## UNIT-I

# Mathematical Logic

## Statements and notations:

A proposition or statement is a declarative sentence that is either true or false (but not both). For instance, the following are propositions: “Paris is in France” (true), “London is in Denmark” (false), “2 < 4” (true), “4 = 7 (false)”. However the following are not propositions: “what is your name?” (this is a question), “do your homework” (this is a command), “this sentence is false” (neither true nor false), “x is an even number” (it depends on what x represents),

“Socrates” (it is not even a sentence). The truth or falsehood of a proposition is called its truth value.

## Connectives:

Connectives are used for making compound propositions. The main ones are the following (p and q represent given propositions):

| Name | Represented | Meaning |
|---|---|---|
| Negation | $\neg p$ | “not p” |
| Conjunction | $p \wedge q$ | “p and q” |
| Disjunction | $p \vee q$ | “p or q (or both)” |
| Exclusive Or | $p \oplus q$ | “either p or q, but not both” |
| Implication | $p \rightarrow q$ | “if p then q” |
| Biconditional | $p \leftrightarrow q$ | “p if and only if q” |

## Truth Tables:

### Logical identity

Logical identity is an operation on one logical value, typically the value of a proposition that produces a value of *true* if its operand is true and a value of *false* if its operand is false.

The truth table for the logical identity operator is as follows:

| Logical Identity | |
|---|---|
| ***p*** | ***p*** |
| T | T |
| F | F |

## Logical negation

Logical negation is an operation on one logical value, typically the value of a proposition that produces a value of *true* if its operand is false and a value of *false* if its operand is true.

The truth table for **NOT p** (also written as **¬p** or **~p**) is as follows:

| Logical Negation | |
|---|---|
| ***p*** | **¬*p*** |
| T | F |
| F | T |

# Binary operations

## Truth table for all binary logical operators

Here is a truth table giving definitions of all 16 of the possible truth functions of 2 binary variables (P,Q are thus boolean variables):

| ***P*** | ***Q*** | | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **T** | **T** | | F | F | F | F | F | F | F | F | T | T | T | T | T | T | T | T |
| **T** | **F** | | F | F | F | F | T | T | T | T | F | F | F | F | T | T | T | T |
| **F** | **T** | | F | F | T | T | F | F | T | T | F | F | T | T | F | F | T | T |
| **F** | **F** | | F | T | F | T | F | T | F | T | F | T | F | T | F | T | F | T |

where T = true and F = false.

Key:

0, false, Contradiction

1, NOR, Logical NOR

2, Converse nonimplication

3, **¬p**, Negation

4, Material nonimplication

5, **¬q**, Negation

6, XOR, Exclusive disjunction

7, NAND, Logical NAND

8, AND, Logical conjunction

9, XNOR, If and only if, Logical

biconditional 10, **q**, Projection function

11, if/then, Logical implication

12, **p**, Projection function

13, then/if, Converse implication

14, OR, Logical disjunction

15, true, Tautology

Logical operators can also be visualized using Venn diagrams.

## Logical conjunction

Logical conjunction is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if both of its operands are true.

**The truth table for p AND q (also written as p ∧ q, p & q, or p q) is as follows:**

| **Logical Conjunction** | | |
|---|---|---|
| ***p*** | ***q*** | *p ∧ q* |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

In ordinary language terms, if both *p* and *q* are true, then the conjunction *p* ∧ *q* is true. For all other assignments of logical values to *p* and to *q* the conjunction *p* ∧ *q* is false.
It can also be said that if *p*, then *p* ∧ *q* is *q*, otherwise *p* ∧ *q* is *p*.

## Logical disjunction

Logical disjunction is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if at least one of its operands is true.

**The truth table for p OR q (also written as p ∨ q, p || q, or p + q) is as follows:**

| Logical Disjunction | | |
|---|---|---|
| ***p*** | ***q*** | ***p ∨ q*** |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

## Logical implication

Logical implication and the material conditional are both associated with an operation on two logical values, typically the values of two propositions, that produces a value of *false* just in the singular case the first operand is true and the second operand is false.The truth table associated with the material conditional **if p then q** (symbolized as **p → q**) and the logical implication **p implies q** (symbolized as **p ⇒ q**) is as follows:

| Logical Implication | | |
|---|---|---|
| ***p*** | ***q*** | ***p → q*** |
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

## Logical equality

Logical equality (also known as biconditional) is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if both operands are false or both operands are true.The truth table for **p XNOR q** (also written as **p ↔ q** ,**p = q**, or **p ≡ q**) is as follows:

| Logical Equality | | |
|---|---|---|
| ***p*** | ***q*** | ***p ≡ q*** |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

## Exclusive disjunction

Exclusive disjunction is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if one but not both of its operands is true.The truth table for **p XOR q** (also written as **p ⊕ q**, or **p ≠ q**) is as follows:

| Exclusive Disjunction | | |
|---|---|---|
| ***p*** | ***q*** | *p ⊕ q* |
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | F |

## Logical NAND

The logical NAND is an operation on two logical values, typically the values of two propositions, that produces a value of *false* if both of its operands are true. In other words, it produces a value of *true* if at least one of its operands is false.The truth table for **p NAND q** (also written as **p ↑ q** or **p | q**) is as follows:

| **Logical NAND** | | |
|---|---|---|
| ***p*** | ***q*** | ***p* ↑ *q*** |
| T | T | F |
| T | F | T |
| F | T | T |
| F | F | T |

It is frequently useful to express a logical operation as a compound operation, that is, as an operation that is built up or composed from other operations. Many such compositions are possible, depending on the operations that are taken as basic or "primitive" and the operations that are taken as composite or "derivative".In the case of logical NAND, it is clearly expressible as a compound of NOT and AND.The negation of a conjunction: $\neg(p \wedge q)$, and the disjunction of negations: $(\neg p) \vee (\neg q)$ can be tabulated as follows:

| ***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 |

## Logical NOR

The logical NOR is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if both of its operands are false. In other words, it produces a value of *false* if at least one of its operands is true. ↓ is also known as the Peirce arrow after its inventor, Charles Sanders Peirce, and is a Sole sufficient operator.

The truth table for **p NOR q** (also written as **p ↓ q** or **p ⊥ q**) is as follows:

| **Logical NOR** | | |
|---|---|---|
| ***p*** | ***q*** | ***p ↓ q*** |
| T | T | F |
| T | F | F |
| F | T | F |
| F | F | T |

The negation of a disjunction ¬(*p* ∨ *q*), and the conjunction of negations (¬*p*) ∧ (¬*q*) can be tabulated as follows:

| $p$ | $q$ | $p \vee q$ | $\neg(p \vee q)$ | $\neg p$ | $\neg q$ | $(\neg p) \wedge (\neg q)$ |
|---|---|---|---|---|---|---|
| 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 |

Inspection of the tabular derivations for NAND and NOR, under each assignment of logical values to the functional arguments $p$ and $q$, produces the identical patterns of functional values for $\neg(p \wedge q)$ as for $(\neg p) \vee (\neg q)$, and for $\neg(p \vee q)$ as for $(\neg p) \wedge (\neg q)$. Thus the first and second expressions in each pair are logically equivalent, and may be substituted for each other in all contexts that pertain solely to their logical values.

This equivalence is one of De Morgan's laws.

The truth value of a compound proposition depends only on the value of its components.

Writing F for "false" and T for "true", we can summarize the meaning of the connectives in the following way:

| p | q | ¬p | p ∧ q | p ∨ q | p ⊕ q | p → q | p ↔ q |
|---|---|---|---|---|---|---|---|
| T | T | F | T | T | F | T | T |
| T | F | F | F | T | T | F | F |
| F | T | T | F | T | T | T | F |
| F | F | T | F | F | F | T | T |

**Note that ∨ represents a non-exclusive or, i.e., p ∨ q is true when any ofp, q is true and also when both are true. On the other hand ⊕ represents an exclusive or, i.e., p ⊕ q is true only when exactly one of p and q is true.**

## Well formed formulas(wff):

Not all strings can represent propositions of the predicate logic. Those which produce a proposition when their symbols are interpreted must follow the rules given below, and they are called wffs(well-formed formulas) of the first order predicate logic.

**Rules for constructing Wffs**

A predicate name followed by a list of variables such as P(x, y), where P ispredicate name, and x and y are variables, is called an atomic formula.

A well formed formula of predicate calculus is obtained by using the following rules.

1. An atomic formula is a wff.
2. If A is a wff, then 7A is also a wff.
3. If A and B are wffs, then (A V B), (A ^ B), (A → B) and (A D B),
4. If A is a wff and x is a any variable, then (x)A and ($x)A are wffs.
5. Only those formulas obtained by using (1) to (4) are wffs.

Since we will be concerned with only wffs, we shall use the term formulas for wff. We shall follow the same conventions regarding the use of parentheses as was done in the case of statement formulas.

**Wffs are constructed using the following rules:**

1. *True* and *False* are wffs.
2. Each propositional constant (i.e. specific proposition), and each propositional variable (i.e. a variable representing propositions) are wffs.
3. Each atomic formula (i.e. a specific predicate with variables) is a wff.
4. If *A, B,* and *C* are wffs, then so are $\neg A$, $(A \wedge B)$, $(A \vee B)$, $(A \rightarrow B)$, and $(A \leftrightarrow B)$.
5. If *x* is a variable (representing objects of the universe of discourse), and *A* is a wff, then so are $\forall x\ A$ and $\exists x\ A$ .

For example, "The capital of Virginia is Richmond." is a specific proposition. Hence it is a wff by Rule 2.

Let B be a predicate name representing "being blue" and let x be a variable. Then B(x) is an atomic formula meaning "x is blue". Thus it is a wff by Rule 3. above. By applying Rule 5. to B(x), ∀xB(x) is a wff and so is ∃xB(x). Then by applying Rule 4. to them ∀x B(x) ∧∃x B(x) is seen to be a wff. Similarly if R is a predicate name representing "being round". Then R(x) is an atomic formula. Hence it is a wff. By applying Rule 4 to B(x) and R(x), a wff B(x) ∧R(x) is obtained.

In this manner, larger and more complex wffs can be constructed following the rules given above.

Note, however, that strings that can not be constructed by using those rules are not wffs. For example, ∀xB(x)R(x), and B( ∃x ) are NOT wffs, NOR are B( R(x) ), and B( ∃x R(x) ) . More examples: To express the fact that Tom is taller than John, we can use the atomic formula ***taller*(Tom, John)**, which is a wff. This wff can also be part of some compound statement such as ***taller*(Tom, John) ∧¬*taller*(John, Tom)**, which is also a wff. ***If x is a variable representing people in the world, then taller(x,Tom), ∀x taller(x*,Tom), *∃x taller(x*,Tom), *∃x ∀y taller(x,y)*** are all wffs among others. However, *taller(∃ x*,John) and *taller*(Tom ∧Mary, Jim), for example, are **NOT** wffs.

## Tautology, Contradiction, Contingency:

A proposition is said to be a tautology if its truth value is T for any assignment of truth values to its components. Example: The proposition p ∨ ¬p is a tautology.

A proposition is said to be a contradiction if its truth value is F for any assignment of truth values to its components. Example: The proposition p ∧ ¬p is a contradiction.

A proposition that is neither a tautology nor a contradiction is called a contingency.

| p | ¬p | p ∨ ¬p | p ∧ ¬p |
|---|---|---|---|
| T | F | T | F |
| T | F | T | F |
| F | T | T | F |
| F | T | T | F |

## Equivalence Implication:

We say that the statements *r* and *s* are logically equivalent if their truth tables are identical. For example the truth table of $\neg p \vee q$

| $p$ | $q$ | $\neg p \vee q$ |
|---|---|---|
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

shows that $\neg p \vee q$ is equivalent to $p \to q$. It is easily shown that the statements *r* and *s* are equivalent if and only if $r \leftrightarrow s$ is a tautology.

## Normal forms:

Let A(P1, P2, P3, …, Pn) be a statement formula where P1, P2, P3, …, Pn are the atomic variables. If A has truth value T for all possible assignments of the truth values to the variables P1, P2, P3, …, Pn , then A is said to be a tautology. If A has truth value F, then A is said to be identically false or a contradiction.

### Disjunctive Normal Forms

A product of the variables and their negations in a formula is called an elementary product. A sum of the variables and their negations is called an elementary sum. That is, a sum of elementary products is called a disjunctive normal form of the given formula.

Example:

$$A \tag{1}$$

$$(A \wedge B) \vee (!\, A \wedge C) \tag{2}$$

$$(A \wedge B \wedge !\, A) \vee (C \wedge !\, B) \vee (A \wedge !\, C) \tag{3}$$

$$A \wedge B \tag{4}$$

$$A \vee (B \wedge C), \tag{5}$$

### Conjunctive Normal Forms

A formula which is equivalent to a given formula and which consists of a product of elementary sums is called a conjunctive normal form of a given formula.

Example:

$$A \tag{1}$$

$$(A \vee B) \wedge (!\, A \vee C) \tag{2}$$

$$A \vee B \tag{3}$$

$$A \wedge (B \vee C), \tag{4}$$

# Predicates

## Predicative logic:

A predicate or propositional function is a statement containing variables. For instance "x + 2 = 7", "X is American", "x < y", "p is a prime number" are predicates. The truth value of the predicate depends on the value assigned to its variables. For instance if we replace x with 1 in the predicate "x + 2 = 7" we obtain "1 + 2 = 7", which is false, but if we replace it with 5 we get "5 + 2 = 7", which is true. We represent a predicate by a letter followed by the variables enclosed between parenthesis: P (x), Q(x, y), etc. An example for P (x) is a value of x for which P (x) is true. A counterexample is a value of x for which P (x) is false. So, 5 is an example for "x + 2 = 7", while 1 is a counterexample. Each variable in a predicate is assumed to belong to a universe (or domain) of discourse, for instance in the predicate "n is an odd integer" 'n' represents an integer, so the universe of discourse of n is the set of all integers. In "X is American" we may assume that X is a human being, so in this case the universe of discourse is the set of all human beings.

## Free & Bound variables:

Let's now turn to a rather important topic: the distinction between free variable s and **bound variable**s.

Have a look at the following formula:

$$\neg(\mathrm{THERAPIST}(x) \vee \forall x(\mathrm{MORON}(x) \wedge \forall y \mathrm{PERSON}(y)))$$

The first occurrence of x is *free*, whereas the second and third occurrences of x are *bound*, namely by the first occurrence of the quantifier ∀. The first and second occurrences of the variable y are also bound, namely by the second occurrence of the quantifier ∀.

Informally, the concept of a *bound variable* can be explained as follows: Recall that quantifications are generally of the form:

$\forall x \varphi$

or

$\exists x \varphi$

where $x$ may be any variable. Generally, all occurences of this variable within the quantification are bound. But we have to distinguish two cases. Look at the following formula to see why:

$\exists x(\text{MAN}(x) \wedge (\forall x \text{WALKS}(x)) \wedge \text{HAPPY}(x))$

1. $x$ may occur within another, embedded, quantification $\forall x \psi$ or $\exists x \psi$, such as the $x$ in $\text{WALKS}(x)$ in our example. Then we say that it is bound by the quantifier of this embedded quantification (and so on, if there's another embedded quantification over $x$ within $\psi$).
2. Otherwise, we say that it is bound by the top-level quantifier (like all other occurences of $x$ in our example).

Here's a full formal simultaneous definition of *free* and *bound*:

1. Any occurrence of any variable is free in any atomic formula.
2. No occurrence of any variable is bound in any atomic formula.
3. If an occurrence of any variable is free in $\varphi$ or in $\psi$, then that same occurrence is free in $\neg\varphi$, $(\varphi \rightarrow \psi)$, $(\varphi \vee \psi)$, and $(\varphi \wedge \psi)$.
4. If an occurrence of any variable is bound in $\varphi$ or in $\psi$, then that same occurrence is bound in $\neg\varphi$, $(\varphi \rightarrow \psi)$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$. Moreover, that same occurrence is bound in $\forall y \varphi$ and $\exists y \varphi$ as well, for any choice of variable y.
5. In any formula of the form $\forall y \varphi$ or $\exists y \varphi$ (where y can be any variable at all in this case) the occurrence of y that immediately follows the initial quantifier symbol is bound.
6. If an occurrence of a variable x is free in $\varphi$, then that same occurrence is free in $\forall y \varphi$ and $\exists y \varphi$, for any variable y distinct from x. On the other hand, all occurrences of x that are free in $\varphi$, are bound in $\forall x \varphi$ and in $\exists x \varphi$.

If a formula contains no occurrences of free variables we call it a ***sentence***.

## Rules of inference:

The two rules of inference are called rules P and T.

Rule P: A premise may be introduced at any point in the derivation.

Rule T: A formula S may be introduced in a derivation if s is tautologically implied by any one or more of the preceding formulas in the derivation.

Before proceeding the actual process of derivation, some important list of implications and equivalences are given in the following tables.

***Implications***

| | | |
|---|---|---|
| I1 | P^Q =>P | } Simplification |
| I2 | PQ^ =>Q | |
| I3 | P=>PVQ | } Addition |
| I4 | Q =>PVQ | |
| I5 | 7P => P→ Q | |
| I6 | Q => P→ Q | |
| I7 | 7(P→Q) =>P | |
| I8 | 7(P → Q) => 7Q | |
| I9 | P, Q => P ^ Q | |
| I10 | 7P, PVQ => Q | ( disjunctive syllogism) |
| I11 | P, P→ Q => Q | ( modus ponens ) |
| I12 | 7Q, P → Q => 7P | (modus tollens ) |
| I13 | P → Q, Q → R => P → R | ( hypothetical syllogism) |
| I14 | P V Q, P → Q, Q → R => R | (dilemma) |

***Equivalences***

| | | |
|---|---|---|
| E1 | 77P <=>P | |
| E2 | P ^ Q <=> Q ^ P | } Commutative laws |
| E3 | P V Q <=> Q V P | |
| E4 | (P ^ Q) ^ R <=> P ^ (Q ^ R) | } Associative laws |
| E5 | (P V Q) V R <=> PV (Q V R) | |
| E6 | P ^ (Q V R) <=> (P ^ Q) V (P ^ R) | } Distributive laws |
| E7 | P V (Q ^ R) <=> (P V Q) ^ (PVR) | |
| E8 | 7(P ^ Q) <=> 7P V7Q | |
| E9 | 7(P V Q) <=>7P ^ 7Q | } De Morgan's laws |
| E10 | P V P <=> P | |
| E11 | P ^ P <=> P | |

E12 R V (P ^ 7P) <=>R

E13 R ^ (P V 7P) <=>R

E14 R V (P V 7P) <=>T

E15 R ^ (P ^ 7P) <=>F

E16 P → Q <=> 7P V Q

E17 7 (P→ Q) <=> P ^ 7Q

E18 P → Q <=> 7Q → 7P

E19 P → (Q → R) <=> (P ^ Q) → R

E20 7(PD Q) <=> P D 7Q

E21 PDQ <=> (P → Q) ^ (Q → P)

E22 (PDQ) <=> (P ^ Q) V (7 P ^ 7Q)

Example 1.Show that R is logically derived from P → Q, Q → R, and P

| Solution. | | | | |
|---|---|---|---|---|
| | {1} | (1) | P → Q | Rule P |
| | {2} | (2) | P | Rule P |
| | {1, 2} | (3) | Q | Rule (1), (2) and I11 |
| | {4} | (4) | Q → R | Rule P |
| | {1, 2, 4} | (5) | R | Rule (3), (4) and I11. |

Example 2.Show that S V R tautologically implied by ( P V Q) ^ ( P → R) ^ ( Q → S ).

| Solution . | | | | |
|---|---|---|---|---|
| | {1} | (1) | P V Q | Rule P |
| | {1} | (2) | 7P → Q | T, (1), E1 and E16 |
| | {3} | (3) | Q → S | P |
| | {1, 3} | (4) | 7P → S | T, (2), (3), and I13 |
| | {1, 3} | (5) | 7S → P | T, (4), E13 and E1 |
| | {6} | (6) | P → R | P |
| | {1, 3, 6} | (7) | 7S → R | T, (5), (6), and I13 |
| | {1, 3, 6) | (8) | S V R | T, (7), E16 and E1 |

Example 3. Show that 7Q, P→ Q => 7P

| Solution . | | | | |
|---|---|---|---|---|
| | {1} | (1) | P → Q | Rule P |
| | {1} | (2) | 7P → 7Q | T, and E 18 |

{3} (3) 7Q P

{1, 3} (4) 7P T, (2), (3), and I11 .

Example 4 .Prove that R ∧ ( P V Q ) is a valid conclusion from the premises PVQ ,
Q → R, P → M and 7M.

| | | | |
|---|---|---|---|
| Solution . | {1} | (1) P → M | P |
| | {2} | (2) 7M | P |
| | {1, 2} | (3) 7P | T, (1), (2), and I12 |
| | {4} | (4) P V Q | P |
| | {1, 2 , 4} | (5) Q | T, (3), (4), and I10. |
| | {6} | (6) Q → R | P |
| | {1, 2, 4, 6} | (7) R | T, (5), (6) and I11 |
| | {1, 2, 4, 6} | (8) R ∧ (PVQ) | T, (4), (7), and I9. |

**There is a third inference rule, known as rule CP or rule of *conditional proof*.**

**Rule CP**: If we can derives s from R and a set of premises , then we can derive R → S from the set of premises alone.

Note. 1. Rule CP follows from the equivalence E10 which states that
( P ∧ R ) → S óP → (R → S).

2. Let P denote the conjunction of the set of premises and let R be any formula The above equivalence states that if R is included as an additional premise and S is derived from P ∧ R then R → S can be derived from the premises P alone.

3. Rule CP is also called the *deduction theorem* and is generally used if the conclusion is of the form R → S. In such cases, R is taken as an additional premise and S is derived from the given premises and R.

Example 5 .Show that R → S can be derived from the premises
P → (Q → S), 7R V P , and Q.

Solution.

| | | |
|---|---|---|
| {1} | (1) 7R V P | P |
| {2} | (2) R | P, assumed premise |
| {1, 2} | (3) P | T, (1), (2), and I10 |
| {4} | (4) P → (Q → S) | P |
| {1, 2, 4} | (5) Q → S | T, (3), (4), and I11 |
| {6} | (6) Q | P |
| {1, 2, 4, 6} | (7) S | T, (5), (6), and I11 |
| {1, 4, 6} | (8) R → S | CP. |

Example 6.Show that P → S can be derived from the premises, 7P V Q, 7Q V R, and R → S .

Solution.

| | | | |
|---|---|---|---|
| {1} | (1) | 7P V Q | P |
| {2} | (2) | P | P, assumed premise |
| {1, 2} | (3) | Q | T, (1), (2) and I11 |
| {4} | (4) | 7Q V R | P |
| {1, 2, 4} | (5) | R | T, (3), (4) and I11 |
| {6} | (6) | R → S | P |
| {1, 2, 4, 6} | (7) | S | T, (5), (6) and I11 |
| {2, 7} | (8) | P → S | CP |

Example 7. " If there was a ball game , then traveling was difficult. If they arrived on time, then traveling was not difficult. They arrived on time. Therefore, there was no ball game". Show that these statements constitute a valid argument.

Solution. Let P: There was a ball game

Q: Traveling was difficult.

R: They arrived on time.

Given premises are: P → Q, R → 7Q and R conclusion is: 7P

| | | |
|---|---|---|
| {1} | (1) $P \to Q$ | P |
| {2} | (2) $R \to 7Q$ | P |
| {3} | (3) R | P |
| {2, 3} | (4) 7Q | T, (2), (3), and I11 |
| {1, 2, 3} | (5) 7P | T, (2), (4) and I12 |

## Consistency of premises:

**Consistency**

A set of formulas H1, H2, …, Hm is said to be consistent if their conjunction has the truth value T for some assignment of the truth values to be atomic appearing in H1, H2, …, Hm.

**Inconsistency**

If for every assignment of the truth values to the atomic variables, at least one of the formulas H1, H2, … Hm is false, so that their conjunction is identically false, then the formulas H1, H2, …, Hm are called inconsistent.

A set of formulas H1, H2, …, Hm is inconsistent, if their conjunction implies a contradiction, that is H1 $\wedge$ H2 $\wedge$ … $\wedge$ Hm => R $\wedge$ 7R

Where R is any formula. Note that R $\wedge$ 7R is a contradiction and it is necessary and sufficient that H1, H2, …,Hm are inconsistent the formula.

**Indirect method of proof**

In order to show that a conclusion C follows logically from the premises H1, H2,…, Hm, we assume that C is false and consider 7C as an additional premise. If the new set of premises is inconsistent, so that they imply a contradiction, then the assumption that 7C is true does not hold simultaneously with H1 $\wedge$ H2 $\wedge$ ..… $\wedge$ Hm being true. Therefore, C is true whenever H1 $\wedge$ H2 $\wedge$ ..… $\wedge$ Hm is true. Thus, C follows logically from the premises H1, H2 ….., Hm.

Example 8 Show that 7(P $\wedge$ Q) follows from 7P $\wedge$ 7Q.

Solution.

We introduce 77 (P^ Q) as an additional premise and show that this additional premise leads to a contradiction.

| | | | |
|---|---|---|---|
| {1} | (1) | 77(P^ Q) | P assumed premise |
| {1} | (2) | P^ Q | T, (1) and E1 |
| {1} | (3) | P | T, (2) and I1 |
| {1} | {4) | 7P^7Q | P |
| {4} | (5) | 7P | T, (4) and I1 |
| {1, 4} | (6) | P^ 7P | T, (3), (5) and I9 |

Here (6) P^ 7P is a contradiction. Thus {1, 4} viz. 77(P^ Q) and 7P^ 7Q leads to a contradiction P ^ 7P.

Example 9Show that the following premises are inconsistent.

1. If Jack misses many classes through illness, then he fails high school.
2. If Jack fails high school, then he is uneducated.
3. If Jack reads a lot of books, then he is not uneducated.
4. Jack misses many classes through illness and reads a lot of books.

*Solution.*

P: Jack misses many classes.

Q: Jack fails high school.

R: Jack reads a lot of books.

S: Jack is uneducated.

The premises are P→ Q, Q → S, R→ 7S and P^ R

| | | | |
|---|---|---|---|
| {1} | (1) | P→Q | P |
| {2} | (2) | Q→ S | P |
| {1, 2} | (3) | P → S | T, (1), (2) and I13 |
| {4} | (4) | R→ 7S | P |
| {4} | (5) | S → 7R | T, (4), and E18 |
| {1, 2, 4} | (6) | P→7R | T, (3), (5) and I13 |
| {1, 2, 4} | (7) | 7PV7R | T, (6) and E16 |
| {1, 2, 4} | (8) | 7(P^R) | T, (7) and E8 |

{9}  (9)P∧ R  P

{1, 2, 4, 9)  (10)  (P∧ R) ∧ 7(P∧ R)  T, (8), (9) and I9

The rules above can be summed up in the following table. The "Tautology" column shows how to interpret the notation of a given rule.

| **Rule of inference** | **Tautology** | **Name** |
|---|---|---|
| $\begin{array}{r} p \\ \hline \therefore\ p \vee q \end{array}$ | $p \rightarrow (p \vee q)$ | Addition |
| $\begin{array}{r} p \wedge q \\ \hline \therefore\ p \end{array}$ | $(p \wedge q) \rightarrow p$ | Simplification |
| $\begin{array}{r} p \\ q \\ \hline \therefore\ p \wedge q \end{array}$ | $((p) \wedge (q)) \rightarrow (p \wedge q)$ | Conjunction |
| $\begin{array}{r} p \\ p \rightarrow q \\ \hline \therefore\ q \end{array}$ | $((p \wedge (p \rightarrow q)) \rightarrow q$ | Modus ponens |
| $\begin{array}{r} \neg q \\ p \rightarrow q \\ \hline \therefore\ \neg p \end{array}$ | $((\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$ | Modus tollens |
| $\begin{array}{r} p \rightarrow q \\ q \rightarrow r \\ \hline \therefore\ p \rightarrow r \end{array}$ | $((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$ | Hypothetical syllogism |
| $\begin{array}{r} p \vee q \\ \neg p \\ \hline \therefore\ q \end{array}$ | $((p \vee q) \wedge \neg p) \rightarrow q$ | Disjunctive syllogism |
| $\begin{array}{r} p \vee q \\ \neg p \vee r \\ \hline \therefore\ q \vee r \end{array}$ | $((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$ | Resolution |

## Example 1

Let us consider the following assumptions: "If it rains today, then we will not go on a canoe today. If we do not go on a canoe trip today, then we will go on a canoe trip tomorrow. Therefore (Mathematical symbol for "therefore" is $\therefore$), if it rains today, we will go on a canoe trip tomorrow. To make use of the rules of inference in the above table we let $p$ be the proposition "If it rains today", $q$ be " We will not go on a canoe today" and let $r$ be "We will go on a canoe trip tomorrow". Then this argument is of the form:

$$\begin{array}{rl} & p \rightarrow q \\ & q \rightarrow r \\ \hline \therefore & p \rightarrow r \end{array}$$

## Example 2

Let us consider a more complex set of assumptions: "It is not sunny today and it is colder than yesterday". "We will go swimming only if it is sunny", "If we do not go swimming, then we will have a barbecue", and "If we will have a barbecue, then we will be home by sunset" lead to the conclusion "We will be home before sunset." Proof by rules of inference: Let $p$ be the proposition "It is sunny this today", $q$ the proposition "It is colder than yesterday", $r$ the proposition "We will go swimming", $s$ the proposition "We will have a barbecue", and $t$ the proposition "We will be home by sunset". Then the hypotheses become $\neg p \wedge q, r \rightarrow p, \neg r \rightarrow s$ and $s \rightarrow t$. Using our intuition we conjecture that the conclusion might be $t$. Using the Rules of Inference table we can proof the conjecture easily:

| Step | Reason |
|---|---|
| 1. $\neg p \wedge q$ | Hypothesis |
| 2. $\neg p$ | Simplification using Step 1 |

3. $r \rightarrow p$ Hypothesis

4. $\neg r$ Modus tollens using Step 2 and 3

5. $\neg r \rightarrow s$ Hypothesis

6. $s$ Modus ponens using Step 4 and 5

7. $s \rightarrow t$ Hypothesis

8. $t$ Modus ponens using Step 6 and 7

## Proof of contradiction:

The "Proof by Contradiction" is also known as reductio ad absurdum, which is probably Latin for "reduce it to something absurd".

Here's the idea:

1. Assume that a given proposition is untrue.
2. Based on that assumption reach two conclusions that contradict each other.

This is based on a classical formal logic construction known as Modus Tollens: If P implies Q and Q is false, then P is false. In this case, Q is a proposition of the form (R and not R) which is always false. P is the negation of the fact that we are trying to prove and if the negation is not true then the original proposition must have been true. If computers are not "not stupid" then they are stupid. (I hear that "stupid computer!" phrase a lot around here.)

### Example:

Lets prove that there is no largest prime number (this is the idea of Euclid's original proof). Prime numbers are integers with no exact integer divisors except 1 and themselves.

1. To prove: "There is no largest prime number" by contradiction.
2. Assume: There is a largest prime number, call it p.
3. Consider the number N that is one larger than the product of all of the primes smaller than or equal to p. N=1*2*3*5*7*11...*p + 1. Is it prime?
4. N is at least as big as p+1 and so is larger than p and so, by Step 2, cannot be prime.
5. On the other hand, N has no prime factors between 1 and p because they would all leave a remainder of 1. It has no prime factors larger than p because Step 2 says that there are no primes larger than p. So N has no prime factors and therefore must itself be prime (see note below).

We have reached a contradiction (N is not prime by Step 4, and N is prime by Step 5) and therefore our original assumption that there is a largest prime must be false.

Note: The conclusion in Step 5 makes implicit use of one other important theorem: The Fundamental Theorem of Arithmetic: Every integer can be uniquely represented as the product of primes. So if N had a composite (i.e. non-prime) factor, that factor would itself have prime factors which would also be factors of N.

## Automatic Theorem Proving:

**Automatic Theorem Proving** (ATP) deals with the development of computer programs that show that some statement (the *conjecture*) is a *logical consequence* of a set of statements (the *axioms* and *hypotheses*). ATP systems are used in a wide variety of domains. For examples, a mathematician might prove the conjecture that groups of order two are commutative, from the axioms of group theory; a management consultant might formulate axioms that describe how organizations grow and interact, and from those axioms prove that organizational death rates decrease with age; a hardware developer might validate the design of a circuit by proving a conjecture that describes a circuit's performance, given axioms that describe the circuit itself; or a frustrated teenager might formulate the jumbled faces of a Rubik's cube as a conjecture and prove, from axioms that describe legal changes to the cube's configuration, that the cube can be rearranged to the solution state. All of these are tasks that can be performed by an ATP system, given an appropriate formulation of the problem as axioms, hypotheses, and a conjecture.

The **language** in which the conjecture, hypotheses, and axioms (generically known as *formulae*) are written is a logic, often classical 1st order logic, but possibly a non-classical logic and possibly a higher order logic. These languages allow a precise formal statement of the necessary information, which can then be manipulated by an ATP system. This formality is the underlying strength of ATP: there is no ambiguity in the statement of the problem, as is often the case when using a natural language such as English. Users have to describe the problem at hand precisely and accurately, and this process in itself can lead to a clearer understanding of the problem domain. This in turn allows the user to formulate their problem appropriately for submission to an ATP system.

The **proofs** produced by ATP systems describe how and why the conjecture follows from the axioms and hypotheses, in a manner that can be understood and agreed upon by everyone, even other computer programs. The proof output may not only be a convincing argument that the conjecture is a logical consequence of the axioms and hypotheses, it often also describes a process that may be implemented to solve some problem. For example, in the Rubik's cube example mentioned above, the proof would describe the sequence of moves that need to be made in order to solve the puzzle.

**ATP systems** are enormously powerful computer programs, capable of solving immensely difficult problems. Because of this extreme capability, their application and operation sometimes needs to be guided by an expert in the domain of application, in order to solve problems in a reasonable amount of time. Thus ATP systems, despite the name, are often used by domain experts in an interactive way. The interaction may be at a very detailed level, where the user guides the inferences made by the system, or at a much higher level where the user determines intermediate lemmas to be proved on the way to the proof of a conjecture. There is often a synergetic relationship between ATP system users and the systems themselves:

- The system needs a precise description of the problem written in some logical form,
- the user is forced to think carefully about the problem in order to produce an appropriate formulation and hence acquires a deeper understanding of the problem,
- the system attempts to solve the problem,
- if successful the proof is a useful output,

- if unsuccessful the user can provide guidance, or try to prove some intermediate result, or examine the formulae to ensure that the problem is correctly described,
- and so the process iterates.

ATP is thus a **technology** very suited to situations where a clear thinking domain expert can interact with a powerful tool, to solve interesting and deep problems. Potential ATP users need not be concerned that they need to write an ATP system themselves; there are many ATP systems readily available for use.