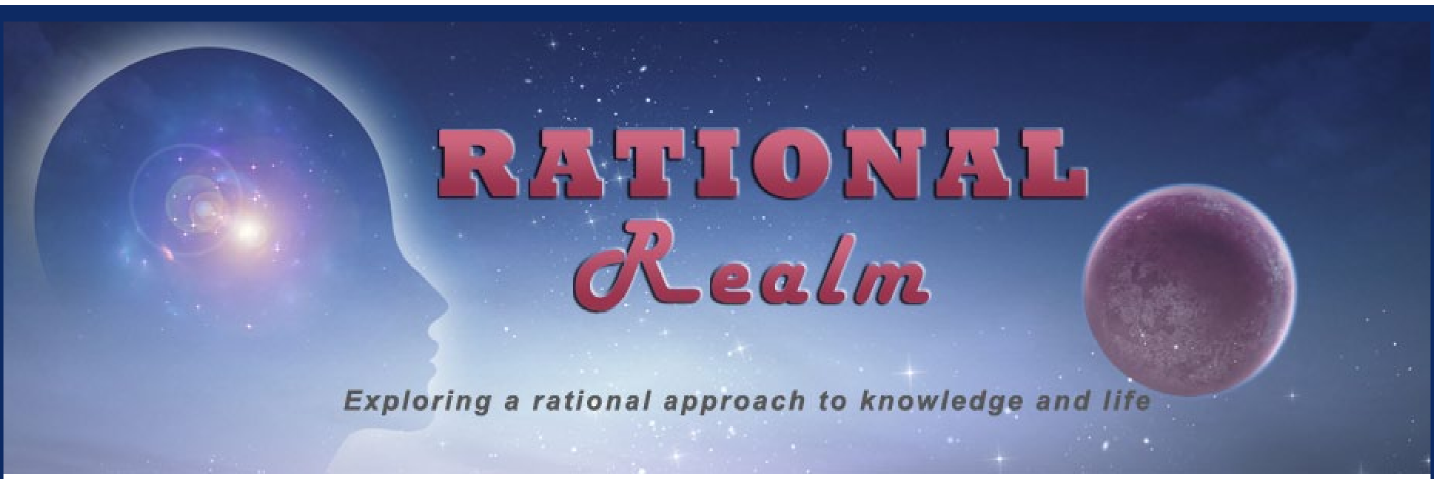


Propositional Logic – A Primer

Leslie Allan

Published online: 1 May 2023 (rev. 30 Jun 2023)

This tutorial is for beginners wanting to learn the basics of propositional logic; the simplest of the formal systems of logic. Leslie Allan introduces students to the nature of arguments, validity, formal proofs, logical operators and rules of inference. With many examples, Allan shows how these concepts are employed through the application of three different methods for proving the formal validity of arguments.

1. Introduction

Propositional logic is a type of formal logic that deals with the logical relationships between propositions. Propositions are statements that are either true or false. The purpose of propositional logic is to provide a formal system for representing and analyzing these logical relationships. Propositional logic uses logical operators, such as ‘and’, ‘or’ and ‘not’ to combine propositions and create more complex statements. Propositional logic is sometimes called ‘sentential logic’ or ‘propositional calculus’.

The main function of propositional logic is to provide a way to reason logically and systematically about the truth or falsity of propositions, and to create proofs or arguments that are logically valid. It is a powerful tool for identifying the logical structure of arguments and evaluating the validity of conclusions based on premises.

As such, propositional logic is an important tool in philosophy and mathematics for analyzing arguments and proofs. It has many other practical applications, such as in computer science for designing logical circuits and in natural language processing for analyzing the logical structure of sentences. Overall, propositional logic is a fundamental tool for reasoning and problem-solving in a wide range of fields.

What is an ‘argument’ in propositional logic? An argument consists of one or more propositions, each listed as a premise in the argument. It also consists of another proposition labeled the ‘conclusion’. An argument is considered valid if it is impossible for all of the premises to be true and the conclusion false. Here is a simple argument:

Premise 1: If it is raining, it is wet outside.

Premise 2: It is raining.

Conclusion: It is wet outside.

This argument is valid because it is impossible for its two premises to be true and the conclusion false. In propositional logic, an argument can be proved to be valid using one of three methods:

1. truth table method
2. natural deduction method
3. truth tree method (also called the ‘tableaux’ method)

In this primer, I will introduce you to all three methods.

First, what are the components of propositional logic? Propositional logic consists of a system of syntactical symbols and their arrangement, semantics that give meaning to the symbols and rules of inference that determine the validity of the conclusions of arguments.

The basic syntactical elements of propositional logic are:

Propositions: Propositions are statements represented by letters, such as ‘P’ and ‘Q’.

Logical Operators: Logical operators are used to connect propositions and create more complex statements. The most common operators in propositional logic are:

Negation: The negation operator (¬) is used to express the opposite of a proposition. For example, if ‘P’ is the proposition ‘It is raining’, then ‘¬P’ is the proposition ‘It is not raining’.

Conjunction: The conjunction operator (∧) is used to express that two propositions are true at the same time. For example, if ‘P’ is the proposition ‘It is raining’ and ‘Q’ is the proposition ‘It is cold’, then ‘P ∧ Q’ is the proposition ‘It is raining and cold’.

Disjunction: The disjunction operator (∨) is used to express that at least one of two propositions is true. For example, if ‘P’ is the proposition ‘It is raining’ and ‘Q’ is the proposition ‘It is sunny’, then ‘P ∨ Q’ is the proposition ‘It is raining or sunny’.

Implication: The implication operator (→) is used to express that if one proposition is true, then another proposition is also true. For example, if ‘P’ is the proposition ‘It is raining’ and ‘Q’ is the proposition ‘The streets are wet’, then ‘P → Q’ is the proposition ‘If it is raining, then the streets are wet’.

Biconditional: The biconditional operator (↔) is used to express that two propositions are true if and only if each other is true. For example, if ‘P’ is the proposition ‘You are a cat’ and ‘Q’ is the proposition ‘You have fur’, then ‘P ↔ Q’ is the proposition ‘You are a cat if and only if you have fur’.

The semantics of propositional logic is based on truth values of propositions. In this system, we stipulate that propositions can only be either true or false and we use logical operators to combine them to create more complex propositions.

Truth tables are often used to determine the truth value of a complex compound proposition. A truth table lists all possible combinations of truth values for the propositions that make up a compound proposition. The final column of the truth table displays the resulting truth value of the compound proposition as a function of the truth values of the component propositions.

Lastly, propositional logic includes a set of rules of inference that are used to determine whether the conclusion of an argument is valid based on the propositions listed in its premises. Some of the most common rules of inference in propositional logic include:

Modus Ponens:	If ‘If P then Q’ is true and ‘P’ is true, then ‘Q’ is true.
Modus Tollens:	If ‘If P then Q’ is true and ‘Q’ is false, then ‘P’ is false.
Disjunctive Syllogism:	If ‘P’ or ‘Q’ is true and the negation of ‘P’ is true, then ‘Q’ is true.
Conjunction:	If ‘P’ is true and ‘Q’ is true, then ‘P and Q’ is true.

Now that we’ve covered the basics, in the next sections, we’ll delve more into the inner workings of propositional logic.

2. Logical Operators and Truth Tables

In propositional logic, propositions are represented using a propositional variable, which is a symbol that stands for a proposition. Typically, logicians use uppercase letters such as ‘P’, ‘Q’, ‘R’, and so on, to represent propositional variables.

Propositional variables are combined using logical connectives to form more complex propositions, known as compound propositions. The main logical connectives in propositional logic are:

Negation (¬):	The negation of a proposition ‘P’, denoted by ‘¬P’, is the opposite of ‘P’. It is true if ‘P’ is false and false if ‘P’ is true.
Conjunction (∧):	The conjunction of two propositions ‘P’ and ‘Q’, denoted by ‘P ∧ Q’, is true if both ‘P’ and ‘Q’ are true and false otherwise.
Disjunction (∨):	The disjunction of two propositions ‘P’ and ‘Q’, denoted by ‘P ∨ Q’, is true if at least one of ‘P’ and ‘Q’ is true and false if both are false.
Implication (→):	The implication of two propositions ‘P’ and ‘Q’, denoted by ‘P → Q’, is true if ‘P’ implies ‘Q’. That is, if ‘P’ is true, then ‘Q’ must also be true. It is false if ‘P’ is true and ‘Q’ is false, and true otherwise.
Biconditional (↔):	The biconditional of two propositions ‘P’ and ‘Q’, denoted by ‘P ↔ Q’, is true if ‘P’ and ‘Q’ have the same truth value. That is, both are true or both are false. It is false if ‘P’ and ‘Q’ have opposite truth values.

These logical connectives combine in various ways to form more complex compound propositions. For example, we can form the proposition:

$$(P \land Q) \rightarrow (R \lor \neg S)$$

by combining conjunction, implication, disjunction, and negation.

Note that in other texts on logic, different symbols for the logical connectives may be given. Here are the main variations you may see:

Negation (¬):	alternatively as ~, -
Conjunction (∧):	alternatively as ·, &
Implication (→):	alternatively as ⇒, ⊃
Biconditional (↔):	alternatively as ⇔, ≡

Truth tables are used in propositional logic to show the truth values of compound propositions for all possible combinations of truth values of their component propositions. A truth table has one row for each possible combination of truth values of the component propositions and one column for each component proposition and for the resulting compound proposition. The truth values in the last column of the truth table indicate the truth value of the compound proposition for each combination of truth values of the component propositions.

Importantly, the truth value of a proposition is determined by its truth-functional semantics. That is, the truth value of a compound proposition is determined solely by the truth values of its component propositions and the truth-functional connectives that combine them.

How each of the logical connectives function to determine the truth of a compound proposition is specified in a truth table. For example, the truth table for the negation operator (¬) is:

Truth table – Negation

P	¬P
T	F
F	T

In this truth table, 'P' represents a proposition that can be either true (T) or false (F). The second column shows the truth value of the negation of 'P', which is true only when 'P' is false. Similarly, truth tables can be constructed for each of the other logical connectives, as shown below.

The truth table for the conjunction operator (∧) is:

Truth table – Conjunction

P	Q	P ∧ Q
T	T	T
T	F	F
F	T	F
F	F	F

The first two columns in the truth table represent the possible truth values for the propositions or expressions being evaluated, 'P' and 'Q'. The propositions being conjoined are known as ‘conjuncts’. The column labeled ‘P ∧ Q’ shows the truth value of the combined expression for each possible combination of truth values for its two conjuncts, ‘P’ and ‘Q’. The truth table shows that the conjunction of ‘P’ and ‘Q’ is true only if both ‘P’ and ‘Q’ are true, and is false otherwise.

The truth table for the disjunction operator (∨) is:

Truth table – Disjunction

P	Q	P ∨ Q
T	T	T
T	F	T
F	T	T
F	F	F

This truth table shows that the disjunction of ‘P’ and ‘Q’ is true only if either ‘P’ is true or ‘Q’ is true, or both are true, and is false if both ‘P’ and ‘Q’ are false. For disjunctions, the propositions or expressions being evaluated are known as ‘disjuncts’.

The truth table for the implication operator (→) is:

Truth table – Implication

P	Q	P → Q
T	T	T
T	F	F
F	T	T
F	F	T

This truth table shows that the implication from ‘P’ to ‘Q’ is true whenever ‘P’ is false or ‘Q’ is true, and false otherwise. For implications, the proposition preceding the operator is known as the ‘antecedent’, while the proposition following is known as the ‘consequent’.

The truth table for the biconditional operator (↔) is:

Truth table – Biconditional

P	Q	P ↔ Q
T	T	T
T	F	F
F	T	F
F	F	T

This truth table shows that the equivalence of ‘P’ and ‘Q’ is true only if ‘P’ and ‘Q’ are either both true or both false, and false otherwise.

Evaluating the formal validity of an argument requires first converting the argument given in natural language to symbolic form. Identifying the truth-functional components of a sentence and how they relate logically to the other components of the sentence requires practice with more complex sentences.

Here are some basic examples using simple sentences.

1. ‘If it rains today, I will stay at home.’

In this example, the proposition ‘it rains today’ can be represented by the variable ‘P’ and the proposition ‘I will stay at home’ can be represented by the variable ‘Q’. The statement can then be represented in symbolic form as:

$P \rightarrow Q$

2. ‘The cat is black and white.’

Here, ‘the cat is black’ can be represented by the variable ‘P’ and the proposition ‘the cat is white’ can be represented by the variable ‘Q’. The statement can then be represented in symbolic form as:

$P \land Q$

3. ‘Either it is sunny today or it is not raining today.’

In this example, the proposition ‘it is sunny today’ can be represented by the variable ‘P’ and the proposition ‘it is not raining today’ can be represented by the variable ‘Q’. The statement can then be represented in symbolic form as:

$P \lor \neg Q$

Once you are able to convert faithfully ordinary language arguments into symbolic form, you are then ready to move onto the next step: using truth tables to evaluate the logical validity of arguments and proofs by showing whether the conclusion logically follows from the premises.

3. Proofs Using Truth Tables

Of the three methods for determining the formal validity of an argument, the truth table method is the simplest. Consider this basic argument from the Introduction.

Premise 1:	If it is raining, it is wet outside.
Premise 2:	It is raining.
Conclusion:	It is wet outside.

Translating it into symbolic form using propositional variables and logical connectives, we get the following:

P1:	$P \rightarrow Q$
P2:	P
C:	Q

The argument is valid if the conclusion follows logically from the two premises. To use a truth table to determine the validity of the argument, we first construct a truth table that includes all of the premises and the conclusion. Testing for validity, we then verify that there is no case in which the premises are true and the conclusion false.

First, list all possible truth values for the component propositions in the left-most columns. Then, in the next column, work out the possible truth values for the compound proposition. For the final column, add the conclusion with its possible truth values. The truth table looks as follows:

Example 1 – Truth table proof

P	Q	P → Q	Q
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	F

To determine validity, look at every row in which both premises (‘P → Q’; ‘P’) are true. For those rows (the first row only, in this case), is there a case in which the conclusion (‘Q’) is not true? There is not. Therefore, the argument is valid.

Consider a more complicated argument with three premises.

Premise 1:	Either Tom is a man or Tom is a teacher.
Premise 2:	If Tom is a man, then Tom is a bachelor.
Premise 3:	It is not the case that Tom is a teacher.
Conclusion:	Tom is a bachelor.

Translating to symbolic form yields:

P1:	P ∨ Q
P2:	P → R
P3:	¬Q
C:	R

We construct the truth table as follows.

Example 2 – Truth table proof

P	Q	R	P ∨ Q	P → R	¬Q	R
T	T	T	T	T	F	T
T	T	F	T	F	F	F
T	F	T	T	T	T	T
T	F	F	T	F	T	F
F	T	T	T	T	F	T
F	T	F	T	T	F	F
F	F	T	F	T	T	T
F	F	F	F	T	T	F

The first three columns list all of the possible truth values for the atomic propositions. The next three columns list the possible truth values of the compound propositions making up the three premises. The final column lists the possible truth values for the conclusion ('R'). This final column is a repeat of the third column in order to make a comparison of the truth values of the conclusion with those of the three premises. On inspection, we see that there is no line of the truth table in which all three premises are true (the third row only, in this case) and the conclusion is false. Therefore, the argument is valid.

In propositional logic, a valid argument is an argument in which the conclusion necessarily follows from the premises. In other words, if the premises of a valid argument are true, then the conclusion must also be true. Note that the conclusion of a valid argument is not necessarily true. The truth of the conclusion is only guaranteed by the truth of the premises and the logical structure of the argument. If one or more of the premises are false, the conclusion may be false. If both conditions are met (that is, the premises are true and the argument is valid), logicians call that kind of argument a ‘sound’ argument.

Now, some compound propositions are necessarily true independently of the truth of other propositions. Logicians call this kind of proposition a ‘tautology’. For a tautology, it is impossible for it to be false under any circumstances. They are always true, regardless of the truth values of its component propositions. ‘It is raining or it is not the case that it is raining’ (expressed as ‘P ∨ ¬P’) is an example of a tautology. It can’t be otherwise than true.

We can use truth tables to prove a compound proposition is a tautology. By constructing a truth table and inspecting it, we can verify that a proposition is true for all possible truth values of its component propositions.

As a simple example, we can prove that ‘P ∨ ¬P’ is a tautology. Lay out all of the possible truth values of the component propositions in a truth table along with the resulting possible truth values of the tautology.

Example 3 – Truth table proof

P	¬P	P ∨ ¬P
T	F	T
F	T	T

The truth table shows that ‘P ∨ ¬P’ is true for all possible truth values of ‘P’ and ‘¬P’. Therefore, ‘P ∨ ¬P’ is a tautology.

Just as we can use truth tables to prove a compound proposition to be necessarily true, we can also use them to prove a proposition to be necessarily false—a logical contradiction. A logical contradiction is a compound proposition that is always false, regardless of the truth values of its component propositions. For example, we can prove that ‘P ∧ ¬P’ is a logical contradiction using the following truth table:

Example 4 – Truth table proof

P	¬P	P ∧ ¬P
T	F	F
F	T	F

The truth table shows that ‘P ∧ ¬P’ is false for all possible truth values of ‘P’ and ‘¬P’. Therefore ‘P ∧ ¬P’ is a contradiction.

Let’s now prove a more complex tautology, such as this equivalence to the negation of a conjunction:

¬(P ∧ Q) ↔ ¬P ∨ ¬Q

The equivalence is necessarily true if both sides of the equivalence operator have the same truth value for all possible truth values of ‘P’ and ‘Q’.

Example 5 – Truth table proof

P	Q	P ∧ Q	¬(P ∧ Q)	¬P	¬Q	¬P ∨ ¬Q
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

Look down each row of the truth table to see whether for one or more truth assignments of ‘P’ and ‘Q’ the truth value of ‘¬(P ∧ Q)’ is not identical to the truth value of ‘¬P ∨ ¬Q’. As there are no such instances, ‘¬(P ∧ Q) ↔ ¬P ∨ ¬Q’ is a tautology.

We can do the same now with the negation of a disjunction:

¬(P ∨ Q) ↔ ¬P ∧ ¬Q

Is this also a tautology? As before, construct the truth table and inspect the result.

Example 6 – Truth table proof

P	Q	P ∨ Q	¬(P ∨ Q)	¬P	¬Q	¬P ∧ ¬Q
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

The truth table shows that '¬(P ∨ Q)' and '¬P ∧ ¬Q' have the same truth values for all possible truth values of 'P' and 'Q'. Therefore, '¬(P ∨ Q) ↔ ¬P ∧ ¬Q' is a tautology.

These latter two tautologies are known as De Morgan's Laws. They state that the negation of a conjunction is the disjunction of the negations, and the negation of a disjunction is the conjunction of the negations.

It is important to note that while truth tables are useful for proving validity and tautologies in propositional logic, they can become very large and difficult to read for compound propositions with many component propositions. In such cases, it is often more practical to use other methods, such as the natural deduction method and truth trees.

4. Rules of Inference

In propositional logic, arguments are made up of a set of premises and a conclusion. Arguments are proved valid by using logical rules of inference. These are formally defined rules that allow one to infer a conclusion from a set of premises. A valid argument is one in which the conclusion necessarily follows from the premises according to these rules.

There are several logical rules of inference that can be used to prove the validity of an argument. The most common rules you will come across are:

Modus Ponens (MP):	If ‘If P then Q’ is true and ‘P’ is true, then ‘Q’ is true. If ‘$P \rightarrow Q$’ and ‘P’ are true, then ‘Q’ is true. Example: If it is raining, the ground is wet. It is raining. Therefore, the ground is wet.
Modus Tollens (MT):	If ‘If P then Q’ is true and ‘Q’ is false, then ‘P’ is false. If ‘$P \rightarrow Q$’ and ‘$\neg Q$’ are true, then ‘$\neg P$’ is true. Example: If it is raining, the ground is wet. The ground is not wet. Therefore, it is not raining.
Hypothetical Syllogism (HS):	If ‘If P then Q’ is true and ‘If Q then R’ is true, then ‘If P then R’ is true. If ‘$P \rightarrow Q$’ and ‘$Q \rightarrow R$’ are true, then ‘$P \rightarrow R$’ is true. Example: If it is raining, the ground is wet. If the ground is wet, the grass is slippery. Therefore, if it is raining, the grass is slippery.
Conjunction (CONJ):	If ‘P’ is true and ‘Q’ is true, then ‘P and Q’ is true. If ‘P’ and ‘Q’ are true, then ‘$P \land Q$’ is true. Example: It is raining. The grass is slippery. Therefore, it is raining and the grass is slippery.

Here are some other rules of inference that are used in proofs of validity:

Disjunctive Syllogism (DS):	If ‘P or Q’ is true and the negation of ‘P’ is true, then ‘Q’ is true. If ‘$P \lor Q$’ and ‘$\neg P$’ are true, then ‘Q’ is true. Example: The sky is blue or grey. The sky is not blue. Therefore, the sky is grey.

Addition (ADD):

If ‘P’ is true, then ‘P or Q’ is also true.

If ‘P’ is true, then ‘P ∨ Q’ is true.

Example: The sky is blue. Therefore, the sky is blue or the grass is green.

Simplification (SIMP):

If ‘P and Q’ is true, then ‘P’ and ‘Q’ are both true.

If ‘P ∧ Q’ is true, then ‘P’ and ‘Q’ are true.

Example: The sun is shining and the sky is blue. Therefore, the sun is shining. The sky is blue.

Constructive Dilemma (CD):

If ‘If P then Q’ is true and ‘If R then S’ is true and ‘P’ or ‘R’ is true, then ‘Q’ or ‘S’ is true.

If ‘P → Q’ and ‘R → S’ are true and ‘P’ or ‘R’ is true, then ‘Q ∨ S’ is true.

Example: If it is raining, the ground is wet. If the sun is shining, the sky is blue. It is raining or the sun is shining. Therefore, the ground is wet or the sky is blue.

Double Negation (DN):

If ‘P’ is true, then the negation of the negation of ‘P’ is true.

If ‘P’, then ‘¬¬P’ is true.

Example: The sun is shining. Therefore, it is not the case that the sun is not shining.

Exportation (EXP):

If ‘If “P and Q” then R’ is true, then ‘If P then “If Q then R”’ is true.

If ‘“P ∧ Q” → R’ is true, then ‘P → “Q → R”’ is true.

Example: If it is raining and the tent is uncovered, the tent is wet, then if it is raining, then if the tent is uncovered, the tent is wet.

Material Implication (MI):

If ‘If P then Q’ is true, then ‘P’ is false or ‘Q’ is true.

If ‘P → Q’, then ‘¬P ∨ Q’ is true.

Example: If it is raining, the ground is wet. Therefore, it is not raining or the ground is wet.

De Morgan’s Law (DM):

If ‘P and Q’ is false, then ‘P’ is false or ‘Q’ is false.

If ‘P ∧ Q’ is false, then ‘¬P ∨ ¬Q’ is true.

Example: The sun is not shining and it is not raining. Therefore, the sun is not shining or it is not raining.

To sum up, rules of inference are used in propositional logic to derive new propositions from existing ones. A rule of inference is a logical principle that allows us to infer the truth of one proposition from the truth of other propositions. In the next session, we will explore how these rules of inference are used to validate arguments.

5. Proofs Using Natural Deduction

To prove that an argument is valid, one must show that the conclusion necessarily follows from the premises using the logical rules of inference given in the previous section or other such additional rules of inference. If an argument is valid, then it is impossible for the premises to be true and the conclusion to be false at the same time. For a valid argument, to assert the premises while denying the conclusion is to assert a contradiction. We say that the premises logically entail the conclusion.

As a simple example, suppose we have the following premises:

Premise 1:	If it is raining, the streets are wet.
Premise 2:	It is raining.

Using the rule of inference called Modus Ponens (MP), we can derive the conclusion:

Conclusion: The streets are wet.

The proof can be written as follows:

Example 1 – Natural deduction proof

P1:	If it is raining, the streets are wet.	[P]
P2:	It is raining.	[P]
C:	The streets are wet.	[1,2 MP]

This proof demonstrates that if the conditional, if it is raining, then the streets are wet, is true and it is also true that it is raining, we can validly conclude that the streets are wet.

The letters and numbers that appear to the right of each proposition are the abbreviation of the inference rules applied and the premise numbers to which the rule applies.

In this example, the [P] rule is the Premise Introduction rule. This rule states that a given premise may be introduced at any stage in the proof. The [MP] rule is Modus Ponens and here it is applied to Premises 1 and 2.

For a more complex example, suppose we want to prove the three premises ‘P → ¬Q’ and ‘R → Q’ and ‘R’ entail ‘¬P’.

The proof proceeds as follows.

Example 2 – Natural deduction proof

(1)	$R \rightarrow Q$	[P]
(2)	R	[P]
(3)	Q	[1,2 MP]
(4)	$P \rightarrow \neg Q$	[P]
(5)	$\neg P$	[3,4 MT]

Line (1) is the second premise in our original list of premises. Line (2) is the third premise in our original list of premises. Line (3) is derived from lines (1) and (2) using the rule Modus Ponens. Line (4) is the first premise in our original list of premises. Line (5) is the proved conclusion, reached by deriving it from lines (3) and (4) using the rule Modus Tollens.

Note how the validity of an argument is a function of the syntactical structure of the argument and the truth of each of the premises. We can replace the letters representing each proposition with any interpretation (such as replacing ‘P’ with ‘It is raining’) and our argument will remain valid. The only proviso is that we remain consistent with our interpretation of each letter. For example, if we make ‘P’ stand for ‘It is raining’, ‘It is raining’ must be the interpretation of ‘P’ in all places where ‘P’ occurs in the argument.

Here is a third example using the natural deduction method of proof. The task is to prove ‘$\neg P \rightarrow (R \land S)$’ and ‘$P \rightarrow Q$’ and ‘$\neg Q$’ entail ‘R’. The proof is as follows.

Example 3 – Natural deduction proof

(1)	$P \rightarrow Q$	[P]
(2)	$\neg Q$	[P]
(3)	$\neg P$	[1,2 MT]
(4)	$\neg P \rightarrow (R \land S)$	[P]
(5)	$R \land S$	[3,4 MP]
(6)	R	[5 SIMP]

An important learning to note here is that the conditional proposition constructed from the conjunction of the initial premises forming the antecedent of the conditional and the proved proposition constituting the consequent form a tautology. This is true for all valid arguments. For the preceding proof, the tautology constructed is:

$$((\neg P \rightarrow (R \land S)) \land (P \rightarrow Q) \land \neg Q) \rightarrow R$$

This proposition is true for all interpretations of the component propositions. It must be true for all possible truth values we assign to 'P', 'Q', 'R' and 'S'. To indicate this logical entailment from the conjunction of the premises to the conclusion, we use a special symbol: ⊨

We can then express the entailment thus:

((¬P → (R ∧ S)) ∧ (P → Q) ∧ ¬Q) ⊨ R

Using the entailment symbol, we indicate that the conditional is not only true for some interpretations of the component propositions, but for all interpretations. Some beginners in logic get confused between implication '→' (sometimes called 'material implication') and logical entailment '⊨'. Remember, if 'P' implies 'Q' is true, all this means is that 'P' is false or 'Q' is true. If 'P' entails 'Q', on the other hand, then there is no interpretation of 'P' and 'Q' in which 'P' is true and 'Q' is false. If 'P' is true, then there is no possibility that 'Q' is false in virtue of the components of 'P'.

Let us look at one more proof using the natural deduction method. Let's prove 'P ∧ Q' and 'P → ¬(Q ∧ R)' and 'S → R' entail '¬S'. The proof is as follows.

Example 4 – Natural deduction proof

(1)	P ∧ Q	[P]
(2)	P	[1 SIMP]
(3)	Q	[1 SIMP]
(4)	P → ¬(Q ∧ R)	[P]
(5)	¬(Q ∧ R)	[2,4 MP]
(6)	¬Q ∨ ¬R	[5 DM]
(7)	¬R	[3,6 DS]
(8)	S → R	[P]
(9)	¬S	[7,8 MT]

A proof is a sequence of propositions that follow from a given set of premises using rules of inference. A valid argument is one in which the conclusion necessarily follows from the premises. There are several rules of inference that can be used to determine the validity of an argument, such as Modus Ponens, Modus Tollens, Simplification and Disjunctive Syllogism. In this section, we have seen how the natural deduction method is used to prove the validity of arguments. In the next section, we will examine a more graphical method for conducting formal proofs in propositional logic.

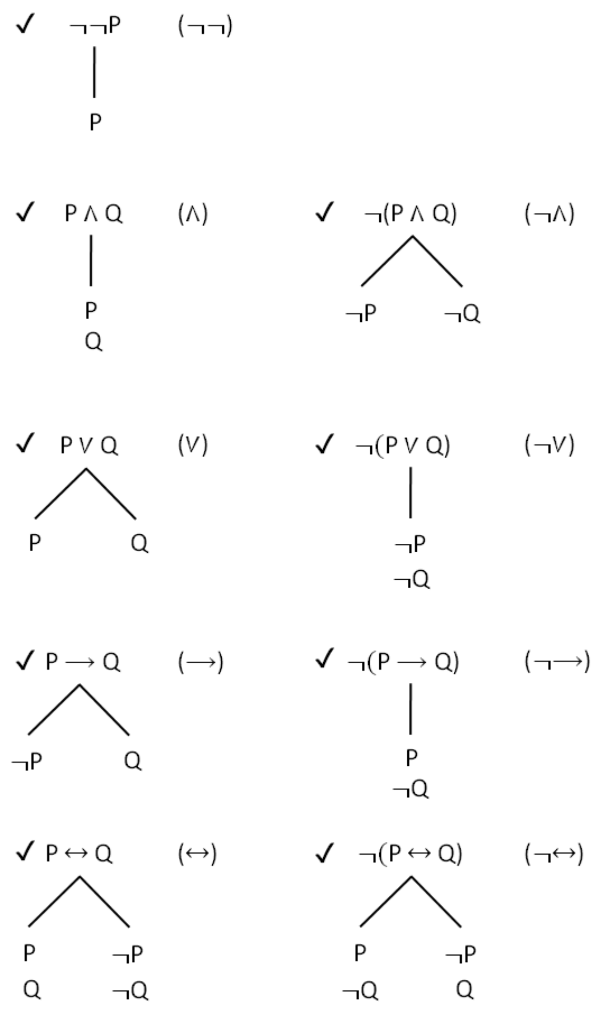
6. Proofs Using Truth Trees

For proving the validity of arguments and demonstrating tautologies, the truth table method is effective up to a point. Beyond around three letters representing atomic propositions, the truth table becomes unwieldy with so many possible truth values to map. The natural deduction method is a powerful alternative, but for the more complex proofs it requires a lot of practice and ingenuity. The truth tree method has the advantage of handling the more complex proofs while providing a mechanical method for proving validity and tautologousness. For these two reasons, the truth tree method is my preferred method.

This method tests for whether it is possible for all of the premises to be true while the conclusion be false. We do that by assuming the premises true, negating the conclusion and testing for consistency for the set of propositions. If it is possible for the premises to be true and the conclusion false, then we can conclude that the argument is not valid according to the inference rules of propositional logic. We do this graphically, drawing a truth tree to map the components of composite propositions, and by applying a small number of fixed rules using a set procedure.

With this method, there are nine rules for decomposing propositions. These are as shown below, with the name for each rule shown in parentheses to the right of the rule. The letters 'P' and 'Q' stand for any proposition.

Truth tree decomposition rules



Note how each of these decomposition rules mirror a truth table and a rule of inference. Take the Double Negation (DN) rule of inference: If ‘P’ is true, then the negation of the negation of ‘P’ is true. The first decomposition rule (¬¬) represents that equivalence graphically with a line from ‘¬¬P’ to ‘P’.

Similarly, the second decomposition rule (∧) represents the inference rule Simplification (SIMP) and the truth table for (∧). The single line in the decomposition rule indicates that ‘P’ and ‘Q’ are true together.

The third decomposition rule (¬∧) is an example of branching. This rule represents the inference rule De Morgan’s Law (DM): If ‘P and Q’ is false, then ‘P’ is false or ‘Q’ is false. The branching indicates that at least one of the branches is true; in this case, ‘¬P’ or ‘¬Q’. The same is true for the remaining decomposition rules.

The procedure for applying the above decomposition rules in building the tree and testing for validity is as follows.

STEP 1: List each of the initial premises of the argument on a separate line.

STEP 2: Write under the list of initial premises the negation of the conclusion.

STEP 3: Select a proposition from the list and apply the correct decomposition rule, adding the decomposition to every open path that can be reached from the decomposed proposition.

Add a check mark (✓) to the left of the decomposed proposition to indicate that it has been decomposed.

STEP 4: Close any path that has a component proposition and its negation appearing in it.

Mark the path as closed by drawing a cross ‘X’ under it.

STEP 5: Inspect to see if all paths are closed. If YES, stop. The argument is valid.

If NOT, have all propositions been decomposed?

If NOT, return to STEP 3. If YES, the argument is invalid.

If there are any paths left open after you have decomposed all of the propositions, it means there is a way for the premises to be true and the conclusion false. Each open path shows which of the component propositions are true in those scenarios in which the premises are true and the conclusion false. The open paths show you how the argument is invalid.

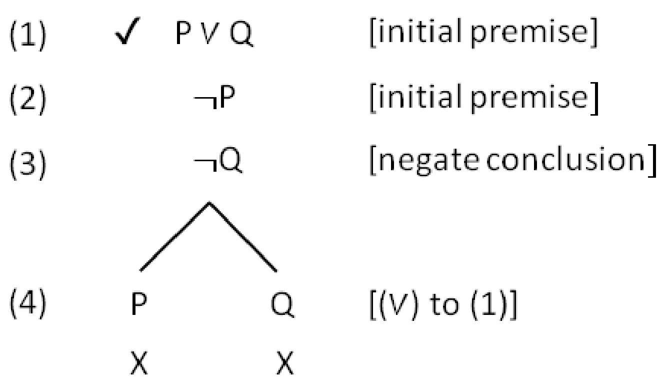
If all the paths end up closed, that means there is no possible way in which the premises are true and the negation of the conclusion is also true. In other words, there is no way for the premises to be true and the conclusion false. Hence, the argument is demonstrated to be valid.

Here are a couple of tips for keeping the branching simple:

1. When selecting a complex proposition for decomposing, prefer one that gives a trunk and not branches.
2. If branching is unavoidable, choose a complex proposition that you know will result in a closed branch.

Let’s start with a simple example of a proof using the truth tree method. The task is to test whether ‘P ∨ Q’ and ‘¬P’ entail ‘Q’. Our test looks as follows.

Example 1 – Truth tree proof



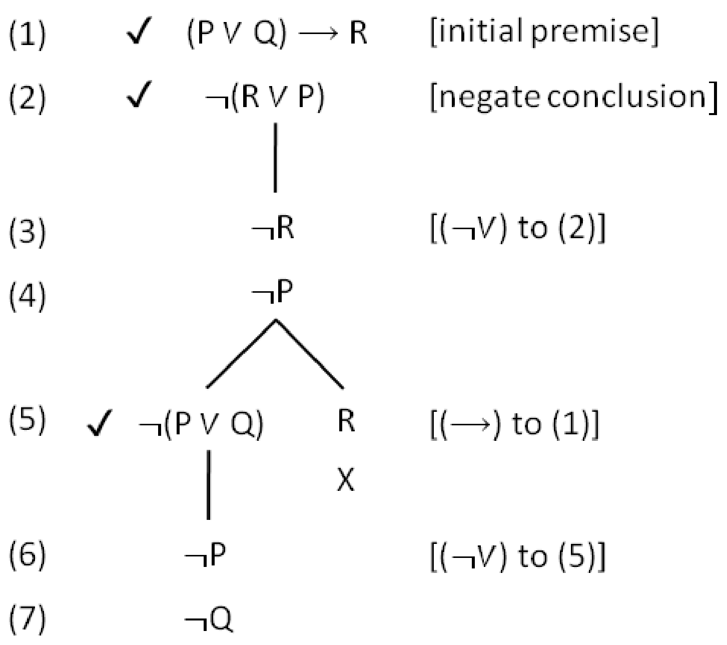
The numbers to the left of the tree are the line numbers of the proof. The bracketed comments to the right of the tree indicate the source of the line and the decomposition rule applied. Lines (1) and (2) are the application of STEP 1. Line (3) is the application of STEP 2. We now apply the decomposition rule (∨) to line (1) (STEP 3) to create the branch resulting in line (4). We also add a check mark to line (1) to show that we have dealt with it.

Inspecting each path for a contradiction (STEP 4), we can see that every path contains a proposition and its exact negation. Hence, we mark both paths as closed with an ‘X’. As all paths are closed (STEP 5), we have proved that the argument is valid. It’s impossible for the premises to be true and the conclusion false.

Note that if all paths had not been closed, the propositions in lines (2) and (3) do not require further decomposition as the negation of an atomic proposition cannot be broken down into simpler components.

Let us now try a second example. In this example, we want to find out whether the single premise ‘(P ∨ Q) → R’ entails ‘R ∨ P’. The final truth tree appears as follows.

Example 2 – Truth tree proof


(1) ✓ (P ∨ Q) ⟶ R [initial premise]
(2) ✓ ¬(R ∨ P) [negate conclusion]
(3) ¬R [(¬∨) to (2)]
(4) ¬P
(5) ✓ ¬(P ∨ Q) R [(⟶) to (1)]
X
(6) ¬P [(¬∨) to (5)]
(7) ¬Q

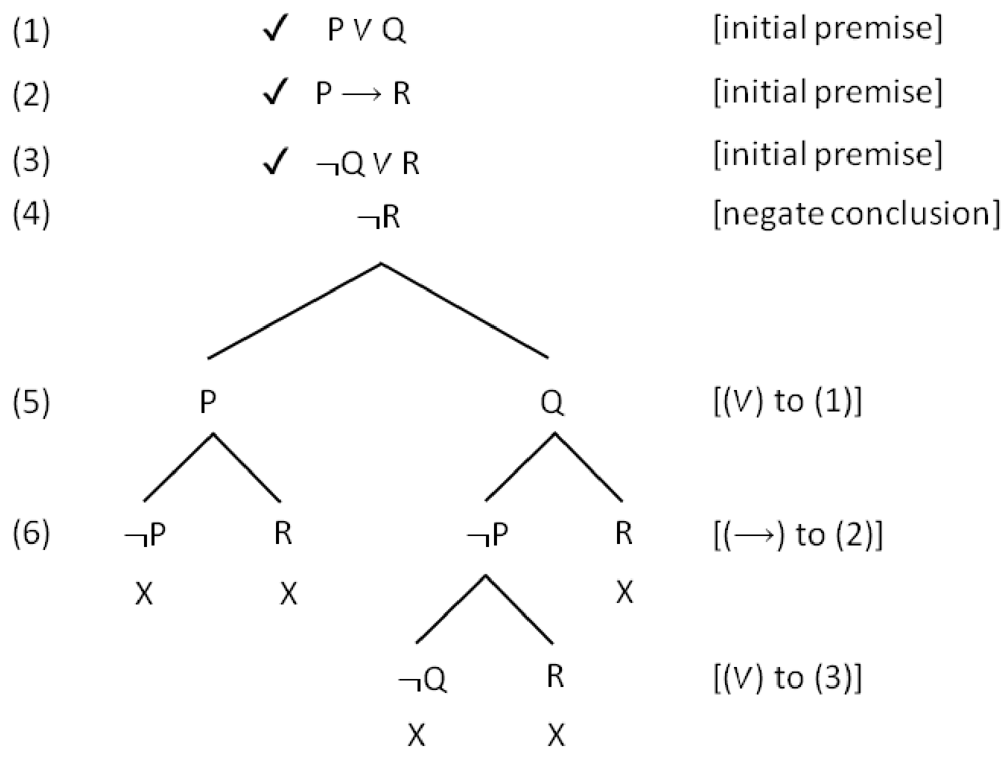
Line (1) is the initial premise, with line (2) listing the negation of the conclusion (STEPS 1 and 2). Applying the decomposition rule (¬∨) to line (2), we write in lines (3) and (4) (STEP 3). We also check off line (2) to show that we have decomposed it. So far, no proposition and its negation appear in any path in the tree. So, we continue from STEP 5 to decompose the next proposition. Applying the decomposition rule (→) to the proposition in line (1) gives us line (5). We check off line (1). Inspecting for contradictions in each path (STEP 4), we close of the right path as 'R' and its negation appear in the path.

The left path remains open, so we must decompose a proposition (STEP 5). There is only one proposition left to decompose; that appearing in line (5) as '¬(P ∨ Q)'. We apply the decomposition rule (¬∨) to it to write in lines (6) and (7). Inspecting that remaining path, we see that it remains open with no contradictions showing in the path. We conclude that the argument is not valid according to the rules of propositional logic.

From the open path, we can see the truth values of the atomic sentences in the case where the premises are true and the conclusion is false. We see that occurs when 'P', 'Q' and 'R' are all false. Importantly, note that because we have proved that the argument is invalid according to the rules of propositional logic, we cannot conclude that the original argument prior to translation to symbolic logic is necessarily invalid. It may be that a further decomposition of the internal structure of the atomic sentences using a more advanced logic shows the argument to be valid after all.

For our final example, let's prove whether or not the three premises 'P ∨ Q', 'P → R' and '¬Q ∨ R' entail 'R'.

Example 3 – Truth tree proof


(1) ✓ P ∨ Q [initial premise]
(2) ✓ P ⟶ R [initial premise]
(3) ✓ ¬Q ∨ R [initial premise]
(4) ¬R [negate conclusion]
(5) P Q [(∨) to (1)]
(6) ¬P R ¬P R [(⟶) to (2)]
X X X
¬Q R [(∨) to (3)]
X X

Lines (1), (2) and (3) list the premises while line (4) is the negation of the conclusion. Decomposition rule (∨) is first applied to line (1) to give the first branch and then checked off. With no paths to close, decomposition rule (→) is applied to line (2) and checked off. Note how the application of the (→) rule appears in both open branches in line (5). This is because the proposition in line (2) appears at the head of each path. Following decomposition in line (6), we can now close three of the four paths. One open path remains, so we must decompose the proposition in line (3) that has so far not been decomposed. Applying the (∨) rule gives us the final branch in the tree. On inspection, we close each of the final two paths. With all paths closed, we determine the argument is valid.

The truth tree method can also be used to prove tautologies. In this case, we negate the tautology and then proceed to decompose the negated proposition using the standard decomposition rules. Once the single premise is fully decomposed, if all paths are closed, we have proved that the proposition is a tautology. We have proved that there is no possible interpretation of the atomic propositions that makes the composite proposition false.

The truth tree method is a simple but effective way of proving the validity of arguments and tautologies. By applying a small set of decomposition rules and following some procedural steps, we have available a mechanical method for demonstrating the logical entailment of a conclusion from a set of premises.

7. Logical Fallacies

With the tools and methods illustrated in the previous sections, you can formally prove the validity and invalidity of some basic arguments expressed in propositional logic. In addition, you are also now able to identify reasonably quickly some formal logical fallacies. These are errors in reasoning that arise from violations of the inference rules of propositional logic covered previously.

Keep in mind that there are many informal errors of reasoning that are sometimes labeled incorrectly as 'logical fallacies'. These are more typically cognitive biases and include such errors as loss aversion, anchoring bias and confirmation bias.

There are four basic logical fallacies identified using propositional logic. Each of these formal fallacies is included under the general umbrella term: 'non-sequiter'(from Latin 'it does not follow'). In different ways, the invalid conclusion does not follow logically from the stated premises. These four basic logical errors are as follows.

Affirming the Consequent

Form: If P then Q

Q

Conclusion: P

Example: If it is raining, then the ground is wet. The ground is wet. Therefore, it is raining. In fact, a pipe burst.

Explanation: This fallacy mistakenly assumes that if the consequent of a conditional statement is true, the antecedent must also be true. However, there may be other reasons for the consequent to be true.

Denying the Antecedent

Form: If P then Q

Not P

Conclusion: Not Q

Example: If it is raining, then the ground is wet. It is not raining. Therefore, the ground is not wet. In fact, a pipe burst.

Explanation: This fallacy incorrectly concludes that if the antecedent of a conditional statement is false, the consequent must also be false. However, there may be other reasons for the consequent to be true.

Affirming a Disjunct

Form: P or Q

P

Conclusion: Not Q

Example: Sue is working or Sue is at home. Sue is working. Therefore, Sue is not at home. In fact, Sue is working at home.

Explanation: This fallacy incorrectly concludes that if one of the disjuncts of a disjunction is true, the other disjunct must be false. However, there may be instances where both disjuncts are true together.

Denying a Conjunct

Form: Not P and Q

Not P

Conclusion: Q

Example: It is not the case that both Sue is at work and Sue is at home. Sue is not at work. Therefore, Sue is at home. In fact, Sue is at a park.

Explanation: This fallacy incorrectly concludes that if one of the conjuncts of a conjunction is false, the other conjunct must be true. However, there may be instances where both conjuncts are false together.

You are now in a position to identify some basic logical errors using your knowledge of propositional logic. See how many of these errors you can spot in your readings and in conversations with others.

8. Applications and Limitations

Propositional logic is a powerful and versatile system for reasoning about propositions. Its simplicity and clarity make it a valuable tool in many different fields. By understanding the basics of propositional logic, including truth tables, rules of inference and proofs, we can develop a deeper understanding of logical reasoning and critical thinking.

Propositional logic has many practical applications in fields such as computer science, mathematics and philosophy. Some examples of its applications are:

1. Digital Circuits: Propositional logic is used to design and analyze digital circuits, which are the building blocks of modern computers and electronic devices. In digital circuits, propositional logic is used to represent and manipulate Boolean functions, such as AND, OR and NOT. Propositional logic describes the behaviour of these circuits in terms of inputs, functions and outputs.

2. Programming: Propositional logic is used in programming languages to represent and manipulate logical expressions, which are used to control the flow of a program. For example, the conditional statements in programming languages, such as 'if-then' and 'switch-case', are based on propositional logic. At the end of the process, propositional logic is used to verify the correctness of computer programs and software systems.

3. Philosophy: Propositional logic is used in philosophy to analyze arguments and to evaluate their logical validity. It is also used to develop theories of knowledge through reasoning about and clarifying concepts such as 'truth', 'belief' and 'knowledge'. The principles of propositional logic are used to construct more complex logical systems, such as predicate logic, modal logic and epistemic logic. These are used to model different types of knowledge and belief.

4. Mathematics: Propositional logic is used in mathematics to prove theorems, establish logical relationships between mathematical concepts and to study the foundations of mathematics. For example, propositional logic can be used to prove that a certain mathematical statement is always true, or to prove that two mathematical statements are equivalent.

5. Linguistics: Propositional logic is used in linguistics to study the structure and meaning of natural language sentences. Logical operators, such as conjunction, disjunction and negation, are used to analyze the structure of sentences and the principles of propositional logic are used to develop logical models of language understanding and interpretation.

6. Artificial Intelligence: Propositional logic is used in artificial intelligence to represent and manipulate knowledge in the form of logical statements. For example, propositional logic can be used to represent the knowledge of a robot about its environment and to reason about its actions.

Propositional logic provides a way to reason logically and systematically about the truth or falsity of propositions and to create proofs or arguments that are logically valid. In spite of its strengths, it is important to note that propositional logic has some limitations that make it unsuitable for certain types of reasoning and analysis.

One limitation is that it cannot handle quantifiers, such as ‘all’, ‘some’ and ‘none’, which are used to express the scope of a proposition. For example, the proposition, ‘All men are mortal’, cannot be expressed in propositional logic.

Another limitation is that propositional logic does not provide a way to represent the meaning of words or concepts. For example, the proposition, ‘Paris is the capital of France’, can be expressed in propositional logic, but the meaning of the terms ‘Paris’ and ‘France’ are not represented.

Finally, propositional logic assumes that propositions are either true or false, without any stipulation of the degree of uncertainty or probability. This can limit its usefulness in analyzing complex systems or situations where there is a degree of doubt or ambiguity.

To overcome these limitations, several extensions of propositional logic have been developed, including predicate logic, modal logic and fuzzy logic. These extensions allow for a more expressive and flexible representation of logical relationships. They have allowed us to gain a deeper understanding in fields such as artificial intelligence, robotics, and natural language processing.

Despite its limitations, propositional logic remains a powerful tool for analyzing and evaluating arguments, and for developing logical systems in a wide range of fields. Its simplicity and ease of use make it an ideal starting point for learning about logic and logical reasoning. By understanding and using propositional logic, we can improve our ability to reason about complex systems and concepts.

Further Reading

Baron, Richard 2014. Introduction to Logic, URL = <https://www.rbphilo.com/intrologic.pdf>.

Hodges, Wilfrid 1977. *Logic*, Harmondsworth: Penguin.

Klement, Kevin C. 2023. Propositional Logic, *The Internet Encyclopedia of Philosophy*, ISSN 2161-0002, URL = <https://iep.utm.edu/propositional-logic-sentential-logic/>.

de Swart, Harrie 2018. *Philosophical and Mathematical Logic*, Cham: Springer.

First published	May 1, 2023
Substantive revision	Jun 30, 2023 (§7: added section on logical fallacies, expanded reading list)