is true.

• The truth table for  $\Rightarrow$  is:

$$\begin{array}{c|cc} p & q & p \Rightarrow q \\ \hline F & F & T \\ F & T & T \\ T & F & F \\ T & T & T \end{array}$$

Lecture 7 Software Engineering • The  $\Rightarrow$  operator is the hardest to understand of the operators we have considered so far, and yet it is extremely important. • If you find it difficult to understand, just remember that the  $p \Rightarrow q$  means 'if *p* is true, then *q* is true'. If *p* is false, then we don't care about *q*, and by default, make  $p \Rightarrow q$  evaluate to *T* in this case. • Terminology: if  $\phi$  is the formula  $p \Rightarrow q$ , then *p* is the *antecedent* of  $\phi$  and *q* is the consequent.

Lecture 7 Iff • Another common form of statement in maths is: p is true if, and only if, q is true. • The sense of such statements is captured using the *biconditional* operator. • **Definition:** If *p* and *q* are arbitrary propositions, then the *biconditional* of *p* and *q* is written:  $p \Leftrightarrow q$ and will be true iff either: 1. *p* and *q* are both true; or 2. *p* and *q* are both false.



```
Lecture 7
```



- All of the connectives we have considered so far have been *binary*: they have taken *two* arguments.
- The final connective we consider here is *unary*. It only takes *one* argument.
- Any proposition can be prefixed by the word 'not' to form a second proposition called the *negation* of the original.
- **Definition:** If *p* is an arbitrary proposition then the *negation* of *p* is written

 $\neg p$ 

and will be true iff *p* is false.

• Truth table for  $\neg$ :

$$\begin{array}{c|c} p & \neg p \\ \hline F & T \\ T & F \end{array}$$

## Comments

- We can *nest* complex formulae as deeply as we want.
- We can use *parentheses* i.e., ),(, to *disambiguate* formulae.
- EXAMPLES. If *p*, *q*, *r*, *s* and *t* are atomic propositions, then all of the following are formulae:

$$p \land q \Rightarrow r$$
  

$$p \land (q \Rightarrow r)$$
  

$$(p \land (q \Rightarrow r)) \lor s$$
  

$$((p \land (q \Rightarrow r)) \lor s) \land t$$

whereas none of the following is:

$$p \land p \land q$$
  
 $p \land q$   
 $p \neg$ 



- Given a particular formula, can you tell if it is true or not?
- No you usually need to know the truth values of the component atomic propositions in order to be able to tell whether a formula is true.
- **Definition:** A *valuation* is a function which assigns a truth value to each primitive proposition.
- In Modula-2, we might write:

```
PROCEDURE Val(p : AtomicProp):
    BOOLEAN;
```

• Given a valuation, we can say for any formula whether it is true or false.

Software Engineering

• EXAMPLE. Suppose we have a valuation *v*, such that:

$$v(p) = F$$
  

$$v(q) = T$$
  

$$v(r) = F$$

Then we truth value of  $(p \lor q) \Rightarrow r$  is evaluated by:

$$(v(p) \lor v(q)) \Rightarrow v(r)$$
 (1)

$$= (F \lor T) \Rightarrow F \tag{2}$$

$$=T \Rightarrow F$$
 (3)

$$=F$$
 (4)

Line (3) is justified since we know that  $F \lor T = T$ .

Line (4) is justified since  $T \Rightarrow F = F$ .

If you can't see this, look at the truth tables for  $\lor$  and  $\Rightarrow$ .

- When we consider formulae in terms of interpretations, it turns out that some have interesting properties.
- Definition:
  - 1. A formula is a *tautology* iff it is true under *every* valuation;
  - 2. A formula is *consistent* iff it is true under *at least one* valuation;
  - 3. A formula is *inconsistent* iff it is not made true under *any* valuation.
- Now, each line in the truth table of a formula correponds to a valuation.
- So, we can use truth tables to determine whether or not formulae are tautologies.