# **Section 6.1 How do we reason**

# **Section 6.2 Propositional Calculus**

**Basic Logic**

It’s thinking clearly about making logical arguments; it boils down to AND, OR, NOT, like Java.

**Propositional Calculus**

Calculus:

* Language of expressions
* Each expression has a value
* There are rules to transform one expression into another that has the same value

Propositions:

* Expressions that are true or false

**Syntax of Propositional Calculus**

We represent propositions by formulas called well-formed formulas (WFFS) or “woofs”. It is also a grammatically correct expression.

WFF alphabet (a WFF is a…):

* Truth symbols: T (True) & F (False)
* Propositional variables: uppercase letters (P, Q, R, etc.)
* Connectives (operators):

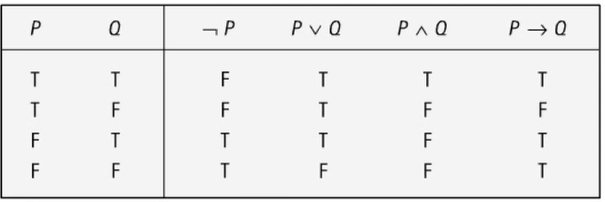
￢ (not, negation) (1 WFF)

∧ (and, conjunction) (2 WFFs)

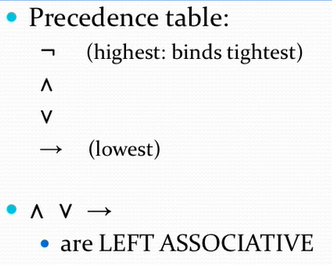
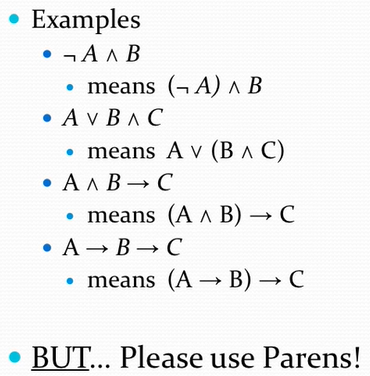
∨ (or, disjunction) (2 WFFs)

⟶ (conditional, implication) (1 WFF from another)

* Parentheses symbols: ( and ) (WFF surrounded by parens)



Truth table for WFF operators:



Precedence table / associatives:

## 

## 

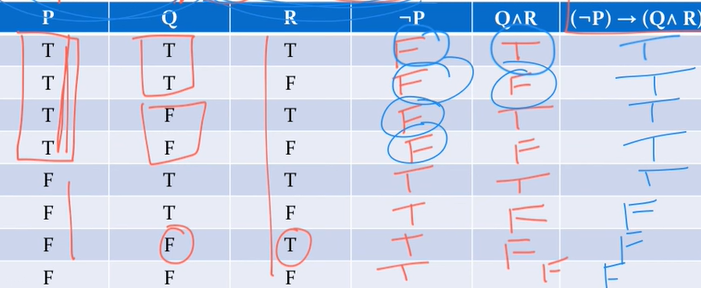
**Question:** Is P ∧ Q ∨ R a WFF?

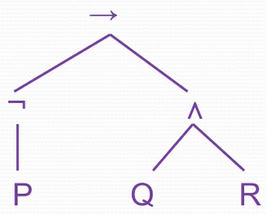
Before we even start to answer the question, we should probably parenthesize the “and” and the “or”. So, use the precedence table, so know that ^ (and) has a higher precedence.

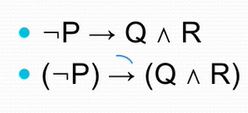
Thus, the new question is, is (P ∧ Q) ∨ R a WFF?

Let’s step through it and give justifications based on the definitions.

* Step 1: P is a WFF (prop var)
* Step 2: Q is a WFF (prop var)
* Step 3: R is a WFF (prop var)
* Step 4: P ∧ Q is a WFF (conj)
* Step 5: (P ∧ Q) is a WFF (paren)
* Step 6: (P ∧ Q) ∨ R is a WFF (disj)

Every WFF has a unique syntax tree and truth table due to the precedence table. Once you know the order of operations, you can draw a syntax tree / truth table. For example:





**Definitions**

A WFF is a tautology if its truth table values are all true.

A WFF is a contradiction if its truth table values are all false.

A WFF is a contingency if it has both true and false values.

## **Logical Equivalence**

Two WFFs, A and B, are logically equivalent (or equivalent) (A ≡ B) if they have the same truth value for each assignment of truth values to the set of all propositional variables occurring in the WFFs.

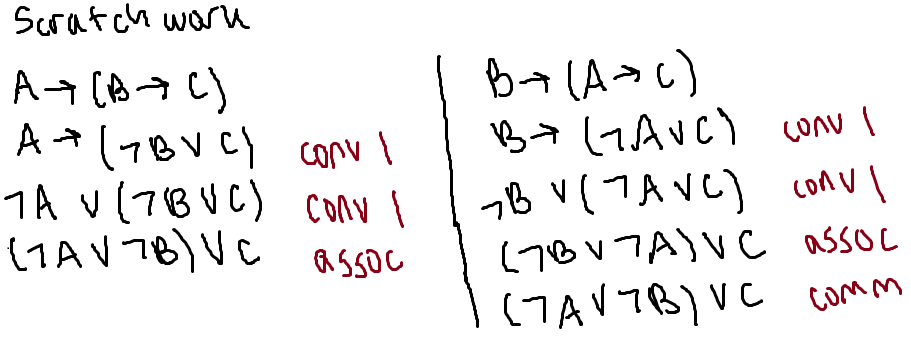
Two WFFs don’t need to use the same variables to be equivalent.

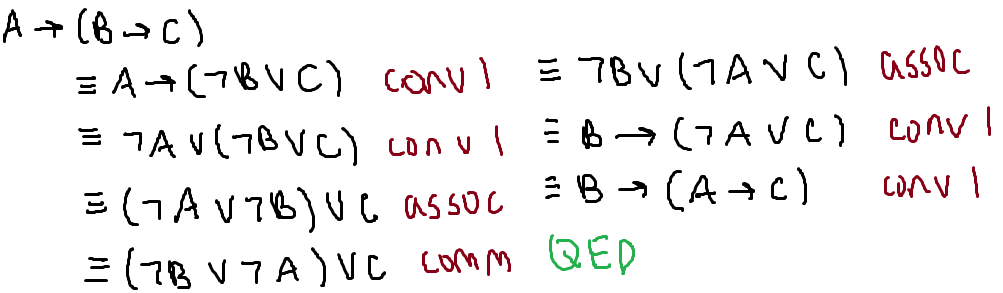
Another way to define equivalence is if and only if (iff) you can prove A ⟶ B and B ⟶ A. Remember that ≡ is an equivalence relation:

* Reflexive: R ≡ R
* Symmetric: If R ≡ S, then S ≡ R
* Transitive: If R ≡ S and S ≡ T, then R ≡ T

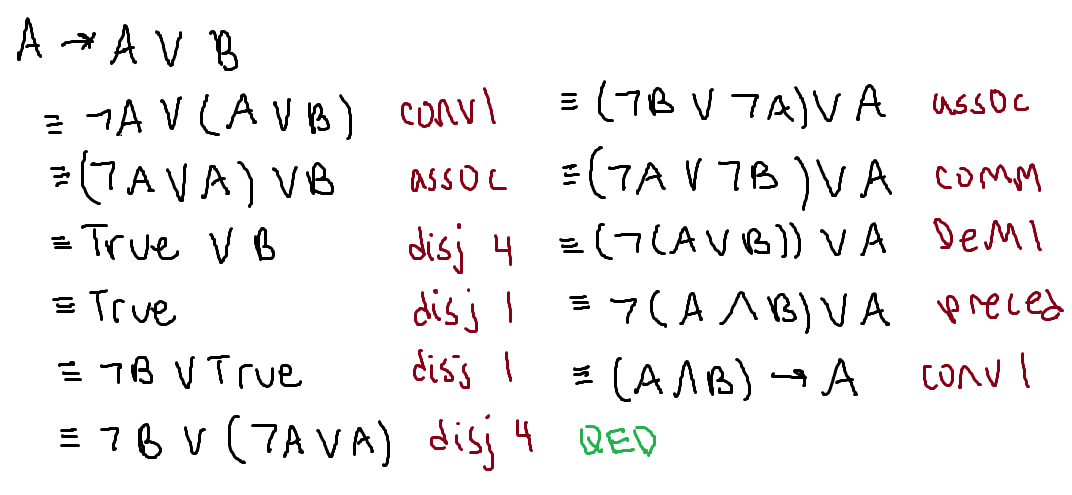
**Problem:** Prove that A ⟶ (B ⟶ C) ≡ B ⟶ (A ⟶ C).

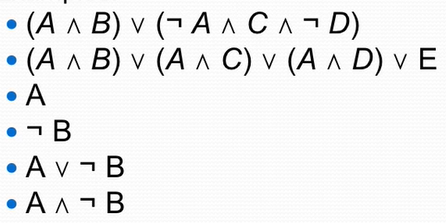
You could use truth tables or equivalency rules.





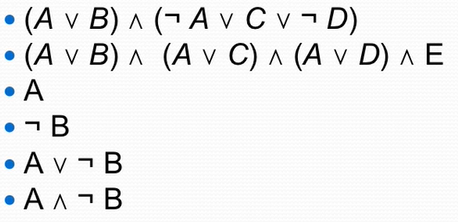
**Problem:** Prove that A ⟶ A ∨ B ≡ A ∧ B ⟶ A.





**Another Thing**

A literal is either a propositional variable or its negation. Examples include A, ¬A

Disjunctive normal form (DNF) is a WFF of the form C1∨ or ∨ C*n*, where each C*i* is a conjunction of literals. Examples include the ones shown to the right.

Conjunctive normal form (CNF) is a WFF of the form C1∧ or ∧C*n*, where each C*i* is a disjunction of literals. Examples include the ones shown on the bottom right.

Any WFF has:

* An equivalent WFF in disjunctive normal form
* An equivalent WFF in conjunctive normal form

How to turn any WFF into DNF (or CNF):

1. Get rid of the ⟶

* Use the equivalence: A ⟶ B ≡ (¬A) ∨ B (conv 1)

1. Use DeMorgan’s Laws to “push” ¬ into parentheses

* ¬(A ∧ B) ≡ ¬A ∨ ¬B (DeM 1)
* ¬(A ∨ B) ≡ ¬A ∧ ¬B (DeM 2)

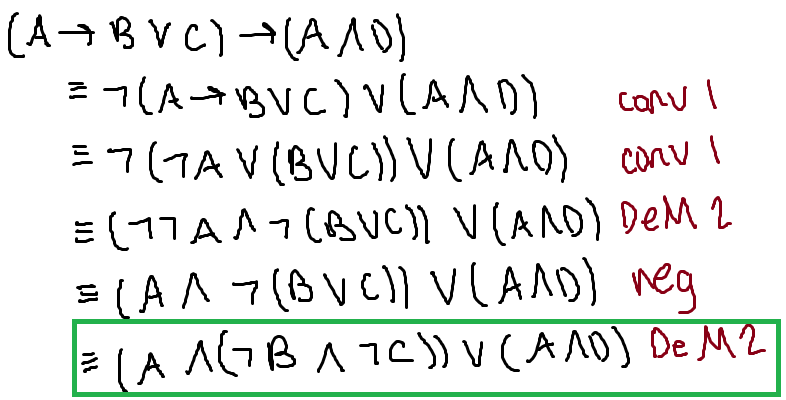
1. Get rid of double negation

* ¬ ¬A ≡ A (neg)

1. Use distributive and associative equivalences

* A ∧ (B ∨ C) ≡ (A ∧ B) ∨ (A ∧ C) (dist and)
* A ∨ (B ∧ C) ≡ (A ∨ B) ∧ (A ∨ C) (dist or)
* A ∨ (B ∨ C) ≡ (A ∨ B) ∨ C (assoc)
* A ∧ (B ∧ C) ≡ (A ∧ B) ∧ C (assoc)

**Converting to DNF: Example 1**: (A ⟶ B ∨ C) ⟶ (A ∧ D)



**Converting to DNF: Example 2:** (A ∨ B) ∧ (C ∨ D) (this is in CNF)

