A proposition is a declarative sentence that is either true or false. If it is always true, then it is denoted by “T” and, if it is always false, then it is denoted by “F”. They can't be commands, questions, or have unknown variables.

Compound propositions are constructed from logical connectives and other propositions

The negation of a proposition p is denoted by ¬p. It is the opposite of the proposition. If p is T, then ¬p is F. If p is F, then ¬p is T.

For propositions p and q, it is denoted by p ^ q. It’s a compound statement formed by joining two statements with the connector AND. It is true if and only if both statements are true; otherwise, it is false.

For propositions p and q, it is denoted by p v q. It’s a compound statement formed by joining two statements with the connector OR. A disjunction is false if and only if both statements or false; otherwise it is true.

OR connective, a disjunction that remains true if either or both of its arguments are true.

XOR connective, a disjunction that is true if only one, but not both of its arguments are true, and is false if neither or both are true, denoted by ⊕.

Conditional statement, if p and q are propositions, then p → q is the implication which reads “if p, then q”. In this case, p is the hypothesis (aka antecedent or premise) and q is the conclusion. [ If p = T and q = T, then p → q = T [If p = T and q = F, then p → q = F [ If p = F and q = T, then p → q = T [ If p = F and q = F, then p → q = T

Flip the hypothesis and conclusion of p → q [ The converse of p → q is q → p

Flip and negate p → q [ The contrapositive of p → q is ¬q → ¬p

Negate p → q [ The inverse of p → q is ¬p → ¬q

True whenever both parts have the same truth value, if p and q are propositions, then p ↔ q is the biconditional statement [ If p and q both have the same true or false value, then p ↔ q = T. Else, p ↔ q = F.

Shows true or false for compound propositions.

Two propositions are equivalent if they always have the same truth value.

As follows ¬ (negation) > ^ (conjunction) > v (disjunction) > → (implication) > ↔ (biconditional)

Puzzles that can be solved using logical reasoning. These are excellent for practicing working with the rules of logic.

A proposition which is always true [ Ex: p v ¬p

A proposition which is always false [ Ex: p ^ ¬p

A proposition which is neither a tautology nor a contradiction [ Ex: p

A two compound propositions p and q are logically equivalent if p ↔ q is a tautology. This is written as p ⇔ q or as p ≡ q where p and q are compound propositions.

As follows [ ¬(p ^ q) ≡ ¬p v ¬q [ ¬(p v q) ≡ ¬p ^ ¬q

As follows [ p ^ T ≡ p [ p v F ≡ p

As follows [ p v T ≡ T [ p ^ F ≡ F

As follows [ p v p ≡ p [ p ^ p ≡ p

As follows ¬(¬p) ≡ p

As follows [ p v ¬p ≡ T [ p ^ ¬p ≡ F

As follows [ p v q ≡ q v p [ p ^ q ≡ q ^ p

As follows [ (p ^ q) ^ r ≡ p ^ (q ^ r) [ (p v q) v r ≡ p v (q ^ r)

As follows [ (p v (q ^ r)) ≡ (p v q) ^ (p v r) [ (p ^ (q v r)) ≡ (p ^ q) v (p ^ r)

As follows [ p v (p ^ q) ≡ p [ p ^ (p v q) ≡ p

A compound proposition is satisfiable if there is an assignment of truth values to its variables that make it true

When no assignment that makes a compound proposition true exists, then it’s unsatisfiable. A compound proposition is unsatisfiable if an only if its negation is a tautology.

Made up of variables (x, y, z, etc.), predicates (P(x), M(x), etc.), and quantifiers (∀, ∃, etc.). It is a Boolean-valued function P: X → {true, false} and is called the predicate on X.

Represented as P(x), they become propositions (and have truth values) when their variables are each replaced by a value from the domain of the quantifier

Connectives from propositional logic carry over to predicate logic. You combine 2 or more propositional functions to form propositional expressions.

Quantifiers are used to express the extent to which a predicate is true over a range of elements. In English, some of these terms used for quantification are all, some, many, none, and few. In logic, ∀ and ∃ are used. The quantifiers ∀ and ∃ also have higher precedence than all the logical operators. The truth value of quantified expressions depends on both the propositional function P(x) and on the domain U.

Asserts that the propositional P(x) is true for all values x in the domain. It is denoted as ∀; ∀xP(x) reads as “for every x P(x)”.

Asserts that there exists an element x in the domain which that P(x). The notation used for this is ∃ and ∃xP(x) reads as “there exists an element x in the domain such that P(x)”

Denoted by ∃!x P(x), it means that P(x) is true for one and only one x in the domain. It can also be expressed as ∃x(P(x)∧∀y(P(y)→y=x))

Statements involving predicates and quantifiers are logically equivalent if and only if they have the same truth value.

Rules of negations for quantifiers [ ¬∃xP(x) ≡ ∀x¬P(x) [ ¬∀xP(x) ≡ ∃x¬P(x)

Predicate logic is used for specifying properties that systems must satisfy.

There are premises and a conclusion statements in an argument.

Necessary to express the meaning of logic in various fields such as English and computer science and mathematics concepts.

Quantified expressions are read from left to right. [ Negation of Nested Quantifiers - The steps do doing this is as follows: [ Use quantifiers to express a statement [ Use De Morgan’s Laws to move the negation as far inward as possible

They are the essential building blocks in the construction of valid arguments. Propositional Logic uses inference rules. For predicate logic, they use inference rules for propositional logic as well as additional interference rules to handle variables and quantifiers.

In propositional logic, it is a sequence of propositions

The statements in an argument aside from the last statement

The last statement in an argument

As follows (p ^ (p → q)) → q

As follows (¬q ^ (p → q)) → ¬p

As follows ((p → q) ^ (q → r)) → (p → r)

As follows (¬p ^ (p v q)) → q

As follows p → (p v q)

As follows (p ^ q) → q

As follows ((p) ^ (q)) → (p ^ q)

As follows ((¬p v r) ^ (p v q)) → (q v r)

A sequence of statements where each statement is either a premise or follows from previous statements by rules of inference. The last statement is the conclusion.

It’s a generalization of simplification defined on the whole domain [ ∀xP(x) = P(c)

It’s a generalization of conjunction with the propositional value for all variables in the domain being true → universal quantifier true [ P(c) for an arbitrary c = ∀xP(x)

As follows [ ∃xP(x) = P(c) for some element c

As follows [ P(c) for some element c = ∃xP(x)

Combines universal instantiation and modus ponens into one rule [ ∀x(P(x) → Q(x)) [ P(a) [ = Q(a)

A valid argument that establishes the truth of a statement

A statement that can be shown to be true using definitions, other theorems, axioms, rules of interference. Less important theorems are sometimes called propositions.

A ‘helping theorem’ or a result which is needed to prove a theorem

A result which follows directly from a theorem

A statement that is being proposed to be true. Once a proof of a conjecture is found, it becomes a theorem. It may turn out to be false.

Assume that p is true. Use rules of inference, axioms, and logical equivalences to show that q must also be true.

Assume ¬q and show ¬p is also true. This is sometimes called the indirect proof method. If we give a direct proof of ¬q → ¬p, then we have a proof of p ¬p → q [ p → q ≡ ¬q → ¬p

To prove p, assume ¬p and derive a contradiction such as p ^ ¬p (an indirect form of proof). Since we have shown that ¬p → F is true, it follows that the contrapositive T → p also holds.

To prove a theorem that is a biconditional statement, we show that p → q and q → p are both true