# logic

# Contents

# Conjunctive normal form

In Boolean logic, a formula is in **conjunctive normal form (CNF)** or **clausal normal form** if it is a conjunction of clauses, where a clause is a disjunction of literals; otherwise put, it is an AND of ORs. As a normal form, it is useful in automated theorem proving. It is similar to the product of sums form used in circuit theory.

All conjunctions of literals and all disjunctions of literals are in CNF, as they can be seen as conjunctions of one-literal clauses and conjunctions of a single clause, respectively. As in the disjunctive normal form (DNF), the only propositional connectives a formula in CNF can contain are and, or, and not. The not operator can only be used as part of a literal, which means that it can only precede a propositional variable or a predicate symbol.

In automated theorem proving, the notion "*clausal normal form*" is often used in a narrower sense, meaning a particular representation of a CNF formula as a set of sets of literals.

## Examples and counterexamples

All of the following formulas are **in CNF**:

$$\neg A \wedge (B \vee C)$$
$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$
$$A \vee B$$
$$A \wedge B$$

The last formula is in CNF because it can be seen as the conjunction of the two single-literal clauses $A$ and $B$. Incidentally, the last two formulae are also in disjunctive normal form.

The following formulas are **not in CNF**:

$$\neg (B \vee C)$$
$$(A \wedge B) \vee C$$
$$A \wedge (B \vee (D \wedge E)).$$

The above three formulas are respectively equivalent to the following three formulas that are **in CNF**:

$$\neg B \wedge \neg C$$
$$(A \vee C) \wedge (B \vee C)$$
$$A \wedge (B \vee D) \wedge (B \vee E).$$

## Conversion into CNF

Every propositional formula can be converted into an equivalent formula that is in CNF. This transformation is based on rules about logical equivalences: the double negative law, De Morgan's laws, and the distributive law.

Since all logical formulae can be converted into an equivalent formula in conjunctive normal form, proofs are often based on the assumption that all formulae are CNF. However, in some cases this conversion to CNF can lead to an exponential explosion of the formula. For example, translating the following non-CNF formula into CNF produces a formula with $2^n$ clauses:

$$(X_1 \wedge Y_1) \vee (X_2 \wedge Y_2) \vee \ldots \vee (X_n \wedge Y_n).$$

In particular, the generated formula is:

$$(X_1 \vee \cdots \vee X_{n-1} \vee X_n) \wedge (X_1 \vee \cdots \vee X_{n-1} \vee Y_n) \wedge \cdots \wedge (Y_1 \vee \cdots \vee Y_{n-1} \vee Y_n).$$

This formula contains $2^n$ clauses; each clause contains either $X_i$ or $Y_i$ for each $i$.

There exist transformations into CNF that avoid an exponential increase in size by preserving satisfiability rather than equivalence.[1][2] These transformations are guaranteed to only linearly increase the size of the formula, but introduce new variables. For example, the above formula can be transformed into CNF by adding variables $Z_1, \ldots, Z_n$ as follows:

$$(Z_1 \vee \cdots \vee Z_n) \wedge (\neg Z_1 \vee X_1) \wedge (\neg Z_1 \vee Y_1) \wedge \cdots \wedge (\neg Z_n \vee X_n) \wedge (\neg Z_n \vee Y_n).$$

An interpretation satisfies this formula only if at least one of the new variables is true. If this variable is $Z_i$, then both $X_i$ and $Y_i$ are true as well. This means that every model that satisfies this formula also satisfies the original one. On the other hand, only some of the models of the original formula satisfy this one: since the $Z_i$ are not mentioned in the original formula, their values are irrelevant to satisfaction of it, which is not the case in the last formula. This means that the original formula and the result of the translation are equisatisfiable but not equivalent.

An alternative translation, the Tseitin transformation, includes also the clauses $Z_i \vee \neg X_i \vee \neg Y_i$. With these clauses, the formula implies $Z_i \equiv X_i \wedge Y_i$; this formula is often regarded to "define" $Z_i$ to be a name for $X_i \wedge Y_i$.

## First-order logic

In first order logic, conjunctive normal form can be taken further to yield the clausal normal form of a logical formula, which can be then used to perform first-order resolution. In resolution-based automated theorem-proving, a CNF formula

$$( l_{11} \vee \cdots \vee l_{1n_1} ) \wedge \cdots \wedge ( l_{m1} \vee \cdots \vee l_{mn_m} )$$

, with $l_{ij}$ literals, is commonly represented as a set of sets

$$\{ \{ l_{11}, \ldots, l_{1n_1} \}, \ldots, \{ l_{m1}, \ldots, l_{mn_m} \} \}.$$

See below for an example.

## Computational complexity

An important set of problems in computational complexity involves finding assignments to the variables of a boolean formula expressed in Conjunctive Normal Form, such that the formula is true. The *k*-SAT problem is the problem of finding a satisfying assignment to a boolean formula expressed in CNF in which each disjunction contains at most *k* variables. 3-SAT is NP-complete (like any other *k*-SAT problem with *k*>2) while 2-SAT is known to have solutions in polynomial time. As a consequence,[3] the task of converting a formula into a DNF, preserving satisfiability, is NP-hard; dually, converting into CNF, preserving validity, is also NP-hard; hence equivalence-preserving conversion into DNF or CNF is again NP-hard.

Typical problems in this case involve formulas in "3CNF": conjunctive normal form with no more than three variables per conjunct. Examples of such formulas encountered in practice can be very large, for example with 100,000 variables and 1,000,000 conjuncts.

A formula in CNF can be converted into an equisatisfiable formula in "*k*CNF" (for k>=3) by replacing each conjunct with more than *k* variables $X_1 \vee \cdots \vee X_k \vee \cdots \vee X_n$ by two conjuncts $X_1 \vee \cdots \vee X_{k-1} \vee Z$ and $\neg Z \vee X_k \cdots \vee X_n$ with $Z$ a new variable, and repeating as often as necessary.

## Converting from first-order logic

To convert first-order logic to CNF:[4]

1. Convert to negation normal form.
   1. Eliminate implications and equivalences: repeatedly replace $P \rightarrow Q$ with $\neg P \lor Q$ ; replace $P \leftrightarrow Q$ with $(P \lor \neg Q) \land (\neg P \lor Q)$ . Eventually, this will eliminate all occurrences of $\rightarrow$ and $\leftrightarrow$ .
   2. Move NOTs inwards by repeatedly applying De Morgan's Law. Specifically, replace $\neg(P \lor Q)$ with $(\neg P) \land (\neg Q)$; replace $\neg(P \land Q)$ with $(\neg P) \lor (\neg Q)$; and replace $\neg\neg P$ with $P$; replace $\neg(\forall x P(x))$ with $\exists x \neg P(x)$; $\neg(\exists x P(x))$ with $\forall x \neg P(x)$. After that, a $\neg$ may occur only immediately before a predicate symbol.
2. Standardize variables
   1. For sentences like $(\forall x P(x)) \lor (\exists x Q(x))$ which use the same variable name twice, change the name of one of the variables. This avoids confusion later when dropping quantifiers later. For example, $\forall x[\exists y Animal(y) \land \neg Loves(x,y)] \lor [\exists y Loves(y,x)]$ is renamed to $\forall x[\exists y Animal(y) \land \neg Loves(x,y)] \lor [\exists z Loves(z,x)]$.
3. Skolemize the statement
   1. Move quantifiers outwards: repeatedly replace $P \land (\forall x Q(x))$ with $\forall x (P \land Q(x))$; replace $P \lor (\forall x Q(x))$ with $\forall x (P \lor Q(x))$; replace $P \land (\exists x Q(x))$ with $\exists x (P \land Q(x))$; replace $P \lor (\exists x Q(x))$ with $\exists x (P \lor Q(x))$. These replacements preserve equivalence, since the previous variable standardization step ensured that $x$ doesn't occur in $P$. After these replacements, a quantifier may occur only in the initial prefix of the formula, but never inside a $\neg$, $\land$, or $\lor$.
   2. Repeatedly replace $\forall x_1 \ldots \forall x_n\ \exists y\ P(y)$ with $\forall x_1 \ldots \forall x_n\ P(f(x_1,\ldots,x_n))$, where $f$ is a new $n$ -ary function symbol, a so-called "skolem function". This is the only step that preserves only satisfiability rather than equivalence. It eliminates all existential quantifiers.
4. Drop all universal quantifiers.
5. Distribute ORs inwards over ANDs: repeatedly replace $P \lor (Q \land R)$ with $(P \lor Q) \land (P \lor R)$.

As an example, the formula saying *"Who loves all animals, is in turn loved by someone"* is converted into CNF (and subsequently into clause form in the last line) as follows (highlighting replacement rule redices in red):

$$(\forall y\ Animal(y) \rightarrow Loves(x,y)) \rightarrow (\exists y\ Loves(y,x))$$

$$(\forall y\ \neg Animal(y) \lor Loves(x,y)) \rightarrow (\exists y\ Loves(y,x))$$

$$(\forall y\ \neg Animal(y) \lor Loves(x,y)) \lor (\exists y\ Loves(y,x))$$

$$(\exists y\ \neg(\neg Animal(y) \lor Loves(x,y))) \lor (\exists y\ Loves(y,x))$$

$$(\exists y\ \neg\neg Animal(y) \land \neg Loves(x,y)) \lor (\exists y\ Loves(y,x))$$

$$(\exists y\ Animal(y) \land \neg Loves(x,y)) \lor (\exists y\ Loves(y,x))$$

$$(\exists y\ Animal(y) \land \neg Loves(x,y)) \lor (\exists z\ Loves(z,x))$$

$$(\exists y\ Animal(y) \land \neg Loves(x,y)) \lor Loves(z,x)$$

$$\exists y\ (Animal(y) \land \neg Loves(x,y)) \lor Loves(z,x)$$

$$\exists y\ (Animal(y) \land \neg Loves(x,y)) \lor Loves(g(x),x)$$

$$(Animal(f(x)) \land \neg Loves(x,f(x))) \lor Loves(g(x),x)$$

$$Animal(f(x)) \lor Loves(g(x),x)) \land (\neg Loves(x,f(x)) \lor Loves(g(x),x)$$

$$Animal(f(x)) , Loves(g(x),x) \} , \{ \neg Loves(x,f(x)) , Loves(g(x),x)$$

Informally, the skolem function $g(x)$ can be thought of as yielding the person by whom $x$ is loved, while $f(x)$ yields the animal (if any) that $x$ doesn't love. The 3rd last line from below then reads as *" $x$ doesn't love the animal $f(x)$, or else $x$ is loved by $g(x)$ "*.
The 2nd last line from above, $(Animal(f(x)) \vee Loves(g(x), x)) \wedge (\neg Loves(x, f(x)) \vee Loves(g(x), x))$, is the CNF.

## Notes

[1] Tseitin (1968)
[2] Jackson and Sheridan (2004)
[3] since one way to check a CNF for satisfiability is to convert it into a DNF, the satisfiability of which can be checked in linear time
[4] Artificial Intelligence: A modern Approach [1995...] Russel and Norvig

## References

- Paul Jackson, Daniel Sheridan: Clause Form Conversions for Boolean Circuits. In: Holger H. Hoos, David G. Mitchell (Eds.): Theory and Applications of Satisfiability Testing, 7th International Conference, SAT 2004, Vancouver, BC, Canada, May 10–13, 2004, Revised Selected Papers. Lecture Notes in Computer Science 3542, Springer 2005, pp. 183–198
- G.S. Tseitin: On the complexity of derivation in propositional calculus. In: Slisenko, A.O. (ed.) Structures in Constructive Mathematics and Mathematical Logic, Part II, Seminars in Mathematics (translated from Russian), pp. 115–125. Steklov Mathematical Institute (1968)

## External links

- Hazewinkel, Michiel, ed. (2001), "Conjunctive normal form" (http://www.encyclopediaofmath.org/index.php?title=p/c025090), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Java applet for converting to CNF and DNF, showing laws used (http://www.izyt.com/BooleanLogic/applet.php)
- 3D application showing the difference between CNF and DNF for different De Morgan Triples (http://fuzziness.org/fuzzynorms)

# Disjunctive normal form

In boolean logic, a **disjunctive normal form** (DNF) is a standardization (or normalization) of a logical formula which is a disjunction of conjunctive clauses; otherwise put, it is an OR of ANDs also known as a Sum of products. As a normal form, it is useful in automated theorem proving. A logical formula is considered to be in DNF if and only if it is a disjunction of one or more conjunctions of one or more literals. A DNF formula is in **full disjunctive normal form** if each of its variables appears exactly once in every clause. As in conjunctive normal form (CNF), the only propositional operators in DNF are and, or, and not. The *not* operator can only be used as part of a literal, which means that it can only precede a propositional variable. For example, all of the following formulas are in DNF:

$$A \wedge B$$

$$A$$

$$(A \wedge B) \vee C$$

$$(A \wedge \neg B \wedge \neg C) \vee (\neg D \wedge E \wedge F)$$

However, the following formulas are **NOT** in DNF:

$\neg(A \vee B)$ — NOT is the outermost operator

$A \vee (B \wedge (C \vee D))$ — an OR is nested within an AND

Converting a formula to DNF involves using logical equivalences, such as the double negative elimination, De Morgan's laws, and the distributive law.

All logical formulas can be converted into disjunctive normal form. However, in some cases conversion to DNF can lead to an exponential explosion of the formula. For example, in DNF, logical formulas of the following form have $2^n$ terms:

$$(X_1 \vee Y_1) \wedge (X_2 \vee Y_2) \wedge \ldots \wedge (X_n \vee Y_n)$$

Any particular Boolean function can be represented by one and only one full disjunctive normal form, one of the two canonical forms.

The following is a formal grammar for DNF:

1. *disjunct* → *conjunct*
2. *disjunct* → *disjunct* ∨ *conjunct*
3. *conjunct* → *literal*
4. *conjunct* → (*conjunct* ∧ *literal*)
5. *literal* → *variable*
6. *literal* → ¬*variable*

Where *variable* is any variable.

## External links

- Hazewinkel, Michiel, ed. (2001), "Disjunctive normal form" [1], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Java applet for converting boolean logic expressions to CNF and DNF, showing the laws used [2]

## References

[1] http://www.encyclopediaofmath.org/index.php?title=p/d033300
[2] http://www.izyt.com/BooleanLogic/applet.php

# Truth table

A **truth table** is a mathematical table used in logic—specifically in connection with Boolean algebra, boolean functions, and propositional calculus—to compute the functional values of logical expressions on each of their functional arguments, that is, on each combination of values taken by their logical variables (Enderton, 2001). In particular, truth tables can be used to tell whether a propositional expression is true for all legitimate input values, that is, logically valid.

Practically, a truth table is composed of one column for each input variable (for example, A and B), and one final column for all of the possible results of the logical operation that the table is meant to represent (for example, A XOR B). Each row of the truth table therefore contains one possible configuration of the input variables (for instance, A=true B=false), and the result of the operation for those values. See the examples below for further clarification. Ludwig Wittgenstein is often credited with their invention in the *Tractatus Logico-Philosophicus*.

## Unary operations

### 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*** |
|---|---|
| *Operand* | *Value* |
| 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**, **Np**, **Fpq**, 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 two binary variables (P and Q are thus boolean variables; for details about the operators see below):

| *P* | *Q* | **F**$^{0}$ | **NOR**$^{1}$ | **Xq**$^{2}$ | **¬p**$^{3}$ | **↛**$^{4}$ | **¬q**$^{5}$ | **XOR**$^{6}$ | **NAND**$^{7}$ | **AND**$^{8}$ | **XNOR**$^{9}$ | **q**$^{10}$ | **if/then**$^{11}$ | **p**$^{12}$ | **then/if**$^{13}$ | **OR**$^{14}$ | **T**$^{15}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |
| **Com** | | ▯ | ▯ | | | | | ▯ | ▯ | ▯ | ▯ | | | | | ▯ | ▯ |
| **L id** | | | | F | | | | F | | T | T | TF | T | | | F | |
| **R id** | | | | | | F | | F | | T | T | | | TF | T | F | |

where T = true and F = false. The **Com** row indicates whether an operator, **op**, is commutative - **P op Q = Q op P**. The **L id** row shows the operator's left identity if it has one - a value **I** such that **I op Q = Q**. The **R id** row shows the operator's right identity if it has one - a value **I** such that **P op I = P**.

Key:

| | | | | **Operation name** |
|---|---|---|---|---|
| 0 | O*pq* | F | false | Contradiction |
| 1 | X*pq* | NOR | ↓ | Logical NOR |
| 2 | M*pq* | Xq | | Converse nonimplication |
| 3 | F*pq* | N*p* | **¬p** | Negation |
| 4 | L*pq* | Xp | ▯ | Material nonimplication |
| 5 | G*pq* | N*q* | **¬q** | Negation |
| 6 | J*pq* | XOR | ⊕ | Exclusive disjunction |
| 7 | D*pq* | NAND | ↑ | Logical NAND |
| 8 | K*pq* | AND | ∧ | Logical conjunction |
| 9 | E*pq* | XNOR | If and only if | Logical biconditional |
| 10 | H*pq* | **q** | | Projection function |

| | | | | |
|---|---|---|---|---|
| 11 | C*pq* | XNp | if/then | Material implication |
| 12 | I*pq* | **p** | | Projection function |
| 13 | B*pq* | XNq | then/if | Converse implication |
| 14 | A*pq* | OR | ∨ | Logical disjunction |
| 15 | V*pq* | T | 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**, **Kpq**, **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 $p \wedge q$ is true. For all other assignments of logical values to *p* and to *q* the conjunction $p \wedge q$ is false.

It can also be said that if *p*, then $p \wedge q$ is *q*, otherwise $p \wedge q$ 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**, **Apq**, **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 |

Stated in English, if *p*, then $p \vee q$ is *p*, otherwise $p \vee q$ is *q*.

## Logical implication

Logical implication or 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**, or **Cpq**) is as follows:

### Logical Implication

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

It may also be useful to note that **p → q** is equivalent to **¬p ▯ q**.

## 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**, **Epq**, **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 |

So p EQ q is true if p and q have the same truth value (both true or both false), and false if they have different truth values.

## 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**, **Jpq**, or **p ≠ q**) is as follows:

**Exclusive Disjunction**

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

For two propositions, **XOR** can also be written as (p = 1 ∧ q = 0) ∨ (p = 0 ∧ q = 1).

## 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**, **Dpq**, 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: ¬(*p* ∧ *q*), and the disjunction of negations: (¬*p*) ∨ (¬*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**, **Xpq**, 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* ∨ *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 |

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 ¬(*p* ∧ *q*) as for (¬*p*) ∨ (¬*q*), and for ¬(*p* ∨ *q*) as for (¬*p*) ∧ (¬*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.

## Applications

Truth tables can be used to prove many other logical equivalences. For example, consider the following truth table:

**Logical Equivalence : (*p* → *q*) = (¬*p* ∨ *q*)**

| *p* | *q* | ¬*p* | ¬*p* ∨ *q* | *p* → *q* |
|---|---|---|---|---|
| T | T | F | T | T |
| T | F | F | F | F |
| F | T | T | T | T |
| F | F | T | T | T |

This demonstrates the fact that *p* → *q* is logically equivalent to ¬*p* ∨ *q*.

## Truth table for most commonly used logical operators

Here is a truth table giving definitions of the most commonly used 6 of the 16 possible truth functions of 2 binary variables (P,Q are thus boolean variables):

| $P$ | $Q$ | $P \wedge Q$ | $P \vee Q$ | $P \underline{\vee} Q$ | $P \underline{\wedge} Q$ | $P \Rightarrow Q$ | $P \Leftarrow Q$ | $P \iff Q$ |
|---|---|---|---|---|---|---|---|---|
| T | T | T | T | F | T | T | T | T |
| T | F | F | T | T | F | F | T | F |
| F | T | F | T | T | F | T | F | F |
| F | F | F | F | F | T | T | T | T |

Key:

T = true, F = false

$\wedge$ = AND (logical conjunction)

$\vee$ = OR (logical disjunction)

$\underline{\vee}$ = XOR (exclusive or)

$\underline{\wedge}$ = XNOR (exclusive nor)

$\rightarrow$ = conditional "if-then"

$\leftarrow$ = conditional "(then)-if"

$\iff$ biconditional or "if-and-only-if" is logically equivalent to $\underline{\wedge}$ : XNOR (exclusive nor).

Logical operators can also be visualized using Venn diagrams.

## Condensed truth tables for binary operators

For binary operators, a condensed form of truth table is also used, where the row headings and the column headings specify the operands and the table cells specify the result. For example Boolean logic uses this condensed truth table notation:

| ▯ | **F** | **T** |
|---|---|---|
| **F** | F | F |
| **T** | F | T |

| ▯ | **F** | **T** |
|---|---|---|
| **F** | F | T |
| **T** | T | T |

This notation is useful especially if the operations are commutative, although one can additionally specify that the rows are the first operand and the columns are the second operand. This condensed notation is particularly useful in discussing multi-valued extensions of logic, as it significantly cuts down on combinatoric explosion of the number of rows otherwise needed. It also provides for quickly recognizable characteristic "shape" of the distribution of the values in the table which can assist the reader in grasping the rules more quickly.

## Truth tables in digital logic

Truth tables are also used to specify the functionality of hardware look-up tables (LUTs) in digital logic circuitry. For an n-input LUT, the truth table will have 2^$n$ values (or rows in the above tabular format), completely specifying a boolean function for the LUT. By representing each boolean value as a bit in a binary number, truth table values can be efficiently encoded as integer values in electronic design automation (EDA) software. For example, a 32-bit integer can encode the truth table for a LUT with up to 5 inputs.

When using an integer representation of a truth table, the output value of the LUT can be obtained by calculating a bit index *k* based on the input values of the LUT, in which case the LUT's output value is the *k*th bit of the integer. For example, to evaluate the output value of a LUT given an array of *n* boolean input values, the bit index of the truth table's output value can be computed as follows: if the *i*th input is true, let V*i* = 1, else let V*i* = 0. Then the *k*th bit of the binary representation of the truth table is the LUT's output value, where *k* = V0*2^0 + V1*2^1 + V2*2^2 + ... + V*n**2^*n*.

Truth tables are a simple and straightforward way to encode boolean functions, however given the exponential growth in size as the number of inputs increase, they are not suitable for functions with a large number of inputs. Other representations which are more memory efficient are text equations and binary decision diagrams.

## Applications of truth tables in digital electronics

In digital electronics and computer science (fields of applied logic engineering and mathematics), truth tables can be used to reduce basic boolean operations to simple correlations of inputs to outputs, without the use of logic gates or code. For example, a binary addition can be represented with the truth table:

```
A B | C R
1 1 | 1 0
1 0 | 0 1
0 1 | 0 1
0 0 | 0 0

where

A = First Operand
B = Second Operand
C = Carry
R = Result
```

This truth table is read left to right:

- Value pair (A,B) equals value pair (C,R).
- Or for this example, A plus B equal result R, with the Carry C.

Note that this table does not describe the logic operations necessary to implement this operation, rather it simply specifies the function of inputs to output values.

With respect to the result, this example may be arithmetically viewed as modulo 2 binary addition, and as logically equivalent to the exclusive-or (exclusive disjunction) binary logic operation.

In this case it can be used for only very simple inputs and outputs, such as 1s and 0s. However, if the number of types of values one can have on the inputs increases, the size of the truth table will increase.

For instance, in an addition operation, one needs two operands, A and B. Each can have one of two values, zero or one. The number of combinations of these two values is 2×2, or four. So the result is four possible outputs of C and R. If one were to use base 3, the size would increase to 3×3, or nine possible outputs.

The first "addition" example above is called a half-adder. A full-adder is when the carry from the previous operation is provided as input to the next adder. Thus, a truth table of eight rows would be needed to describe a full adder's logic:

```
A B C* | C R
0 0 0  | 0 0
0 1 0  | 0 1
```

```
1 0 0  | 0 1
1 1 0  | 1 0
0 0 1  | 0 1
0 1 1  | 1 0
1 0 1  | 1 0
1 1 1  | 1 1

Same as previous, but..
C* = Carry from previous adder
```

## History

Irving Anellis has done the research to show that C.S. Peirce appears to be the earliest logician (in 1893) to devise a truth table matrix. From the summary of his paper:

> In 1997, John Shosky discovered, on the verso of a page of the typed transcript of Bertrand Russell's 1912 lecture on "The Philosophy of Logical Atomism" truth table matrices. The matrix for negation is Russell's, alongside of which is the matrix for material implication in the hand of Ludwig Wittgenstein. It is shown that an unpublished manuscript identified as composed by Peirce in 1893 includes a truth table matrix that is equivalent to the matrix for material implication discovered by John Shosky. An unpublished manuscript by Peirce identified as having been composed in 1883-84 in connection with the composition of Peirce's "On the Algebra of Logic: A Contribution to the Philosophy of Notation" that appeared in the American Journal of Mathematics in 1885 includes an example of an indirect truth table for the conditional.

## References

## Further reading

- Bocheński, Józef Maria (1959), *A Précis of Mathematical Logic*, translated from the French and German editions by Otto Bird, Dordrecht, South Holland: D. Reidel.
- Enderton, H. (2001). *A Mathematical Introduction to Logic*, second edition, New York: Harcourt Academic Press. ISBN 0-12-238452-0
- Quine, W.V. (1982), *Methods of Logic*, 4th edition, Cambridge, MA: Harvard University Press.

## External links

- Hazewinkel, Michiel, ed. (2001), "Truth table" (http://www.encyclopediaofmath.org/index.php?title=p/t094370), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Truth Tables, Tautologies, and Logical Equivalence (http://www.millersville.edu/~bikenaga/math-proof/truth-tables/truth-tables.html)
- PEIRCE'S TRUTH-FUNCTIONAL ANALYSIS AND THE ORIGIN OF TRUTH TABLES by Irving H. Anellis (http://arxiv.org/ftp/arxiv/papers/1108/1108.2429.pdf)

# De Morgan's laws

| **Transformation rules** |
|---|
| **Propositional calculus** |
| Rules of inference |
| • *Modus ponens* |
| • *Modus tollens* |
| • Biconditional introduction |
| • Biconditional elimination |
| • Conjunction introduction |
| • Simplification |
| • Disjunction introduction |
| • Disjunction elimination |
| • Disjunctive syllogism |
| • Hypothetical syllogism |
| • Constructive dilemma |
| • Destructive dilemma |
| • Absorption |
| • Negation Introduction |
| Rules of replacement |
| • Associativity |
| • Commutativity |
| • Distributivity |
| • Double negation |
| • De Morgan's laws |
| • Transposition |
| • Material implication |
| • Material equivalence |
| • Exportation |
| • Tautology |
| **Predicate logic** |
| • Universal generalization |
| • Universal instantiation |
| • Existential generalization |
| • Existential instantiation |
| • v |
| • t |
| • e [1] |

In propositional logic and boolean algebra, **De Morgan's laws**[2][3][4] are a pair of transformation rules that are both valid rules of inference. The rules allow the expression of conjunctions and disjunctions purely in terms of each other via negation.

The rules can be expressed in English as:

> The negation of a conjunction is the disjunction of the negations.
> The negation of a disjunction is the conjunction of the negations.

or informally as:

> "***not (A and B)***" is the same as "***(not A) or (not B)***"
> and also,

"***not (A or B)***" is the same as "***(not A) and (not B)***"

The rules can be expressed in formal language with two propositions *P* and *Q* as:

$$\neg(P \land Q) \iff (\neg P) \lor (\neg Q)$$
$$\neg(P \lor Q) \iff (\neg P) \land (\neg Q)$$

where:

- $\neg$ is the negation operator (NOT)
- $\land$ is the conjunction operator (AND)
- $\lor$ is the disjunction operator (OR)
- $\Leftrightarrow$ is a metalogical symbol meaning "can be replaced in a logical proof with"

Applications of the rules include simplification of logical expressions in computer programs and digital circuit designs. De Morgan's laws are an example of a more general concept of mathematical duality.

## Formal notation

The *negation of conjunction* rule may be written in sequent notation:

$$\neg(P \land Q) \vdash (\neg P \lor \neg Q)$$

The *negation of disjunction* rule may be written as:

$$\neg(P \lor Q) \vdash (\neg P \land \neg Q)$$

In rule form: *negation of conjunction*

$$\frac{\neg(P \land Q)}{\therefore \neg P \lor \neg Q}$$

and *negation of disjunction*

$$\frac{\neg(P \lor Q)}{\therefore \neg P \land \neg Q}$$

and expressed as a truth-functional tautology or theorem of propositional logic:

$$\neg(P \land Q) \to (\neg P \lor \neg Q)$$
$$\neg(P \lor Q) \to (\neg P \land \neg Q)$$

where $P$, and $Q$ are propositions expressed in some formal system.

### Substitution form

De Morgan's laws are normally shown in the compact form above, with negation of the output on the left and negation of the inputs on the right. A clearer form for substitution can be stated as:

$$(P \land Q) \equiv \neg(\neg P \lor \neg Q)$$
$$(P \lor Q) \equiv \neg(\neg P \land \neg Q)$$

This emphasizes the need to invert both the inputs and the output, as well as change the operator, when doing a substitution.

## Set theory and Boolean algebra

In set theory and Boolean algebra, it is often stated as "Union and intersection interchange under complementation",[5] which can be formally expressed as:

- $\overline{A \cup B} \equiv \overline{A} \cap \overline{B}$
- $\overline{A \cap B} \equiv \overline{A} \cup \overline{B}$

where:

- $\overline{A}$ is the negation of A, the overline being written above the terms to be negated
- $\cap$ is the intersection operator (AND)
- $\cup$ is the union operator (OR)

The generalized form is:

$$\overline{\bigcap_{i \in I} A_i} \equiv \bigcup_{i \in I} \overline{A_i}$$
$$\overline{\bigcup_{i \in I} A_i} \equiv \bigcap_{i \in I} \overline{A_i}$$

where *I* is some, possibly uncountable, indexing set.

In set notation, De Morgan's law can be remembered using the mnemonic "break the line, change the sign".[6]

## Engineering

In electrical and computer engineering, De Morgan's law is commonly written as:

$$\overline{A \cdot B} \equiv \overline{A} + \overline{B}$$
$$\overline{A + B} \equiv \overline{A} \cdot \overline{B}$$

where:

- $\cdot$ is a logical AND
- $+$ is a logical OR
- the overbar is the logical NOT of what is underneath the overbar.

## Text Searching

De Morgan’s laws commonly apply to text searching using Boolean operators AND, OR, and NOT. Consider a set of documents containing the words “cars” or “trucks”. De Morgan’s laws hold that these two searches will return the same set of documents:

Search A: NOT (cars OR trucks)

Search B: (NOT cars) AND (NOT trucks)

The corpus of documents containing “cars” or “trucks” can be represented by four documents:

Document 1: Contains only the word “cars”.

Document 2: Contains only “trucks”.

Document 3: Contains both “cars” and “trucks”.

Document 4: Contains neither “cars” nor “trucks”.

To evaluate Search A, clearly the search “(cars OR trucks)” will hit on Documents 1, 2, and 3. So the negation of that search (which is Search A) will hit everything else, which is Document 4.

Evaluating Search B, the search “(NOT cars)” will hit on documents that do not contain “cars”, which is Documents 2 and 4. Similarly the search “(NOT trucks)” will hit on Documents 1 and 4. Applying the AND operator to these two searches (which is Search B) will hit on the documents that are common to these two searches, which is Document 4.

A similar evaluation can be applied to show that the following two searches will return the same set of documents (Documents 1, 2, 4):

Search C: NOT (cars AND trucks)

Search D: (NOT cars) OR (NOT trucks)

## History

The law is named after Augustus De Morgan (1806–1871)[7] who introduced a formal version of the laws to classical propositional logic. De Morgan's formulation was influenced by algebraization of logic undertaken by George Boole, which later cemented De Morgan's claim to the find. Although a similar observation was made by Aristotle and was known to Greek and Medieval logicians[8] (in the 14th century, William of Ockham wrote down the words that would result by reading the laws out),[9] De Morgan is given credit for stating the laws formally and incorporating them into the language of logic. De Morgan's Laws can be proved easily, and may even seem trivial.[10] Nonetheless, these laws are helpful in making valid inferences in proofs and deductive arguments.

## Informal proof

De Morgan's theorem may be applied to the negation of a disjunction or the negation of a conjunction in all or part of a formula.

### Negation of a disjunction

In the case of its application to a disjunction, consider the following claim: "it is false that either of A or B is true", which is written as:

$$\neg(A \lor B)$$

In that it has been established that *neither* A nor B is true, then it must follow that both A is not true and B is not true, which may be written directly as:

$$(\neg A) \land (\neg B)$$

If either A or B *were* true, then the disjunction of A and B would be true, making its negation false. Presented in English, this follows the logic that "Since two things are both false, it is also false that either of them is true."

Working in the opposite direction, the second expression asserts that A is false and B is false (or equivalently that "not A" and "not B" are true). Knowing this, a disjunction of A and B must be false also. The negation of said disjunction must thus be true, and the result is identical to the first claim.

### Negation of a conjunction

The application of De Morgan's theorem to a conjunction is very similar to its application to a disjunction both in form and rationale. Consider the following claim: "it is false that A and B are both true", which is written as:

$$\neg(A \land B)$$

In order for this claim to be true, either or both of A or B must be false, for if they both were true, then the conjunction of A and B would be true, making its negation false. Thus, one (at least) or more of A and B must be false (or equivalently, one or more of "not A" and "not B" must be true). This may be written directly as:

$$(\neg A) \lor (\neg B)$$

Presented in English, this follows the logic that "Since it is false that two things are both true, at least one of them must be false."

Working in the opposite direction again, the second expression asserts that at least one of "not A" and "not B" must be true, or equivalently that at least one of A and B must be false. Since at least one of them must be false, then their conjunction would likewise be false. Negating said conjunction thus results in a true expression, and this expression is identical to the first claim.

## Formal proof

The proof that $(A \cap B)^c = A^c \cup B^c$ is done by first proving that $(A \cap B)^c \subseteq A^c \cup B^c$, and then by proving that $A^c \cup B^c \subseteq (A \cap B)^c$

Let $x \in (A \cap B)^c$. Then $x \not\in A \cap B$. Because $A \cap B = \{y | y \in A \text{ and } y \in B\}$, then either $x \not\in A$ or $x \not\in B$. If $x \not\in A$, then $x \in A^c$, so then $x \in A^c \cup B^c$. Otherwise, if $x \not\in B$, then $x \in B^c$, so $x \in A^c \cup B^c$. Because this is true for any arbitrary $x \in (A \cap B)^c$, then $\forall x \in (A \cap B)^c, x \in A^c \cup B^c$, and so $(A \cap B)^c \subseteq A^c \cup B^c$.

To prove the reverse direction, assume that $\exists x \in A^c \cup B^c$ such that $x \not\in (A \cap B)^c$. Then $x \in A \cap B$. It follows that $x \in A$ and $x \in B$. Then $x \not\in A^c$ and $x \not\in B^c$. But then $x \not\in A^c \cup B^c$, in contradiction to the hypothesis that $x \in A^c \cup B^c$. Therefore, $\forall x \in A^c \cup B^c, x \in (A \cap B)^c$, and $A^c \cup B^c \subseteq (A \cap B)^c$.

Because $A^c \cup B^c \subseteq (A \cap B)^c$ and $(A \cap B)^c \subseteq A^c \cup B^c$, then $(A \cap B)^c = A^c \cup B^c$, concluding the proof of De Morgan's Law.

The other De Morgan's Law, that $(A \cup B)^c = A^c \cap B^c$, is proven similarly.

## Extensions

In extensions of classical propositional logic, the duality still holds (that is, to any logical operator we can always find its dual), since in the presence of the identities governing negation, one may always introduce an operator that is the De Morgan dual of another. This leads to an important property of logics based on classical logic, namely the existence of negation normal forms: any formula is equivalent to another formula where negations only occur applied to the non-logical atoms of the formula. The existence of negation normal forms drives many applications, for example in digital circuit design, where it is used to manipulate the types of logic gates, and in formal logic, where it is a prerequisite for finding the conjunctive normal form and disjunctive normal form of a formula. Computer programmers use them to simplify or properly negate complicated logical conditions. They are also often useful in computations in elementary probability theory.

Let us define the dual of any propositional operator P(*p*, *q*, ...) depending on elementary propositions *p*, *q*, ... to be the operator $\mathrm{P}^d$ defined by

$$\mathrm{P}^d(p, q, ...) = \neg P(\neg p, \neg q, \dots).$$

This idea can be generalised to quantifiers, so for example the universal quantifier and existential quantifier are duals:

$$\forall x \, P(x) \equiv \neg \exists x \, \neg P(x),$$
$$\exists x \, P(x) \equiv \neg \forall x \, \neg P(x).$$

To relate these quantifier dualities to the De Morgan laws, set up a model with some small number of elements in its domain *D*, such as

$D = \{a, b, c\}$.

Then

$$\forall x \, P(x) \equiv P(a) \wedge P(b) \wedge P(c)$$

and

$$\exists x \, P(x) \equiv P(a) \vee P(b) \vee P(c).$$

But, using De Morgan's laws,

$$P(a) \wedge P(b) \wedge P(c) \equiv \neg(\neg P(a) \vee \neg P(b) \vee \neg P(c))$$

and

$$P(a) \lor P(b) \lor P(c) \equiv \neg(\neg P(a) \land \neg P(b) \land \neg P(c)),$$

verifying the quantifier dualities in the model.

Then, the quantifier dualities can be extended further to modal logic, relating the box ("necessarily") and diamond ("possibly") operators:

$$\Box p \equiv \neg \Diamond \neg p,$$

$$\Diamond p \equiv \neg \Box \neg p.$$

In its application to the alethic modalities of possibility and necessity, Aristotle observed this case, and in the case of normal modal logic, the relationship of these modal operators to the quantification can be understood by setting up models using Kripke semantics.

## References

[1] http://en.wikipedia.org/w/index.php?title=Template:Transformation_rules&action=edit

[2] Copi and Cohen

[3] Hurley

[4] Moore and Parker

[5] *Boolean Algebra* By R. L. Goodstein. ISBN 0-486-45894-6

[6] 2000 Solved Problems in Digital Electronics (http://books.google.com/books?id=NdAjEDP5mDsC&pg=PA81&lpg=PA81&dq=break+the+line+change+the+sign&source=web&ots=BtUl4oQOja&sig=H1Wz9e6Uv_bNeSbTvN6lr3s47PQ#PPA81,M1) By S. P. Bali

[7] *DeMorgan's Theorems* (http://www.mtsu.edu/~phys2020/Lectures/L19-L25/L3/DeMorgan/body_demorgan.html) at mtsu.edu

[8] Bocheński's *History of Formal Logic*

[9] William of Ockham, Summa Logicae, part II, sections 32 & 33.

[10] Augustus De Morgan (1806 -1871) (http://www.engr.iupui.edu/~orr/webpages/cpt120/mathbios/ademo.htm) by Robert H. Orr

## External links

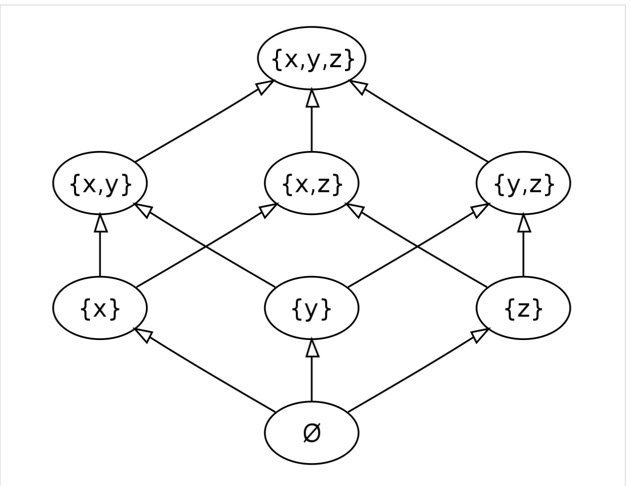
- Hazewinkel, Michiel, ed. (2001), "Duality principle" (http://www.encyclopediaofmath.org/index.php?title=p/d034130), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Weisstein, Eric W., " de Morgan's Laws (http://mathworld.wolfram.com/deMorgansLaws.html)", *MathWorld*.
- de Morgan's laws (http://planetmath.org/encyclopedia/DeMorgansLaws.html) at PlanetMath

# Boolean algebra (structure)

In abstract algebra, a **Boolean algebra** or **Boolean lattice** is a complemented distributive lattice. This type of algebraic structure captures essential properties of both set operations and logic operations. A Boolean algebra can be seen as a generalization of a power set algebra or a field of sets, or its elements can be viewed as generalized truth values. It is also a special case of a De Morgan algebra and a Kleene algebra.

A Boolean ring is essentially the same thing as a Boolean algebra, with ring multiplication corresponding to conjunction or meet ∧, and ring addition to exclusive disjunction or symmetric difference (not disjunction ∨).

## History



Boolean lattice of subsets

The term "Boolean algebra" honors George Boole (1815–1864), a self-educated English mathematician. He introduced the algebraic system initially in a small pamphlet, *The Mathematical Analysis of Logic*, published in 1847 in response to an ongoing public controversy between Augustus De Morgan and William Hamilton, and later as a more substantial book, *The Laws of Thought*, published in 1854. Boole's formulation differs from that described above in some important respects. For example, conjunction and disjunction in Boole were not a dual pair of operations. Boolean algebra emerged in the 1860s, in papers written by William Jevons and Charles Sanders Peirce. The first systematic presentation of Boolean algebra and distributive lattices is owed to the 1890 *Vorlesungen* of Ernst Schröder. The first extensive treatment of Boolean algebra in English is A. N. Whitehead's 1898 *Universal Algebra*. Boolean algebra as an axiomatic algebraic structure in the modern axiomatic sense begins with a 1904 paper by Edward V. Huntington. Boolean algebra came of age as serious mathematics with the work of Marshall Stone in the 1930s, and with Garrett Birkhoff's 1940 *Lattice Theory*. In the 1960s, Paul Cohen, Dana Scott, and others found deep new results in mathematical logic and axiomatic set theory using offshoots of Boolean algebra, namely forcing and Boolean-valued models.

## Definition

A **Boolean algebra** is a six-tuple consisting of a set *A*, equipped with two binary operations ∧ (called "meet" or "and"), ∨ (called "join" or "or"), a unary operation ¬ (called "complement" or "not") and two elements 0 and 1 (called "bottom" and "top", or "least" and "greatest" element, also denoted by the symbols ⊥ and ⊤, respectively), such that for all elements *a*, *b* and *c* of *A*, the following axioms hold:[1]

| | | |
|---|---|---|
| $a \vee (b \vee c) = (a \vee b) \vee c$ | $a \wedge (b \wedge c) = (a \wedge b) \wedge c$ | associativity |
| $a \vee b = b \vee a$ | $a \wedge b = b \wedge a$ | commutativity |
| $a \vee (a \wedge b) = a$ | $a \wedge (a \vee b) = a$ | absorption |
| $a \vee 0 = a$ | $a \wedge 1 = a$ | identity |
| $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$ | $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ | distributivity |
| $a \vee \neg a = 1$ | $a \wedge \neg a = 0$ | complements |

Note, however, that the absorption law can be excluded from the set of axioms as it can be derived by the other axioms.

A Boolean algebra with only one element is called a **trivial Boolean algebra** or a **degenerate Boolean algebra**. (Some authors require 0 and 1 to be *distinct* elements in order to exclude this case.)

It follows from the last three pairs of axioms above (identity, distributivity and complements), or from the absorption axiom, that

$$a = b \wedge a \quad \text{if and only if} \quad a \vee b = b.$$

The relation ≤ defined by $a \leq b$ if these equivalent conditions hold, is a partial order with least element 0 and greatest element 1. The meet $a \wedge b$ and the join $a \vee b$ of two elements coincide with their infimum and supremum, respectively, with respect to ≤.

The first four pairs of axioms constitute a definition of a bounded lattice.

It follows from the first five pairs of axioms that any complement is unique.

The set of axioms is self-dual in the sense that if one exchanges ∨ with ∧ and 0 with 1 in an axiom, the result is again an axiom. Therefore by applying this operation to a Boolean algebra (or Boolean lattice), one obtains another Boolean algebra with the same elements; it is called its **dual**.

## Examples

- The simplest non-trivial Boolean algebra, the two-element Boolean algebra, has only two elements, 0 and 1, and is defined by the rules:

| ▯ | **0** | **1** |
|---|---|---|
| **0** | 0 | 0 |
| **1** | 0 | 1 |

| ▯ | **0** | **1** |
|---|---|---|
| **0** | 0 | 1 |
| **1** | 1 | 1 |

| ***a*** | **0** | **1** |
|---|---|---|
| **¬*a*** | 1 | 0 |

  - It has applications in logic, interpreting 0 as *false*, 1 as *true*, ∧ as *and*, ∨ as *or*, and ¬ as *not*. Expressions involving variables and the Boolean operations represent statement forms, and two such expressions can be shown to be equal using the above axioms if and only if the corresponding statement forms are logically equivalent.
  - The two-element Boolean algebra is also used for circuit design in electrical engineering; here 0 and 1 represent the two different states of one bit in a digital circuit, typically high and low voltage. Circuits are described by expressions containing variables, and two such expressions are equal for all values of the variables if and only if the corresponding circuits have the same input-output behavior. Furthermore, every possible input-output behavior can be modeled by a suitable Boolean expression.
  - The two-element Boolean algebra is also important in the general theory of Boolean algebras, because an equation involving several variables is generally true in all Boolean algebras if and only if it is true in the

two-element Boolean algebra (which can be checked by a trivial brute force algorithm for small numbers of variables). This can for example be used to show that the following laws (*Consensus theorems*) are generally valid in all Boolean algebras:

- $(a \vee b) \wedge (\neg a \vee c) \wedge (b \vee c) \equiv (a \vee b) \wedge (\neg a \vee c)$
- $(a \wedge b) \vee (\neg a \wedge c) \vee (b \wedge c) \equiv (a \wedge b) \vee (\neg a \wedge c)$

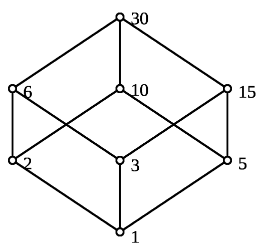
- The power set (set of all subsets) of any given nonempty set *S* forms a Boolean algebra, an algebra of sets, with the two operations ∨ := ∪ (union) and ∧ := ∩ (intersection). The smallest element 0 is the empty set and the largest element 1 is the set *S* itself.
  - After the two-element Boolean algebra, the simplest Boolean algebra is that defined by the power set of two atoms:

| ▯ | **0** | **a** | **b** | **1** |
|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 0 |
| **a** | 0 | a | 0 | a |
| **b** | 0 | 0 | b | b |
| **1** | 0 | a | b | 1 |

| ▯ | **0** | **a** | **b** | **1** |
|---|---|---|---|---|
| **0** | 0 | a | b | 1 |
| **a** | a | a | 1 | 1 |
| **b** | b | 1 | b | 1 |
| **1** | 1 | 1 | 1 | 1 |

| ***x*** | **0** | **a** | **b** | **1** |
|---|---|---|---|---|
| **¬*x*** | 1 | b | a | 0 |

- The set of all subsets of *S* that are either finite or cofinite is a Boolean algebra, an algebra of sets.
- Starting with the propositional calculus with κ sentence symbols, form the Lindenbaum algebra (that is, the set of sentences in the propositional calculus modulo tautology). This construction yields a Boolean algebra. It is in fact the free Boolean algebra on κ generators. A truth assignment in propositional calculus is then a Boolean algebra homomorphism from this algebra to the two-element Boolean algebra.
- Given any linearly ordered set *L* with a least element, the interval algebra is the smallest algebra of subsets of *L* containing all of the half-open intervals [*a*, *b*) such that *a* is in *L* and *b* is either in *L* or equal to ∞. Interval algebras are useful in the study of Lindenbaum-Tarski algebras; every countable Boolean algebra is isomorphic to an interval algebra.
- For any natural number *n*, the set of all positive divisors of *n*, defining *a*≤*b* if *a* divides *b*, forms a distributive lattice. This lattice is a Boolean algebra if and only if *n* is square-free. The bottom and the top element of this Boolean algebra is the natural number 1 and *n*, respectively. The complement of *a* is given by *n*/*a*. The meet and the join of *a* and *b* is given by the greatest common divisor (gcd) and the least common multiple (lcm) of *a* and *b*, respectively. The ring addition *a*+*b* is given by lcm(*a*,*b*)/gcd(*a*,*b*). The picture shows an example for *n* = 30. As a counter-example, considering the non-square-free *n*=60, the greatest common divisor of 30 and its complement 2 would be 2, while it should be the bottom element 1.



Hasse diagram of the Boolean algebra of divisors of 30.

- Other examples of Boolean algebras arise from topological spaces: if *X* is a topological space, then the collection of all subsets of *X* which are both open and closed forms a Boolean algebra with the operations ∨ := ∪ (union) and ∧ := ∩ (intersection).
- If *R* is an arbitrary ring and we define the set of *central idempotents* by
  $A = \{\ e \in R : e^2 = e,\ ex = xe,\ \forall x \in R\ \}$
  then the set *A* becomes a Boolean algebra with the operations $e \vee f := e + f - ef$ and $e \wedge f := ef$.

## Homomorphisms and isomorphisms

A *homomorphism* between two Boolean algebras $A$ and $B$ is a function $f : A \rightarrow B$ such that for all $a$, $b$ in $A$:

$f(a \vee b) = f(a) \vee f(b)$,

$f(a \wedge b) = f(a) \wedge f(b)$,

$f(1) = 1$.

It then follows that $f(\neg a) = \neg f(a)$ for all $a$ in $A$ and that $f(0) = 0$ as well. The class of all Boolean algebras, together with this notion of morphism, forms a full subcategory of the category of lattices.

## Boolean rings

Every Boolean algebra (A, ∧, ∨) gives rise to a ring $(A, +, \cdot)$ by defining $a + b := (a \wedge \neg b) \vee (b \wedge \neg a) = (a \vee b) \wedge \neg(a \wedge b)$ (this operation is called symmetric difference in the case of sets and XOR in the case of logic) and $a \cdot b := a \wedge b$. The zero element of this ring coincides with the 0 of the Boolean algebra; the multiplicative identity element of the ring is the 1 of the Boolean algebra. This ring has the property that $a \cdot a = a$ for all $a$ in $A$; rings with this property are called Boolean rings.

Conversely, if a Boolean ring $A$ is given, we can turn it into a Boolean algebra by defining $x \vee y := x + y + (x \cdot y)$ and $x \wedge y := x \cdot y$. [2][3] Since these two constructions are inverses of each other, we can say that every Boolean ring arises from a Boolean algebra, and vice versa. Furthermore, a map $f : A \rightarrow B$ is a homomorphism of Boolean algebras if and only if it is a homomorphism of Boolean rings. The categories of Boolean rings and Boolean algebras are equivalent.

Hsiang (1985) gave a rule-based algorithm to check whether two arbitrary expressions denote the same value in every Boolean ring. More generally, Boudet, Jouannaud, and Schmidt-Schauß (1989) gave an algorithm to solve equations between arbitrary Boolean-ring expressions. Employing the similarity of Boolean rings and Boolean algebras, both algorithms have applications in automated theorem proving.

## Ideals and filters

An *ideal* of the Boolean algebra $A$ is a subset $I$ such that for all $x$, $y$ in $I$ we have x ∨ y in $I$ and for all $a$ in $A$ we have $a \wedge x$ in $I$. This notion of ideal coincides with the notion of ring ideal in the Boolean ring $A$. An ideal $I$ of $A$ is called *prime* if $I \neq A$ and if $a \wedge b$ in $I$ always implies $a$ in $I$ or $b$ in $I$. Furthermore, for every $a \in A$ we have that $a \wedge -a = 0 \in I$ and then $a \in I$ or $-a \in I$ for every $a \in A$, if $I$ is prime. An ideal $I$ of $A$ is called *maximal* if $I \neq A$ and if the only ideal properly containing $I$ is $A$ itself. For an ideal $I$, if $a \notin I$ and $-a \notin I$, then $I \cup \{a\}$ or $I \cup \{-a\}$ is properly contained in another ideal $J$. Hence, that an $I$ is not maximal and therefore the notions of prime ideal and maximal ideal are equivalent in Boolean algebras. Moreover, these notions coincide with ring theoretic ones of prime ideal and maximal ideal in the Boolean ring $A$.

The dual of an *ideal* is a *filter*. A *filter* of the Boolean algebra $A$ is a subset $p$ such that for all $x$, $y$ in $p$ we have $x \wedge y$ in $p$ and for all $a$ in $A$ we have $a \vee x$ in $p$. The dual of a *maximal* (or *prime*) *ideal* in a Boolean algebra is *ultrafilter*. The statement *every filter in a Boolean algebra can be extended to an ultrafilter* is called the *Ultrafilter Theorem* and can not be proved in ZF, if ZF is consistent. Within ZF, it is strictly weaker than the axiom of choice. The Ultrafilter Theorem has many equivalent formulations: *every Boolean algebra has an ultrafilter*, *every ideal in a Boolean algebra can be extended to a prime ideal*, etc.

## Representations

It can be shown that every *finite* Boolean algebra is isomorphic to the Boolean algebra of all subsets of a finite set. Therefore, the number of elements of every finite Boolean algebra is a power of two.

Stone's celebrated *representation theorem for Boolean algebras* states that *every* Boolean algebra *A* is isomorphic to the Boolean algebra of all clopen sets in some (compact totally disconnected Hausdorff) topological space.

## Axiomatics

| **Proven properties** | |
|---|---|
| **UId$_1$** **If $x \vee o = x$ for all $x$, then $o = 0$**<br>Proof: If $x \vee o = x$, then<br>$0$<br>$= 0 \vee o$ by assumption<br>$= o \vee 0$ by **Cmm$_1$**<br>$= o$ by **Idn$_1$** | **UId$_2$** [dual] If $x \wedge i = x$ for all $x$, then $i = 1$ |
| **Idm$_1$** **$x \vee x = x$**<br>Proof: $x \vee x$<br>$= (x \vee x) \wedge 1$ by **Idn$_2$**<br>$= (x \vee x) \wedge (x \vee \neg x)$ by **Cpl$_1$**<br>$= x \vee (x \wedge \neg x)$ by **Dst$_1$**<br>$= x \vee 0$ by **Cpl$_2$**<br>$= x$ by **Idn$_1$** | **Idm$_2$** [dual] $x \wedge x = x$ |
| **Bnd$_1$** **$x \vee 1 = 1$**<br>Proof: $x \vee 1$<br>$= (x \vee 1) \wedge 1$ by **Idn$_2$**<br>$= 1 \wedge (x \vee 1)$ by **Cmm$_2$**<br>$= (x \vee \neg x) \wedge (x \vee 1)$ by **Cpl$_1$**<br>$= x \vee (\neg x \wedge 1)$ by **Dst$_1$**<br>$= x \vee \neg x$ by **Idn$_2$**<br>$= 1$ by **Cpl$_1$** | **Bnd$_2$** [dual] $x \wedge 0 = 0$ |

**Abs$_1$** $\boldsymbol{x \lor (x \land y) = x}$

Proof: $x \lor (x \land y)$

= $(x \land 1) \lor (x \land y)$ by **Idn$_2$**

= $x \land (1 \lor y)$ by **Dst$_2$**

= $x \land (y \lor 1)$ by **Cmm$_1$**

= $x \land 1$ by **Bnd$_1$**

= $x$ by **Idn$_2$**

**Abs$_2$** [dual] $x \land (x \lor y) = x$

**UNg** **If $\boldsymbol{x \lor x_n = 1}$ and $\boldsymbol{x \land x_n = 0}$, then $\boldsymbol{x_n = \neg x}$**

Proof: If $x \lor x_n = 1$ and $x \land x_n = 0$, then

$x_n$

= $x_n \land 1$ by **Idn$_2$**

= $1 \land x_n$ by **Cmm$_2$**

= $(x \lor \neg x) \land x_n$ by **Cpl$_1$**

= $x_n \land (x \lor \neg x)$ by **Cmm$_2$**

= $(x_n \land x) \lor (x_n \land \neg x)$ by **Dst$_2$**

= $(x \land x_n) \lor (\neg x \land x_n)$ by **Cmm$_2$**

= $0 \lor (\neg x \land x_n)$ by assumption

= $(x \land \neg x) \lor (\neg x \land x_n)$ by **Cpl$_2$**

= $(\neg x \land x) \lor (\neg x \land x_n)$ by **Cmm$_2$**

= $\neg x \land (x \lor x_n)$ by **Dst$_2$**

= $\neg x \land 1$ by assumption

= $\neg x$ by **Idn$_2$**

**DNg** $\boldsymbol{\neg\neg x = x}$

Proof: $\neg x \lor x = x \lor \neg x = 1$ by **Cmm$_1$**, **Cpl$_1$**

and $\neg x \land x = x \land \neg x = 0$ by **Cmm$_2$**, **Cpl$_2$**

hence $x = \neg\neg x$ by **UNg**

**A$_1$** $\boldsymbol{x \lor (\neg x \lor y) = 1}$

Proof: $x \lor (\neg x \lor y)$

= $(x \lor (\neg x \lor y)) \land 1$ by **Idn$_2$**

= $1 \land (x \lor (\neg x \lor y))$ by **Cmm$_2$**

= $(x \lor \neg x) \land (x \lor (\neg x \lor y))$ by **Cpl$_1$**

= $x \lor (\neg x \land (\neg x \lor y))$ by **Dst$_1$**

= $x \lor \neg x$ by **Abs$_2$**

= $1$ by **Cpl$_1$**

**A$_2$** [dual] $x \land (\neg x \land y) = 0$

| | |
|---|---|
| **$B_1$** **$(x \lor y) \lor (\neg x \land \neg y) = 1$**<br>Proof: $(x \lor y) \lor (\neg x \land \neg y)$<br>$= ((x \lor y) \lor \neg x) \land ((x \lor y) \lor \neg y)$ by **$Dst_1$**<br>$= (\neg x \lor (x \lor y)) \land (\neg y \lor (y \lor x))$ by **$Cmm_1$**<br>$= (\neg x \lor (\neg\neg x \lor y)) \land (\neg y \lor (\neg\neg y \lor x))$ by **DNg**<br>$= 1 \land 1$ by **$A_1$**<br>$= 1$ by **$Idn_2$** | **$B_1$** [dual] $(x \land y) \land (\neg x \lor \neg y) = 0$ |
| **$C_1$** **$(x \lor y) \land (\neg x \land \neg y) = 0$**<br>Proof: $(x \lor y) \land (\neg x \land \neg y)$<br>$= (\neg x \land \neg y) \land (x \lor y)$ by **$Cmm_2$**<br>$= ((\neg x \land \neg y) \land x) \lor ((\neg x \land \neg y) \land y)$ by **$Dst_2$**<br>$= (x \land (\neg x \land \neg y)) \lor (y \land (\neg y \land \neg x))$ by **$Cmm_2$**<br>$= 0 \lor 0$ by **$A_2$**<br>$= 0$ by **$Idn_1$** | **$C_2$** [dual] $(x \land y) \lor (\neg x \lor \neg y) = 1$ |
| **$DMg_1$** **$\neg(x \lor y) = \neg x \land \neg y$**<br>Proof: by **$B_1$**, **$C_1$**, and **UNg** | **$DMg_2$** [dual] $\neg(x \land y) = \neg x \lor \neg y$ |
| **$D_1$** **$(x \lor (y \lor z)) \lor \neg x = 1$**<br>Proof: $(x \lor (y \lor z)) \lor \neg x$<br>$= \neg x \lor (x \lor (y \lor z))$ by **$Cmm_1$**<br>$= \neg x \lor (\neg\neg x \lor (y \lor z))$ by **DNg**<br>$= 1$ by **$A_1$** | **$D_2$** [dual] $(x \land (y \land z)) \land \neg x = 0$ |
| **$E_1$** **$y \land (x \lor (y \lor z)) = y$**<br>Proof: $y \land (x \lor (y \lor z))$<br>$= (y \land x) \lor (y \land (y \lor z))$ by **$Dst_2$**<br>$= (y \land x) \lor y$ by **$Abs_2$**<br>$= y \lor (y \land x)$ by **$Cmm_1$**<br>$= y$ by **$Abs_1$** | **$E_2$** [dual] $y \lor (x \land (y \land z)) = y$ |

| | |
|---|---|
| $\mathbf{F_1}$ $\boldsymbol{(x\vee(y\vee z))\vee\neg y=1}$<br>Proof: $(x\vee(y\vee z))\vee\neg y$<br>$=\neg y\vee(x\vee(y\vee z))$ by $\mathbf{Cmm_1}$<br>$=(\neg y\vee(x\vee(y\vee z)))\wedge 1$ by $\mathbf{Idn_2}$<br>$=1\wedge(\neg y\vee(x\vee(y\vee z)))$ by $\mathbf{Cmm_2}$<br>$=(y\vee\neg y)\wedge(\neg y\vee(x\vee(y\vee z)))$ by $\mathbf{Cpl_1}$<br>$=(\neg y\vee y)\wedge(\neg y\vee(x\vee(y\vee z)))$ by $\mathbf{Cmm_1}$<br>$=\neg y\vee(y\wedge(x\vee(y\vee z)))$ by $\mathbf{Dst_1}$<br>$=\neg y\vee y$ by $\mathbf{E_1}$<br>$=y\vee\neg y$ by $\mathbf{Cmm_1}$<br>$=1$ by $\mathbf{Cpl_1}$ | $\mathbf{F_2}$ [dual] $(x\wedge(y\wedge z))\wedge\neg y=0$ |
| $\mathbf{G_1}$ $\boldsymbol{(x\vee(y\vee z))\vee\neg z=1}$<br>Proof: $(x\vee(y\vee z))\vee\neg z$<br>$=(x\vee(z\vee y))\vee\neg z$ by $\mathbf{Cmm_1}$<br>$=1$ by $\mathbf{F_1}$ | $\mathbf{G_2}$ [dual] $(x\wedge(y\wedge z))\wedge\neg z=0$ |
| $\mathbf{H_1}$ $\boldsymbol{\neg((x\vee y)\vee z)\wedge x=0}$<br>Proof: $\neg((x\vee y)\vee z)\wedge x$<br>$=(\neg(x\vee y)\wedge\neg z)\wedge x$ by $\mathbf{DMg_1}$<br>$=((\neg x\wedge\neg y)\wedge\neg z)\wedge x$ by $\mathbf{DMg_1}$<br>$=x\wedge((\neg x\wedge\neg y)\wedge\neg z)$ by $\mathbf{Cmm_2}$<br>$=(x\wedge((\neg x\wedge\neg y)\wedge\neg z))\vee 0$ by $\mathbf{Idn_1}$<br>$=0\vee(x\wedge((\neg x\wedge\neg y)\wedge\neg z))$ by $\mathbf{Cmm_1}$<br>$=(x\wedge\neg x)\vee(x\wedge((\neg x\wedge\neg y)\wedge\neg z))$ by $\mathbf{Cpl_1}$<br>$=x\wedge(\neg x\vee((\neg x\wedge\neg y)\wedge\neg z))$ by $\mathbf{Dst_2}$<br>$=x\wedge(\neg x\vee(\neg z\wedge(\neg x\wedge\neg y)))$ by $\mathbf{Cmm_2}$<br>$=x\wedge\neg x$ by $\mathbf{E_2}$<br>$=0$ by $\mathbf{Cpl_2}$ | $\mathbf{H_2}$ [dual] $\neg((x\wedge y)\wedge z)\vee x=1$ |
| $\mathbf{I_1}$ $\boldsymbol{\neg((x\vee y)\vee z)\wedge y=0}$<br>Proof: $\neg((x\vee y)\vee z)\wedge y$<br>$=\neg((y\vee x)\vee z)\wedge y$ by $\mathbf{Cmm_1}$<br>$=0$ by $\mathbf{H_1}$ | $\mathbf{I_2}$ [dual] $\neg((x\wedge y)\wedge z)\vee y=1$ |

**$J_1$** $\quad \neg((x \lor y) \lor z) \land z = 0$

Proof: $\neg((x \lor y) \lor z) \land z$

$= (\neg(x \lor y) \land \neg z) \land z$ by **$DMg_1$**

$= z \land (\neg(x \lor y) \land \neg z)$ by **$Cmm_2$**

$= z \land ( \neg z \land \neg(x \lor y))$ by **$Cmm_2$**

$= 0$ by **$A_2$**

**$J_2$** [dual] $\neg((x \land y) \land z) \lor z = 1$

**$K_1$** $\quad (x \lor (y \lor z)) \lor \neg((x \lor y) \lor z) = 1$

Proof: $(x \lor (y \lor z)) \lor \neg((x \lor y) \lor z)$

$= (x \lor (y \lor z)) \lor (\neg(x \lor y) \land \neg z)$ by **$DMg_1$**

$= (x \lor (y \lor z)) \lor ((\neg x \land \neg y) \land \neg z)$ by **$DMg_1$**

$= ((x \lor (y \lor z)) \lor (\neg x \land \neg y)) \land ((x \lor (y \lor z)) \lor \neg z)$ by **$Dst_1$**

$= (((x \lor (y \lor z)) \lor \neg x) \land ((x \lor (y \lor z)) \lor \neg y)) \land ((x \lor (y \lor z)) \lor \neg z)$ by **$Dst_1$**

$= (1 \land 1) \land 1$ by **$D_1$,$F_1$,$G_1$**

$= 1$ by **$Idn_2$**

**$K_2$** [dual] $(x \land (y \land z)) \land \neg((x \land y) \land z) = 0$

**$L_1$** $\quad (x \lor (y \lor z)) \land \neg((x \lor y) \lor z) = 0$

Proof: $(x \lor (y \lor z)) \land \neg((x \lor y) \lor z)$

$= \neg((x \lor y) \lor z) \land (x \lor (y \lor z))$ by **$Cmm_2$**

$= (\neg((x \lor y) \lor z) \land x) \lor (\neg((x \lor y) \lor z) \land (y \lor z))$ by **$Dst_2$**

$= (\neg((x \lor y) \lor z) \land x) \lor ((\neg((x \lor y) \lor z) \land y) \lor (\neg((x \lor y) \lor z) \land z))$ by **$Dst_2$**

$= (0 \lor 0) \lor 0$ by **$H_1$,$I_1$,$J_1$**

$= 0$ by **$Idn_1$**

**$L_2$** [dual] $(x \land (y \land z)) \lor \neg((x \land y) \land z) = 1$

**$Ass_1$** $\quad x \lor (y \lor z) = (x \lor y) \lor z$

Proof: by **$K_1$**, **$L_1$**, **UNg**, **DNg**

**$Ass_2$** [dual] $x \land (y \land z) = (x \land y) \land z$

**Abbreviations**

| | |
|---|---|
| **UId** | Unique Identity |
| **Idm** | Idempotence |
| **Bnd** | Boundaries |
| **Abs** | Absorption law |
| **UNg** | Unique Negation |
| **DNg** | Double negation |
| **DMg** | De Morgan's Law |
| **Ass** | Associativity |

| **Huntington 1904 Boolean algebra axioms** | | | |
|---|---|---|---|
| **Idn**$_1$ | $x \vee 0 = x$ | **Idn**$_2$ | $x \wedge 1 = x$ |
| **Cmm**$_1$ | $x \vee y = y \vee x$ | **Cmm**$_2$ | $x \wedge y = y \wedge x$ |
| **Dst**$_1$ | $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ | **Dst**$_2$ | $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ |
| **Cpl**$_1$ | $x \vee \neg x = 1$ | **Cpl**$_2$ | $x \wedge \neg x = 0$ |

**Abbreviations**

| | |
|---|---|
| **Idn** | Identity |
| **Cmm** | Commutativity |
| **Dst** | Distributivity |
| **Cpl** | Complements |

The first axiomatization of Boolean lattices/algebras in general was given by Alfred North Whitehead in 1898.[4][5] It included the above axioms and additionally $x \vee 1=1$ and $x \wedge 0=0$. In 1904, the American mathematician Edward V. Huntington (1874–1952) gave probably the most parsimonious axiomatization based on ∧, ∨, ¬, even proving the associativity laws (see box).[6] He also proved that these axioms are independent of each other.[7] In 1933, Huntington set out the following elegant axiomatization for Boolean algebra. It requires just one binary operation + and a unary functional symbol $n$, to be read as 'complement', which satisfy the following laws:

1. *Commutativity*: $x + y = y + x$.
2. *Associativity*: $(x + y) + z = x + (y + z)$.
3. *Huntington equation*: $n(n(x) + y) + n(n(x) + n(y)) = x$.

Herbert Robbins immediately asked: If the Huntington equation is replaced with its dual, to wit:

4. *Robbins Equation*: $n(n(x + y) + n(x + n(y))) = x$,

do (1), (2), and (4) form a basis for Boolean algebra? Calling (1), (2), and (4) a *Robbins algebra*, the question then becomes: Is every Robbins algebra a Boolean algebra? This question (which came to be known as the Robbins conjecture) remained open for decades, and became a favorite question of Alfred Tarski and his students. In 1996, William McCune at Argonne National Laboratory, building on earlier work by Larry Wos, Steve Winker, and Bob Veroff, answered Robbins's question in the affirmative: Every Robbins algebra is a Boolean algebra. Crucial to McCune's proof was the automated reasoning program EQP he designed. For a simplification of McCune's proof, see Dahn (1998).

## Generalizations

**Algebraic structures**

- v
- t
- e[8]

Removing the requirement of existence of a unit from the axioms of Boolean algebra yields "generalized Boolean algebras". Formally, a distributive lattice *B* is a generalized Boolean lattice, if it has a smallest element 0 and for any elements *a* and *b* in *B* such that $a \leq b$, there exists an element *x* such that $a \wedge x = 0$ and $a \vee x = b$. Defining $a \setminus b$ as the unique *x* such that $(a \wedge b) \vee x = a$ and $(a \wedge b) \wedge x = 0$, we say that the structure $(B,\wedge,\vee,\setminus,0)$ is a *generalized Boolean algebra*, while $(B,\vee,0)$ is a *generalized Boolean semilattice*. Generalized Boolean lattices are exactly the ideals of Boolean lattices.

A structure that satisfies all axioms for Boolean algebras except the two distributivity axioms is called an orthocomplemented lattice. Orthocomplemented lattices arise naturally in quantum logic as lattices of closed subspaces for separable Hilbert spaces.

## Notes

[1] Davey, Priestley, 1990, p.109, 131, 144
[2] Stone, 1936
[3] Hsiang, 1985, p.260
[4] Padmanabhan, p. 73 (http://books.google.com/books?id=JlXSlpmlSv4C&pg=PA73#v=onepage&q&f=false)
[5] Whitehead, 1898, p.37
[6] Huntington, 1904, p.292-293, (first of several axiomatizations by Huntington)
[7] Huntington, 1904, p.296
[8] http://en.wikipedia.org/w/index.php?title=Template:Algebraic_structures&action=edit

## References

- Brown, Stephen; Vranesic, Zvonko (2002), *Fundamentals of Digital Logic with VHDL Design* (2nd ed.), McGraw–Hill, ISBN 978-0-07-249938-4. See Section 2.5.
- A. Boudet, J.P. Jouannaud, M. Schmidt-Schauß (1989). "Unification in Boolean Rings and Abelian Groups" (http://www.sciencedirect.com/science/article/pii/S0747717189800549/pdf?md5=713ed362e4b6f2db53923cc5ed47c818&pid=1-s2.0-S0747717189800549-main.pdf). *Journal of Symbolic Computation* **8**: 449–477.
- Cori, Rene; Lascar, Daniel (2000), *Mathematical Logic: A Course with Exercises*, Oxford University Press, ISBN 978-0-19-850048-3. See Chapter 2.
- Dahn, B. I. (1998), "Robbins Algebras are Boolean: A Revision of McCune's Computer-Generated Solution of the Robbins Problem", *Journal of Algebra* **208** (2): 526–532, doi: 10.1006/jabr.1998.7467 (http://dx.doi.org/10.1006/jabr.1998.7467).
- B.A. Davey, H.A. Priestley (1990). *Introduction to Lattices and Order*. Cambridge Mathematical Textbooks. Cambridge University Press.
- Givant, Steven; Halmos, Paul (2009), *Introduction to Boolean Algebras*, Undergraduate Texts in Mathematics, Springer, ISBN 978-0-387-40293-2.
- Halmos, Paul (1963), *Lectures on Boolean Algebras*, Van Nostrand, ISBN 978-0-387-90094-0.
- Halmos, Paul; Givant, Steven (1998), *Logic as Algebra*, Dolciani Mathematical Expositions **21**, Mathematical Association of America, ISBN 978-0-88385-327-6.

- Hsiang, Jieh (1985). "Refutational Theorem Proving Using Term Rewriting Systems" (http://www.researchgate.net/publication/223327412_Refutational_theorem_proving_using_term-rewriting_systems/file/60b7d5193580e500f1.pdf). *AI* **25**: 255–300.
- Edward V. Huntington (1904). "Sets of Independent Postulates for the Algebra of Logic" (http://www.jstor.org/stable/pdfplus/1986459.pdf). *These Transactions* **5**: 288–309.
- Huntington, E. V. (1933), "New sets of independent postulates for the algebra of logic" (http://www.ams.org/journals/tran/1933-035-01/S0002-9947-1933-1501684-X/S0002-9947-1933-1501684-X.pdf), *Transactions of the American Mathematical Society* (American Mathematical Society) **35** (1): 274–304, doi: 10.2307/1989325 (http://dx.doi.org/10.2307/1989325), JSTOR 1989325 (http://www.jstor.org/stable/1989325).
- Huntington, E. V. (1933), "Boolean algebra: A correction", *Transactions of the American Mathematical Society* (American Mathematical Society) **35** (2): 557–558, doi: 10.2307/1989783 (http://dx.doi.org/10.2307/1989783), JSTOR 1989783 (http://www.jstor.org/stable/1989783).
- Mendelson, Elliott (1970), *Boolean Algebra and Switching Circuits*, Schaum's Outline Series in Mathematics, McGraw–Hill, ISBN 978-0-07-041460-0.
- Monk, J. Donald; Bonnet, R., eds. (1989), *Handbook of Boolean Algebras*, North-Holland, ISBN 978-0-444-87291-3. In 3 volumes. (Vol.1:ISBN 978-0-444-70261-6, Vol.2:ISBN 978-0-444-87152-7, Vol.3:ISBN 978-0-444-87153-4)
- Padmanabhan, Ranganathan; Rudeanu, Sergiu (2008), *Axioms for lattices and boolean algebras*, World Scientific, ISBN 978-981-283-454-6.
- Sikorski, Roman (1966), *Boolean Algebras*, Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer Verlag.
- Stoll, R. R. (1963), *Set Theory and Logic*, W. H. Freeman, ISBN 978-0-486-63829-4. Reprinted by Dover Publications, 1979.
- Marshall H. Stone (1936). "The Theory of Representations for Boolean Algebra". *Trans. AMS* **40**: 37–111.
- A.N. Whitehead (1898). *A Treatise on Universal Algebra* (http://projecteuclid.org/euclid.chmm/1263316509). Cambridge University Press. ISBN 1-4297-0032-7.

## External links

- Hazewinkel, Michiel, ed. (2001), "Boolean algebra" (http://www.encyclopediaofmath.org/index.php?title=p/b016920), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Boolean Algebra (http://www.allaboutcircuits.com/vol_4/chpt_7/1.html) from AllAboutCircuits
- Stanford Encyclopedia of Philosophy: " The Mathematics of Boolean Algebra, (http://plato.stanford.edu/entries/boolalg-math/)" by J. Donald Monk.
- McCune W., 1997. *Robbins Algebras Are Boolean* (http://www.cs.unm.edu/~mccune/papers/robbins/) JAR 19(3), 263—276
- "Boolean Algebra" (http://demonstrations.wolfram.com/BooleanAlgebra/) by Eric W. Weisstein, Wolfram Demonstrations Project, 2007.

A monograph available free online:

- Burris, Stanley N.; Sankappanavar, H. P., 1981. *A Course in Universal Algebra.* (http://www.thoralf.uwaterloo.ca/htdocs/ualg.html) Springer-Verlag. ISBN 3-540-90578-2.
- Weisstein, Eric W., " Boolean Algebra (http://mathworld.wolfram.com/BooleanAlgebra.html)", *MathWorld*.

# Algebraic normal form

In Boolean algebra, the **algebraic normal form (ANF)**, **Zhegalkin normal form**, or **Reed–Muller expansion** is a way of writing logical formulas in one of three subforms:

- The entire formula is purely true or false:

  $1$

  $0$
- One or more variables are ANDed together into a term. One or more terms are XORed together into ANF. No NOTs are permitted:

  $a \oplus b \oplus ab \oplus abc$
- The previous subform with a purely true term:

  $1 \oplus a \oplus b \oplus ab \oplus abc$

Formulas written in ANF are also known as Zhegalkin polynomials (Russian: полиномы Жегалкина) and Positive Polarity (or Parity) Reed–Muller expressions.

## Common uses

ANF is a normal form, which means that two equivalent formulas will convert to the same ANF, easily showing whether two formulas are equivalent for automated theorem proving. Unlike other normal forms, it can be represented as a simple list of lists of variable names. Conjunctive and disjunctive normal forms also require recording whether each variable is negated or not. Negation normal form is unsuitable for that purpose, since it doesn't use equality as its equivalence relation: $a \vee \neg a$ isn't reduced to the same thing as 1, even though they're equal.

Putting a formula into ANF also makes it easy to identify linear functions (used, for example, in linear feedback shift registers): a linear function is one that is a sum of single literals. Properties of nonlinear feedback shift registers can also be deduced from certain properties of the feedback function in ANF.

## Performing operations within algebraic normal form

There are straightforward ways to perform the standard boolean operations on ANF inputs in order to get ANF results.

XOR (logical exclusive disjunction) is performed directly:

$(1 \oplus x) \oplus (1 \oplus x \oplus y)$

$1 \oplus x \oplus 1 \oplus x \oplus y$

$1 \oplus 1 \oplus x \oplus x \oplus y$

$y$

NOT (logical negation) is XORing 1:[1]

$\neg(1 \oplus x \oplus y)$

$1 \oplus (1 \oplus x \oplus y)$

$1 \oplus 1 \oplus x \oplus y$

$x \oplus y$

AND (logical conjunction) is distributed algebraically[2]

$(1 \oplus x)(1 \oplus x \oplus y)$

$1(1 \oplus x \oplus y) \oplus x(1 \oplus x \oplus y)$

(1 ⊕ x ⊕ y) ⊕ (x ⊕ x ⊕ xy)

1 ⊕ x ⊕ x ⊕ x ⊕ y ⊕ xy

1 ⊕ x ⊕ y ⊕ xy

OR (logical disjunction) uses either 1 ⊕ (1 ⊕ a)(1 ⊕ b)[3] (easier when both operands have purely true terms) or a ⊕ b ⊕ ab[4] (easier otherwise):

(1 ⊕ x) + (1 ⊕ x ⊕ y)

1 ⊕ (1 ⊕ 1 ⊕ x)(1 ⊕ 1 ⊕ x ⊕ y)

1 ⊕ x(x ⊕ y)

1 ⊕ x ⊕ xy

## Converting to algebraic normal form

Each variable in a formula is already in pure ANF, so you only need to perform the formula's boolean operations as shown above to get the entire formula into ANF. For example:

x + (y · ¬z)

x + (y(1 ⊕ z))

x + (y ⊕ yz)

x ⊕ (y ⊕ yz) ⊕ x(y ⊕ yz)

x ⊕ y ⊕ xy ⊕ yz ⊕ xyz

## Formal representation

ANF is sometimes described in an equivalent way:

$$\begin{aligned} f(x_1, x_2, \ldots, x_n) = \; & a_0 \oplus \\ & a_1 x_1 \oplus a_2 x_2 \oplus \cdots \oplus a_n x_n \oplus \\ & a_{1,2} x_1 x_2 \oplus \cdots \oplus a_{n-1,n} x_{n-1} x_n \oplus \\ & \cdots \oplus \\ & a_{1,2,\ldots,n} x_1 x_2 \ldots x_n \end{aligned}$$

where $a_0, a_1, \ldots, a_{1,2,\ldots,n} \in \{0,1\}^*$ fully describes $f$.

### Recursively deriving multiargument Boolean functions

There are only four functions with one argument:

- $f(x) = 0$
- $f(x) = 1$
- $f(x) = x$
- $f(x) = 1 \oplus x$

To represent a function with multiple arguments one can use the following equality:

$f(x_1, x_2, \ldots, x_n) = g(x_2, \ldots, x_n) \oplus x_1 h(x_2, \ldots, x_n)$, where

- $g(x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n)$
- $h(x_2, \ldots, x_n) = f(0, x_2, \ldots, x_n) \oplus f(1, x_2, \ldots, x_n)$

Indeed,

- if $x_1 = 0$ then $x_1 h = 0$ and so $f(0, \ldots) = f(0, \ldots)$

- if $x_1 = 1$ then $x_1 h = h$ and so $f(1,\ldots) = f(0,\ldots) \oplus f(0,\ldots) \oplus f(1,\ldots)$

Since both $g$ and $h$ have fewer arguments than $f$ it follows that using this process recursively we will finish with functions with one variable. For example, let us construct ANF of $f(x,y) = x \vee y$ (logical or):

- $f(x,y) = f(0,y) \oplus x(f(0,y) \oplus f(1,y))$
- since $f(0,y) = 0 \vee y = y$ and $f(1,y) = 1 \vee y = 1$
- it follows that $f(x,y) = y \oplus x(y \oplus 1)$
- by distribution, we get the final ANF: $f(x,y) = y \oplus xy \oplus x = x \oplus y \oplus xy$

## References

[1] WolframAlpha NOT-equivalence demonstration: ¬a = 1 ⊕ a (http://www.wolframalpha.com/input/?i=simplify+1+xor+a)

[2] WolframAlpha AND-equivalence demonstration: (a ⊕ b)(c ⊕ d) = ac ⊕ ad ⊕ bc ⊕ bd (http://www.wolframalpha.com/input/?i=(a+xor+b)+and+(c+xor+d)+in+anf)

[3] From De Morgan's laws

[4] WolframAlpha OR-equivalence demonstration: a + b = a ⊕ b ⊕ ab (http://www.wolframalpha.com/input/?i=simplify+a+xor+b+xor+(a+and+b))

# Universal quantification

In predicate logic, a **universal quantification** is a type of quantifier, a logical constant which is interpreted as "given any" or "for all". It expresses that a propositional function can be satisfied by every member of a domain of discourse. In other terms, it is the predication of a property or relation to every member of the domain. It asserts that a predicate within the scope of a universal quantifier is true of every value of a predicate variable.

It is usually denoted by the turned A (∀) logical operator symbol, which, when used together with a predicate variable, is called a **universal quantifier** ("∀x", "∀(x)", or sometimes by "(x)" alone). Universal quantification is distinct from *existential* quantification ("there exists"), which asserts that the property or relation holds only for at least one member of the domain.

Quantification in general is covered in the article on quantification. Symbols are encoded U+2200 ∀ for all (HTML: `&#8704;` `&forall;` as a mathematical symbol).

## Basics

Suppose it is given that

2·0 = 0 + 0, and 2·1 = 1 + 1, and 2·2 = 2 + 2, etc.

This would seem to be a logical conjunction because of the repeated use of "and." However, the "etc." cannot be interpreted as a conjunction in formal logic. Instead, the statement must be rephrased:

For all natural numbers *n*, 2·*n* = *n* + *n*.

This is a single statement using universal quantification.

This statement can be said to be more precise than the original one. While the "etc." informally includes natural numbers, and nothing more, this was not rigorously given. In the universal quantification, on the other hand, the natural numbers are mentioned explicitly.

This particular example is true, because any natural number could be substituted for *n* and the statement "2·*n* = *n* + *n*" would be true. In contrast,

For all natural numbers *n*, 2·*n* > 2 + *n*

is false, because if *n* is substituted with, for instance, 1, the statement "2·1 > 2 + 1" is false. It is immaterial that "2·*n* > 2 + *n*" is true for *most* natural numbers *n*: even the existence of a single counterexample is enough to prove the universal quantification false.

On the other hand, for all composite numbers $n$, $2\cdot n > 2 + n$ is true, because none of the counterexamples are composite numbers. This indicates the importance of the *domain of discourse*, which specifies which values $n$ can take.[1] In particular, note that if the domain of discourse is restricted to consist only of those objects that satisfy a certain predicate, then for universal quantification this requires a logical conditional. For example,

For all composite numbers $n$, $2\cdot n > 2 + n$

is logically equivalent to

For all natural numbers $n$, if $n$ is composite, then $2\cdot n > 2 + n$.

Here the "if ... then" construction indicates the logical conditional.

### Notation

In symbolic logic, the universal quantifier symbol $\forall$ (an inverted "A" in a sans-serif font, Unicode 0x2200) is used to indicate universal quantification.[2]

For example, if $P(n)$ is the predicate "$2\cdot n > 2 + n$" and **N** is the set of natural numbers, then:

$$\forall n \in \mathbb{N}\; P(n)$$

is the (false) statement:

For all natural numbers $n$, $2\cdot n > 2 + n$.

Similarly, if $Q(n)$ is the predicate "$n$ is composite", then

$$\forall n \in \mathbb{N}\; \big(Q(n) \rightarrow P(n)\big)$$

is the (true) statement:

For all natural numbers $n$, if $n$ is composite, then $2\cdot n > 2 + \mathrm{n}$

and since "$n$ is composite" implies that $n$ must already be a natural number, we can shorten this statement to the equivalent:

$$\forall n\; \big(Q(n) \rightarrow P(n)\big)$$

For all composite numbers $n$, $2\cdot n > 2 + n$.

Several variations in the notation for quantification (which apply to all forms) can be found in the quantification article. There is a special notation used only for universal quantification, which is given:

$$(n{\in}\mathbb{N})\, P(n)$$

The parentheses indicate universal quantification by default.

## Properties

### Negation

Note that a quantified propositional function is a statement; thus, like statements, quantified functions can be negated. The notation most mathematicians and logicians utilize to denote negation is: $\neg$. However, some (such as Douglas Hofstadter) use the tilde (~).

For example, if P($x$) is the propositional function "x is married", then, for a Universe of Discourse X of all living human beings, the universal quantification

Given any living person $x$, that person is married

is given:

$$\forall x{\in}\mathbf{X}\, P(x)$$

It can be seen that this is irrevocably false. Truthfully, it is stated that

It is not the case that, given any living person *x*, that person is married

or, symbolically:

$$\lnot\ \forall x{\in}\mathbf{X}\, P(x).$$

If the statement is not true for *every* element of the Universe of Discourse, then, presuming the universe of discourse is non-empty, there must be at least one element for which the statement is false. That is, the negation of $\forall x{\in}\mathbf{X}\, P(x)$ is logically equivalent to "There exists a living person *x* such that he is not married", or:

$$\exists x{\in}\mathbf{X}\, \lnot P(x)$$

Generally, then, the negation of a propositional function's universal quantification is an existential quantification of that propositional function's negation; symbolically,

$$\lnot\ \forall x{\in}\mathbf{X}\, P(x) \equiv\ \exists x{\in}\mathbf{X}\, \lnot P(x)$$

It is erroneous to state "all persons are not married" (i.e. "there exists no person who is married") when it is meant that "not all persons are married" (i.e. "there exists a person who is not married"):

$$\lnot\ \exists x{\in}\mathbf{X}\, P(x) \equiv\ \forall x{\in}\mathbf{X}\, \lnot P(x) \not\equiv\ \lnot\ \forall x{\in}\mathbf{X}\, P(x) \equiv\ \exists x{\in}\mathbf{X}\, \lnot P(x)$$

## Other connectives

The universal (and existential) quantifier moves unchanged across the logical connectives $\land$, $\lor$, $\to$, and $\nleftarrow$, as long as the other operand is not affected; that is:

$$P(x) \land (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \land Q(y))$$
$$P(x) \lor (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \lor Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$
$$P(x) \to (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \to Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$
$$P(x) \nleftarrow (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \nleftarrow Q(y))$$
$$P(x) \land (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \land Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$
$$P(x) \lor (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \lor Q(y))$$
$$P(x) \to (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \to Q(y))$$
$$P(x) \nleftarrow (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \nleftarrow Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$

Conversely, for the logical connectives $\uparrow$, $\downarrow$, $\nrightarrow$, and $\leftarrow$, the quantifiers flip:

$$P(x) \uparrow (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \uparrow Q(y))$$
$$P(x) \downarrow (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \downarrow Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$
$$P(x) \nrightarrow (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \nrightarrow Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$
$$P(x) \leftarrow (\exists y{\in}\mathbf{Y}\, Q(y)) \equiv\ \forall y{\in}\mathbf{Y}\, (P(x) \leftarrow Q(y))$$
$$P(x) \uparrow (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \uparrow Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$
$$P(x) \downarrow (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \downarrow Q(y))$$
$$P(x) \nrightarrow (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \nrightarrow Q(y))$$
$$P(x) \leftarrow (\forall y{\in}\mathbf{Y}\, Q(y)) \equiv\ \exists y{\in}\mathbf{Y}\, (P(x) \leftarrow Q(y)),\ \text{provided that } \mathbf{Y} \neq \emptyset$$

### Rules of inference

A rule of inference is a rule justifying a logical step from hypothesis to conclusion. There are several rules of inference which utilize the universal quantifier.

*Universal instantiation* concludes that, if the propositional function is known to be universally true, then it must be true for any arbitrary element of the Universe of Discourse. Symbolically, this is represented as

$$\forall x \in \mathbf{X}\, P(x) \to \; P(c)$$

where *c* is a completely arbitrary element of the Universe of Discourse.

*Universal generalization* concludes the propositional function must be universally true if it is true for any arbitrary element of the Universe of Discourse. Symbolically, for an arbitrary *c*,

$$P(c) \to \; \forall x \in \mathbf{X}\, P(x).$$

The element *c* must be completely arbitrary; else, the logic does not follow: if *c* is not arbitrary, and is instead a specific element of the Universe of Discourse, then P(*c*) only implies an existential quantification of the propositional function.

### The empty set

By convention, the formula $\forall x \in \emptyset\, P(x)$ is always true, regardless of the formula *P*(*x*); see vacuous truth.

## Universal closure

The **universal closure** of a formula φ is the formula with no free variables obtained by adding a universal quantifier for every free variable in φ. For example, the universal closure of

$$P(y) \wedge \exists x Q(x, z)$$

is

$$\forall y \forall z (P(y) \wedge \exists x Q(x, z)).$$

## As adjoint

In category theory and the theory of elementary topoi, the universal quantifier can be understood as the right adjoint of a functor between power sets, the inverse image functor of a function between sets; likewise, the existential quantifier is the left adjoint.[3]

For a set $X$, let $\mathcal{P}X$ denote its powerset. For any function $f : X \to Y$ between sets $X$ and $Y$, there is an inverse image functor $f^* : \mathcal{P}Y \to \mathcal{P}X$ between powersets, that takes subsets of the codomain of *f* back to subsets of its domain. The left adjoint of this functor is the existential quantifier $\exists_f$ and the right adjoint is the universal quantifier $\forall_f$.

That is, $\exists_f : \mathcal{P}X \to \mathcal{P}Y$ is a functor that, for each subset $S \subset X$, gives the subset $\exists_f S \subset Y$ given by

$$\exists_f S = \{y \in Y | \text{ there exists } x \in S \text{ s.t. } f(x) = y\}.$$

Likewise, the universal quantifier $\forall_f : \mathcal{P}X \to \mathcal{P}Y$ is given by

$$\forall_f S = \{y \in Y | f(x) = y \text{ for all } x \in S\}.$$

The more familiar form of the quantifiers as used in first-order logic is obtained by taking the function *f* to be the projection operator $\pi : X \times \{T, F\} \to \{T, F\}$ where $\{T, F\}$ is the two-element set holding the values true, false, and subsets *S* to be predicates $S \subset X \times \{T, F\}$, so that

$$\exists_\pi S = \{y \mid \exists x\, S(x, y)\}$$

which is either a one-element set (false) or a two-element set (true).

The universal and existential quantifiers given above generalize to the presheaf category.

## Notes

[1] Further information on using domains of discourse with quantified statements can be found in the Quantification article.

[2] The inverted "A" was used in the 19th century by Charles Sanders Peirce as a logical symbol for 'un-American' ("unamerican").

Page 320 in Randall Dipert, " Peirce's deductive logic (http://books.google.com/books?id=3suPBY5qh-cC&pg=PR7&dq=Cheryl+Misak,+unamerican&source=gbs_selected_pages&cad=3#v=onepage&q=unAmerican&f=false)". In Cheryl Misak, ed. *The Cambridge Companion to Peirce*. 2004

[3] Saunders Mac Lane, Ieke Moerdijk, (1992) *Sheaves in Geometry and Logic* Springer-Verlag. ISBN 0-387-97710-4 *See page 58*

## References

- Hinman, P. (2005). *Fundamentals of Mathematical Logic*. A K Peters. ISBN 1-56881-262-0.
- Franklin, J. and Daoud, A. (2011). *Proof in Mathematics: An Introduction* (http://www.maths.unsw.edu.au/~jim/proofs.html). Kew Books. ISBN 978-0-646-54509-7. (ch. 2)

# Existential quantification

In predicate logic, an **existential quantification** is a type of quantifier, a logical constant which is interpreted as "there exists," "there is at least one," or "for some." It expresses that a propositional function can be satisfied by at least one member of a domain of discourse. In other terms, it is the predication of a property or relation to at least one member of the domain. It asserts that a predicate within the scope of an existential quantifier is true of at least one value of a predicate variable.

It is usually denoted by the turned E (∃) logical operator symbol, which, when used together with a predicate variable, is called an **existential quantifier** ("∃x" or "∃(x)"). Existential quantification is distinct from *universal* quantification ("for all"), which asserts that the property or relation holds for *all* members of the domain.

Symbols are encoded U+2203 ∃ there exists (HTML: `&#8707; &exist;` as a mathematical symbol) and U+2204 ∄ there does not exist (HTML: `&#8708;`).

## Basics

Consider a formula that states that some natural number multiplied by itself is 25.

$0 \cdot 0 = 25$, **or** $1 \cdot 1 = 25$, **or** $2 \cdot 2 = 25$, **or** $3 \cdot 3 = 25$, and so on.

This would seem to be a logical disjunction because of the repeated use of "or". However, the "and so on" makes this impossible to integrate and to interpret as a disjunction in formal logic. Instead, the statement could be rephrased more formally as

For some natural number $n$, $n \cdot n = 25$.

This is a single statement using existential quantification.

This statement is more precise than the original one, as the phrase "and so on" does not necessarily include all natural numbers, and nothing more. Since the domain was not stated explicitly, the phrase could not be interpreted formally. In the quantified statement, on the other hand, the natural numbers are mentioned explicitly.

This particular example is true, because 5 is a natural number, and when we substitute 5 for $n$, we produce "$5 \cdot 5 = 25$", which is true. It does not matter that "$n \cdot n = 25$" is only true for a single natural number, 5; even the existence of a single solution is enough to prove the existential quantification true. In contrast, "For some even number $n$, $n \cdot n = 25$" is false, because there are no even solutions.

The *domain of discourse*, which specifies which values the variable $n$ is allowed to take, is therefore of critical importance in a statement's trueness or falseness. Logical conjunctions are used to restrict the domain of discourse to fulfill a given predicate. For example:

For some positive odd number *n*, *n*·*n* = 25

is logically equivalent to

For some natural number *n*, *n* is odd and *n*·*n* = 25.

Here, "and" is the logical conjunction.

In symbolic logic, "∃" (a backwards letter "E" in a sans-serif font) is used to indicate existential quantification.[1] Thus, if *P*(*a*, *b*, *c*) is the predicate "*a*·*b* = c" and $\mathbb{N}$ is the set of natural numbers, then

$$\exists n{\in}\mathbb{N}\, P(n,n,25)$$

is the (true) statement

For some natural number *n*, *n*·*n* = 25.

Similarly, if *Q*(*n*) is the predicate "*n* is even", then

$$\exists n{\in}\mathbb{N}\, \big(Q(n) \wedge P(n,n,25)\big)$$

is the (false) statement

For some natural number *n*, *n* is even and *n*·*n* = 25.

In mathematics, the proof of a "some" statement may be achieved either by a constructive proof, which exhibits an object satisfying the "some" statement, or by a nonconstructive proof which shows that there must be such an object but without exhibiting one.

## Properties

### Negation

A quantified propositional function is a statement; thus, like statements, quantified functions can be negated. The $\neg$ symbol is used to denote negation.

For example, if P(*x*) is the propositional function "x is between 0 and 1", then, for a domain of discourse *X* of all natural numbers, the existential quantification "There exists a natural number *x* which is between 0 and 1" is symbolically stated:

$$\exists x{\in}\mathbf{X}\, P(x)$$

This can be demonstrated to be irrevocably false. Truthfully, it must be said, "It is not the case that there is a natural number *x* that is between 0 and 1", or, symbolically:

$$\neg\ \exists x{\in}\mathbf{X}\, P(x).$$

If there is no element of the domain of discourse for which the statement is true, then it must be false for all of those elements. That is, the negation of

$$\exists x{\in}\mathbf{X}\, P(x)$$

is logically equivalent to "For any natural number *x*, x is not between 0 and 1", or:

$$\forall x{\in}\mathbf{X}\, \neg P(x)$$

Generally, then, the negation of a propositional function's existential quantification is a universal quantification of that propositional function's negation; symbolically,

$$\neg\ \exists x{\in}\mathbf{X}\, P(x) \equiv\ \forall x{\in}\mathbf{X}\, \neg P(x)$$

A common error is stating "all persons are not married" (i.e. "there exists no person who is married") when "not all persons are married" (i.e. "there exists a person who is not married") is intended:

$$\neg\ \exists x{\in}\mathbf{X}\, P(x) \equiv\ \forall x{\in}\mathbf{X}\, \neg P(x) \not\equiv\ \neg\ \forall x{\in}\mathbf{X}\, P(x) \equiv\ \exists x{\in}\mathbf{X}\, \neg P(x)$$

Negation is also expressible through a statement of "for no", as opposed to "for some":

$$\nexists x{\in}\mathbf{X}\, P(x) \equiv \neg\ \exists x{\in}\mathbf{X}\, P(x)$$

Unlike the universal quantifier, the existential quantifier distributes over logical disjunctions:

$$\exists x{\in}\mathbf{X}\, P(x) \lor Q(x) \to\ (\exists x{\in}\mathbf{X}\, P(x) \lor \exists x{\in}\mathbf{X}\, Q(x))$$

## Rules of Inference

| **Transformation rules** |
|---|
| **Propositional calculus** |
| Rules of inference |
| • *Modus ponens* |
| • *Modus tollens* |
| • Biconditional introduction |
| • Biconditional elimination |
| • Conjunction introduction |
| • Simplification |
| • Disjunction introduction |
| • Disjunction elimination |
| • Disjunctive syllogism |
| • Hypothetical syllogism |
| • Constructive dilemma |
| • Destructive dilemma |
| • Absorption |
| • Negation Introduction |
| Rules of replacement |
| • Associativity |
| • Commutativity |
| • Distributivity |
| • Double negation |
| • De Morgan's laws |
| • Transposition |
| • Material implication |
| • Material equivalence |
| • Exportation |
| • Tautology |
| **Predicate logic** |
| • Universal generalization |
| • Universal instantiation |
| • Existential generalization |
| • Existential instantiation |
| • v |
| • t |
| • e [1] |

A rule of inference is a rule justifying a logical step from hypothesis to conclusion. There are several rules of inference which utilize the existential quantifier.

*Existential introduction* (∃I) concludes that, if the propositional function is known to be true for a particular element of the domain of discourse, then it must be true that there exists an element for which the proposition function is true. Symbolically,

$$P(a) \to\ \exists x{\in}\mathbf{X}\, P(x)$$

Existential elimination, when conducted in a Fitch style deduction, proceeds by entering a new sub-derivation while substituting an existentially quantified variable for a subject which does not appear within any active sub-derivation. If a conclusion can be reached within this sub-derivation in which the substituted subject does not appear, then one can exit that sub-derivation with that conclusion. The reasoning behind existential elimination (∃E) is as follows: If it is given that there exists an element for which the proposition function is true, and if a conclusion can be reached by giving that element an arbitrary name, that conclusion is necessarily true, as long as it does not contain the name. Symbolically, for an arbitrary *c* and for a proposition Q in which *c* does not appear:

$$\exists{x}{\in}\mathbf{X}\, P(x) \to\ ((P(c) \to\ Q) \to\ Q)$$

$P(c) \to\ Q$ must be true for all values of *c* over the same domain *X*; else, the logic does not follow: If *c* is not arbitrary, and is instead a specific element of the domain of discourse, then stating P(*c*) might unjustifiably give more information about that object.

### The empty set

The formula $\exists {x}{\in}\emptyset \, P(x)$ is always false, regardless of *P*(*x*). This is because $\emptyset$ denotes the empty set, and no *x* of any description – let alone an *x* fulfilling a given predicate *P*(*x*) – exist in the empty set. See also vacuous truth.

## As adjoint

In category theory and the theory of elementary topoi, the existential quantifier can be understood as the left adjoint of a functor between power sets, the inverse image functor of a function between sets; likewise, the universal quantifier is the right adjoint.[2]

## Notes

[1] This symbol is also known as the *existential operator*. It is sometimes represented with *V*.

[2] Saunders Mac Lane, Ieke Moerdijk, (1992) *Sheaves in Geometry and Logic* Springer-Verlag. ISBN 0-387-97710-4 *See page 58*

## References

- Hinman, P. (2005). *Fundamentals of Mathematical Logic*. A K Peters. ISBN 1-56881-262-0.

# Predicate logic

In mathematical logic, **predicate logic** is the generic term for symbolic formal systems like first-order logic, second-order logic, many-sorted logic, or infinitary logic. This formal system is distinguished from other systems in that its formulae contain variables which can be quantified. Two common quantifiers are the existential ∃ ("there exists") and universal ∀ ("for all") quantifiers. The variables could be elements in the universe under discussion, or perhaps relations or functions over that universe. For instance, an existential quantifier over a function symbol would be interpreted as modifier "there is a function". The foundations of predicate logic were developed independently by Gottlob Frege and Charles Peirce.[1]

In informal usage, the term "predicate logic" occasionally refers to first-order logic. Some authors consider the **predicate calculus** to be an axiomatized form of predicate logic, and the predicate logic to be derived from an informal, more intuitive development.[2]

Predicate logics also include logics mixing modal operators and quantifiers. See Modal logic, Saul Kripke, Barcan Marcus formulae, A. N. Prior, and Nicholas Rescher.

## Footnotes

[1] Eric M. Hammer: Semantics for Existential Graphs, *Journal of Philosophical Logic*, Volume 27, Issue 5 (October 1998), page 489: "Development of first-order logic independently of Frege, anticipating prenex and Skolem normal forms"

[2] Among these authors is Stolyar, p. 166. Hamilton considers both to be calculi but divides them into an informal calculus and a formal calculus.

## References

- A. G. Hamilton 1978, *Logic for Mathematicians*, Cambridge University Press, Cambridge UK ISBN 0-521-21838-1
- Abram Aronovic Stolyar 1970, *Introduction to Elementary Mathematical Logic*, Dover Publications, Inc. NY. ISBN 0-486-645614
- George F Luger, *Artificial Intelligence*, Pearson Education, ISBN 978-81-317-2327-2
- Hazewinkel, Michiel, ed. (2001), "Predicate calculus" (http://www.encyclopediaofmath.org/index.php?title=p/p074360), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

# Prenex normal form

A formula of the predicate calculus is in **prenex**[1] **normal form**[2] if it is written as a string of quantifiers (referred to as the **prefix**) followed by a quantifier-free part (referred to as the **matrix**).

Every formula in classical logic is equivalent to a formula in prenex normal form. For example, if $\phi(y)$, $\psi(z)$, and $\rho(x)$ are quantifier-free formulas with the free variables shown then

$$\forall x \exists y \forall z (\phi(y) \vee (\psi(z) \rightarrow \rho(x)))$$

is in prenex normal form with matrix $\phi(y) \vee (\psi(z) \rightarrow \rho(x))$, while

$$\forall x ((\exists y \phi(y)) \vee ((\exists z \psi(z)) \rightarrow \rho(x)))$$

is logically equivalent but not in prenex normal form.

## Conversion to prenex form

Every first-order formula is logically equivalent (in classical logic) to some formula in prenex normal form. There are several conversion rules that can be recursively applied to convert a formula to prenex normal form. The rules depend on which logical connectives appear in the formula.

### Conjunction and disjunction

The rules for conjunction and disjunction say that

$(\forall x \phi) \wedge \psi$ is equivalent to $\forall x (\phi \wedge \psi)$,

$(\forall x \phi) \vee \psi$ is equivalent to $\forall x (\phi \vee \psi)$;

and

$(\exists x \phi) \wedge \psi$ is equivalent to $\exists x (\phi \wedge \psi)$,

$(\exists x \phi) \vee \psi$ is equivalent to $\exists x (\phi \vee \psi)$.

The equivalences are valid when *x* does not appear as a free variable of ψ; if *x* does appear free in ψ, it must be replaced with another free variable.

For example, in the language of rings,

$(\exists x (x^2 = 1)) \wedge (0 = y)$ is equivalent to $\exists x (x^2 = 1 \wedge 0 = y)$,

but

$(\exists x (x^2 = 1)) \wedge (0 = x)$ is not equivalent to $\exists x (x^2 = 1 \wedge 0 = x)$

because the formula on the left is true in any ring when the free variable *x* is equal to 0, while the formula on the right has no free variables and is false in any nontrivial ring.

### Negation

The rules for negation say that

$\neg\exists x\phi$ is equivalent to $\forall x\neg\phi$

and

$\neg\forall x\phi$ is equivalent to $\exists x\neg\phi$ .

### Implication

There are four rules for implication: two that remove quantifiers from the antecedent and two that remove quantifiers from the consequent. These rules can be derived by rewriting the implication $\phi\rightarrow\psi$ as $\neg\phi\lor\psi$ and applying the rules for disjunction above. As with the rules for disjunction, these rules require that the variable quantified in one subformula does not appear free in the other subformula.

The rules for removing quantifiers from the antecedent are:

$(\forall x\phi)\rightarrow\psi$ is equivalent to $\exists x(\phi\rightarrow\psi)$,

$(\exists x\phi)\rightarrow\psi$ is equivalent to $\forall x(\phi\rightarrow\psi)$.

The rules for removing quantifiers from the consequent are:

$\phi\rightarrow(\exists x\psi)$ is equivalent to $\exists x(\phi\rightarrow\psi)$,

$\phi\rightarrow(\forall x\psi)$ is equivalent to $\forall x(\phi\rightarrow\psi)$.

### Example

Suppose that $\phi$ , $\psi$ , and $\rho$ are quantifier-free formulas and no two of these formulas share any free variable. Consider the formula

$(\phi\lor\exists x\psi)\rightarrow\forall z\rho$.

By recursively applying the rules starting at the innermost subformulas, the following sequence of logically equivalent formulas can be obtained:

$(\phi\lor\exists x\psi)\rightarrow\forall z\rho$.

$(\exists x(\phi\lor\psi))\rightarrow\forall z\rho$,

$\neg(\exists x(\phi\lor\psi))\lor\forall z\rho$,

$(\forall x\neg(\phi\lor\psi))\lor\forall z\rho$,

$\forall x(\neg(\phi\lor\psi)\lor\forall z\rho)$,

$\forall x((\phi\lor\psi)\rightarrow\forall z\rho)$,

$\forall x(\forall z((\phi\lor\psi)\rightarrow\rho))$,

$\forall x\forall z((\phi\lor\psi)\rightarrow\rho)$.

This is not the only prenex form equivalent to the original formula. For example, by dealing with the consequent before the antecedent in the example above, the prenex form

$\forall z\forall x((\phi\lor\psi)\rightarrow\rho)$

can be obtained:

$\forall z((\phi\lor\exists x\psi)\rightarrow\rho)$

$\forall z((\exists x(\phi\lor\psi))\rightarrow\rho)$,

$\forall z(\forall x((\phi\lor\psi)\rightarrow\rho))$,

$\forall z\forall x((\phi\lor\psi)\rightarrow\rho)$.

### Intuitionistic logic

The rules for converting a formula to prenex form make heavy use of classical logic. In intuitionistic logic, it is not true that every formula is logically equivalent to a prenex formula. The negation connective is one obstacle, but not the only one. The implication operator is also treated differently in intuitionistic logic than classical logic; in intuitionistic logic, it is not definable using disjunction and negation.

The BHK interpretation illustrates why some formulas have no intuitionistically-equivalent prenex form. In this interpretation, a proof of

$$(\exists x\phi) \to \exists y\psi \qquad (1)$$

is a function which, given a concrete *x* and a proof of φ(*x*), produces a concrete *y* and a proof of ψ(*y*). In this case it is allowable for the value of *y* to be computed from the given value of *x*. A proof of

$$\exists y(\exists x\phi \to \psi), \qquad (2)$$

on the other hand, produces a single concrete value of *y* and a function that converts any proof of $\exists x\phi$ into a proof of ψ(*y*). If each *x* satisfying φ can be used to construct a *y* satisfying ψ but no such *y* can be constructed without knowledge of such an *x* then formula (1) will not be equivalent to formula (2).
The rules for converting a formula to prenex form that do *fail* in intuitionistic logic are:

(1) $\forall x(\phi \vee \psi)$implies $(\forall x\phi) \vee \psi$ ,
(2) $\forall x(\phi \vee \psi)$implies $\phi \vee (\forall x\psi)$,
(3) $(\forall x\phi) \to \psi$ implies $\exists x(\phi \to \psi)$,
(4) $\phi \to (\exists x\psi)$implies $\exists x(\phi \to \psi)$,
(5) $\neg\forall x\phi$ implies $\exists x\neg\phi$ ,

(*x* does not appear as a free variable of $\psi$ in (1) and (3); *x* does not appear as a free variable of $\phi$ in (2) and (4)).

## Use of prenex form

Some proof calculi will only deal with a theory whose formulae are written in prenex normal form. The concept is essential for developing the arithmetical hierarchy and the analytical hierarchy.

Gödel's proof of his completeness theorem for first-order logic presupposes that all formulae have been recast in prenex normal form.

## Notes

[1] The term 'prenex' comes from the Latin *praenexus* "tied or bound up in front", past participle of *praenectere* (http://cs.nyu.edu/pipermail/fom/2007-November/012328.html).
[2] http://toolserver.org/%7Edispenser/cgi-bin/dab_solver.py?page=Prenex_normal_form&editintro=Template:Disambiguation_needed/editintro&client=Template:Dn

## References

• Hinman, P. (2005), *Fundamentals of Mathematical Logic*, A K Peters, ISBN 978-1-56881-262-5

# Horn clause

In mathematical logic and logic programming, a **Horn clause** is a logical formula of a particular rule-like form which gives it good properties for use in logic programming, formal specification, and model theory. Horn clauses are named for the logician Alfred Horn, who first pointed out their significance in 1951, in the article "On sentences which are true of direct unions of algebras", Journal of Symbolic Logic, 16, 14–21.

## Definition

Horn clause is a clause (a disjunction of literals) with at most one positive, i.e. unnegated, literal.

Conversely, a disjunction of literals with at most one negated literal is called a **dual-Horn clause**.

A Horn clause with exactly one positive literal is a **definite clause**; a definite clause with no negative literals is sometimes called a **fact**; and a Horn clause without a positive literal is sometimes called a **goal clause**. These three kinds of Horn clauses are illustrated in the following propositional example:

| | **Disjunction form** | **Implication form** | **Read intuitively as** |
|---|---|---|---|
| **Definite clause** | $\neg p \lor \neg q \lor ... \lor \neg t \lor u$ | $u \leftarrow p \land q \land ... \land t$ | assume that *u* holds if *p* and *q* and ... and *t* all hold |
| **Fact** | $u$ | $u$ | assume that *u* holds |
| **Goal clause** | $\neg p \lor \neg q \lor ... \lor \neg t$ | $false \leftarrow p \land q \land ... \land t$ | show that *p* and *q* and ... and *t* all hold [1] |

In the non-propositional case, all variables in a clause are implicitly universally quantified with scope the entire clause. Thus, for example:

$\neg\ human(X) \lor mortal(X)$

stands for:

$\forall X(\ \neg\ human(X) \lor mortal(X)\ )$

which is logically equivalent to:

$\forall X\ (\ human(X) \rightarrow mortal(X)\ )$

Horn clauses play a basic role in constructive logic and computational logic. They are important in automated theorem proving by first-order resolution, because the resolvent of two Horn clauses is itself a Horn clause, and the resolvent of a goal clause and a definite clause is a goal clause. These properties of Horn clauses can lead to greater efficiencies in proving a theorem (represented as the negation of a goal clause).

Propositional Horn clauses are also of interest in computational complexity, where the problem of finding truth value assignments to make a conjunction of propositional Horn clauses true is a P-complete problem (in fact solvable in linear time), sometimes called HORNSAT. (The unrestricted Boolean satisfiability problem is an NP-complete problem however.) Satisfiability of first-order Horn clauses is undecidable.

## Logic programming

Horn clauses are also the basis of logic programming, where it is common to write definite clauses in the form of an implication:

$(p \wedge q \wedge ... \wedge t) \rightarrow u$

In fact, the resolution of a goal clause with a definite clause to produce a new goal clause is the basis of the SLD resolution inference rule, used to implement logic programming and the programming language Prolog.

In logic programming a definite clause behaves as a goal-reduction procedure. For example, the Horn clause written above behaves as the procedure:

to show *u*, show *p* and show *q* and ... and show *t*.

To emphasize this reverse use of the clause, it is often written in the reverse form:

$u \leftarrow (p \wedge q \wedge ... \wedge t)$

In Prolog this is written as:

```
u :- p, q, ..., t.
```

In logic programming and datalog, computation and query evaluation are performed by representing the negation of a problem to be solved as a goal clause. For example, the problem of solving the existentially quantified conjunction of positive literals:

$\exists X\ (p \wedge q \wedge ... \wedge t)$

is represented by negating the problem (denying that it has a solution), and representing it in the logically equivalent form of a goal clause:

$\forall X\ (\mathit{false} \leftarrow p \wedge q \wedge ... \wedge t)$

In Prolog this is written as:

```
:- p, q, ..., t.
```

Solving the problem amounts to deriving a contradiction, which is represented by the empty clause (or "false"). The solution of the problem is a substitution of terms for the variables in the goal clause, which can be extracted from the proof of contradiction. Used in this way, goal clauses are similar to conjunctive queries in relational databases, and Horn clause logic is equivalent in computational power to a universal Turing machine.

The Prolog notation is actually ambiguous, and the term “goal clause” is sometimes also used ambiguously. The variables in a goal clause can be read as universally or existentially quantified, and deriving “false” can be interpreted either as deriving a contradiction or as deriving a successful solution of the problem to be solved.

Van Emden and Kowalski (1976) investigated the model theoretic properties of Horn clauses in the context of logic programming, showing that every set of definite clauses **D** has a unique minimal model **M**. An atomic formula **A** is logically implied by **D** if and only if **A** is true in **M**. It follows that a problem **P** represented by an existentially quantified conjunction of positive literals is logically implied by **D** if and only if **P** is true in **M**. The minimal model semantics of Horn clauses is the basis for the stable model semantics of logic programs.

## Notes

[1] Like in resolution theorem proving, intuitive meanings "show φ" and "assume ¬φ" are synonymous (indirect proof); they both correspond to the same formula, viz. ¬φ. This way, a mechanical proving tool needs to maintain only one set of formulas (assumptions), rather than two sets (assumptions and (sub)goals).

## Bibliography

- Alfred Horn (1951), "On sentences which are true of direct unions of algebras", *Journal of Symbolic Logic*, 16, 14–21.
- Dowling, W. and Gallier, J. (1984), "Linear-time algorithms for testing the satisfiability of propositional Horn formulae". *Journal of Logic Programming*, **3**, 267–284.
- M. van Emden and R. Kowalski [1976] *The semantics of predicate logic as a programming language* (http://www.doc.ic.ac.uk/~rak/papers/kowalski-van_emden.pdf). Journal of ACM, Vol. 23, 733–742.

# Article Sources and Contributors

**Conjunctive normal form** *Source*: http://en.wikipedia.org/w/index.php?oldid=599147646 *Contributors*: A3 nm, Alejandrocaro35, Andkore, Aquatopia, Arvindn, Ary29, Aydos, B4hand, BenBaker, Blaisorblade, Bobogoobo, Bryan Derksen, CBM, Cherlin, Cognominally, CyborgTosser, Danlev, Dardasavta, Dresdnhope, ESkog, Eclecticos, Erik Garrison, Fresheneesz, Frostyandy2k, Giftlite, Graham87, Gregbard, Gubbubu, Hairy Dude, Hermel, Hobsonlane, Hofmic, IM Serious, IsleLaMotte, Jacobolus, Jamelan, Jamesx12345, Jleedev, Jochen Burghardt, Jon Awbrey, Jpvinall, Jrtayloriv, Ldo, LilHelpa, Linas, Macrakis, Masnevets, MementoVivere, Mhss, Michael Hardy, Mike Segal, Mikhail Dvorkin, Mikolasj, Mx2323, Myasuda, Niteowlneils, OSDevLabs, Obradovic Goran, Oleg Alexandrov, Policron, Poromenos, PrimeHunter, Robert Merkel, Ross Fraser, Rotemliss, Salix alba, Simeon, Snikeris, Tesi1700, Thesilverbail, Tijfo098, Tizio, Toby Bartels, Wzwz, 79 anonymous edits

**Disjunctive normal form** *Source*: http://en.wikipedia.org/w/index.php?oldid=597060144 *Contributors*: A3r0, Ajm81, Alejandrocaro35, Altenmann, B4hand, BD2412, Batenka, Ben Spinozoan, BenBaker, Brona, Bryan Derksen, CBM, CyborgTosser, DavidCary, Doulos Christos, EmilJ, Fresheneesz, Graham87, Gregbard, Groovenstein, Gryllida, Haham hanuka, Hans Adler, Igor Yalovecky, Intervallic, Jamelan, Jiri 1984, Jon Awbrey, Kundu, Linas, Linket, Macrakis, MementoVivere, Mhss, Policron, Sarrazip, Simeon, Tizio, Tobias Bergemann, Toby Bartels, Tvdm, Wzwz, ZeroOne, 27 anonymous edits

**Truth table** *Source*: http://en.wikipedia.org/w/index.php?oldid=603302231 *Contributors*: 65.68.87.xxx, Abd, Achal Singh, Addihockey10, Aeroknight, AgadaUrbanit, Aitias, Akuindo, Alansohn, Ancheta Wis, Andres, Annina.kari, Antandrus, Antonielly, ArnoldReinhold, Aston Martian, AugPi, Avaya1, BD2412, Banno, Bdesham, Beetstra, Bergin, Bgwhite, Billymac00, Bluemoose, Bookandcoffee, Bryan Derksen, Bubba73, C933103, CBM, CRGreathouse, CWenger, CZ Top, Can't sleep, clown will eat me, Canthusus, Cburnett, Ceklock, Charles Matthews, Cheese Sandwich, Clon89, Cmglee, Conversion script, Creidieki, Crystalllized, Cybercobra, Danlev, Dcoetzee, Delirium, DesertSteve, Dicklyon, Djayjp, Dr Smith, Dysprosia, Flowerpotman, Fox89, Francvs, Fresheneesz, Gaiacarra, Gary, Gershwinrb, Ghettoblaster, Giftlite, Graham87, Gregbard, Gschadow, Hans Adler, Hephaestos, Heron, Holly golightly, InverseHypercube, Ixfd64, JDP90, JVz, Jason Quinn, JimWae, Jimfbleak, Jk2q3jrklse, Johnbrownsbody, Jon Awbrey, Jonsafari, Joyous!, Julian Mendez, Justin Johnson, K ganju, Kbdank71, Kiril Simeonovski, Kitty Davis, KnightRider, KnowledgeOfSelf, Krawi, Kyle Barbour, La marts boys, Larry Sanger, Lee, Lethe, Letranova, Liftarn, Lights, Logan, Lst27, LutzL, Markhurd, Maustrauser, Mav, Mdd, Mets501, Mhss, Michael Hardy, Millermk, Mindmatrix, Nakke, Noosphere, Nortexoid, Olaf, Oleg Alexandrov, On This Continent, Oreo Priest, Pakaran, Parikshit Narkhede, Patrick, Paul August, Policron, Pooryorick, Quad4rax, Quintote, Qwfp, R27smith200245, Racconish, RedWolf, Rich Farmbrough, Rofthorax, Santiago Saint James, Schnits, Sevillana, Sir48, Slazenger, SpuriousQ, Starylon, TBloemink, Tangotango, Tarquin, The Rhymesmith, The Tetrast, Tijfo098, Trovatore, Uthren, Vadmium, Vegetator, Vilerage, Wavelength, Wbm1058, Webmaestro, WikiPuppies, WimdeValk, Wolfmanx122, Wshun, Wstorr, XTerminator2000, Xxpor, 422 anonymous edits

**De Morgan's laws** *Source*: http://en.wikipedia.org/w/index.php?oldid=603004018 *Contributors*: 16@r, Action ben, Adambiswanger1, Alejandrocaro35, Alexius08, Alphax, Art LaPella, AugPi, B1atv, Benjgil, Bkkbrad, Bluemathman, Bongomatic, Boongie, Boredzo, Btyner, Capricorn42, Cdiggins, Chalst, Charles Matthews, Chewings72, Choster, ChromaNebula, Cldoyle, Coolv, Cori.schlegel, Cybercobra, DVdm, DanielZM, DannyAsher, Darktemplar, David Shay, David815, Dcoetzee, DesertSteve, Dorfl, Drae, Drake Redcrest, Dysprosia, ESkog, Ebertek, EmilJ, Epbr123, Epicgenius, Eric-Wester, Fratrep, Fredrik, Futurebird, G S Palmer, General Jazza, Giftlite, Gobonobo, Graham87, Gregbard, Guppyfinsoup, Hadal, Hairy Dude, Hannes Eder, Hans Adler, Helgus, Idonei, Ihcoyc, Into The Fray, JForget, JRSP, JascalX, Javawizard, Jeronimo, Jon Awbrey, Jqavins, Jsorr, Kanags, Kratos 84, Larry V, Linas, Linj, Linket, Loadmaster, Marozols, Mbonet, Melcombe, Mhss, Michael Hardy, Michael Slone, MikeLynch, Mindmatrix, Miserlou, Mitch feaster, Mudlock, Najoj, Nitku, Nutster, Oleg Alexandrov, Petrejo, Policron, R'n'B, RBarryYoung, RDBury, Rapsar, Rodrigoq, Rror, Saric, Scrutchfield, SirPeebles, Smimram, Smoseson, Smylers, Squelle, Starblue, Stdazi, Stpasha, Subtractive, Sylvier11, TWiStErRob, TakuyaMurata, Tarquin, The Anome, The wub, Tide rolls, Ttennebkram, Ttwo, Waleed.598, Wavelength, Widr, Xiaodai, 172 anonymous edits

**Boolean algebra (structure)** *Source*: http://en.wikipedia.org/w/index.php?oldid=602312537 *Contributors*: 2D, ABCD, Aguitarheroperson, Alai, Albmont, Alhutch, Ancheta Wis, Andreasabel, Andrewpmk, Anonymous Dissident, Archelon, Arthur Rubin, Avantman42, AxelBoldt, Baccala@freesoft.org, Barnaby dawson, Bryan Derksen, Bsod2, Btwied, Bunny Angel13, CBM, CBM2, Cacycle, Camembert, Cburnett, Celestianpower, Chalst, Charles Matthews, Charvest, ChrisGualtieri, Ciacchi, Clarepawling, Colin Rowat, Condem, Constructive editor, Conversion script, Cronholm144, Cullinane, Cybercobra, D.Lazard, Dai mingjie, David Eppstein, Dcoetzee, Delusion23, DesertSteve, Dysprosia, ERcheck, Ed Poor, Eduardoporcher, Eequor, Elias, Ellywa, Elwikipedista, Enoksrd, Escher26, Fibonacci, Fredrik, Freeze S, Freiddie, FunnyYetTasty, Fwehrung, GOD, GTBacchus, Gauss, Giftlite, Gonzalcg, Graham87, Gregbard, Hairy Dude, Hans Adler, Helder.wiki, Heron, Honx, Ht686rg90, Hugo Herbelin, Igny, Ilmari Karonen, Ilya, Imc, Incnis Mrsi, Irmy, Isaac Rabinovitch, Ivan Bajlo, Izehar, Jarble, Jeronimo, Jiri 1984, Jitse Niesen, JoanneB, Jochen Burghardt, Joebeone, Johnleemk, Jojit fb, Jon Awbrey, JorgeGG, Josh Cherry, Julian Mendez, Justin Johnson, KHamsun, KSmrq, KamasamaK, Kephir, Kirachinmoku, Klparrot, Kompik, KrakatoaKatie, Kzollman, Lambiam, Lethe, LilHelpa, Linas, MER-C, MSGJ, Macrakis, Magioladitis, Magister Mathematicae, Mani1, MarSch, Markus Krötzsch, Masashi49, Mav, Maximus Rex, Meco, Michael Hardy, Mosesklein, Msh210, Nagytibi, Nortexoid, OkPerson, Omicron18, Paul August, Pce3@ij.net, Perry Bebbington, Pit, Plclark, Pleasantville, Plugwash, Poccil, Policron, Pvemburg, Qwertyus, Robinh, Romanm, Ruakh, Sagaciousuk, Salix alba, Samtheboy, Samuel, Sandman, SaxTeacher, Scythe33, Shellreef, SixWingedSeraph, Slipstream, Sommacal alfonso, Sopoforic, Staecker, StuRat, Tagremover, Talkstosocks, Tarquin, Taw, TedDunning, Tellyaddict, The Tetrast, Thecheesykid, Tijfo098, Timwi, Tobias Bergemann, Toby Bartels, Trovatore, Ukexpat, Uncle G, Vaughan Pratt, Visor, Voodoo, Wiki alf, WimdeValk, Woohookitty, Wrs1864, XJaM, Yamamoto Ichiro, Zero sharp, Zundark, Јованвб, Александър, דניאל ב., 218 anonymous edits

**Algebraic normal form** *Source*: http://en.wikipedia.org/w/index.php?oldid=599805149 *Contributors*: CBM, Charles Matthews, CyborgTosser, GBL, Hans Adler, JackSchmidt, Jiri 1984, Jon Awbrey, Linas, Macrakis, Mairi, Michael Hardy, Ner102, Olathe, Oleg Alexandrov, Salgueiro, 5 anonymous edits

**Universal quantification** *Source*: http://en.wikipedia.org/w/index.php?oldid=603704587 *Contributors*: 16@r, Aeron Daly, Andres, Anonymous Dissident, Arthur Rubin, Avicennasis, Benzi455, Bkell, Bobo192, CBM, Cinephile2, Clconway, Computer97, DBooth, Dcoetzee, DoubleBlue, Dpakoha, Dysprosia, E235, Eigenlambda, Faus, Felliax08, Giftlite, Grafen, Gregbard, Hairy Dude, Hede2000, Henrygb, Ihope127, JMyrleFuller, Jangirke, Jimmaths, Joriki, Jpceayene, Jshadias, Julian Mendez, Jusdafax, Kenj0418, Khukri, Kiefer.Wolfowitz, Lambiam, Laurentius, Lemon-s, Logoprofeta, Maxdamantus, Mets501, Mhss, Michael Hardy, Mild Bill Hiccup, Mindmatrix, Miracle Pen, Neocapitalist, PGSONIC, Pgk, Poor Yorick, RedDotTrail, Robertinventor, Ruud Koot, Ryguasu, Salgueiro, Salix alba, Spug, The Thing That Should Not Be, Thorwald, Toby Bartels, Tomisti, Urhixidur, WikiPerfector, Wsvlqc, Zero sharp, 67 anonymous edits

**Existential quantification** *Source*: http://en.wikipedia.org/w/index.php?oldid=595307630 *Contributors*: Andre Engels, Andres, Anonymous Dissident, Arthur Rubin, Ashley Y, Bradgib, Brews ohare, CBM, Caltrop, Charles Matthews, Chinju, Clconway, Cybercobra, Dcoetzee, DePiep, Dpakoha, Dysprosia, E235, Faus, Giftlite, GigaMario5, Gregbard, Hede2000, Ish ishwar, Jameshfisher, Jengelh, Jimmaths, Jonash36, Joriki, Jsnx, Julian Mendez, Ketil, Kine, Mets501, Mhss, Michael Hardy, Miracle Pen, Naleh, Neelix, Nortexoid, Oleg Alexandrov, Pdelong, Poor Yorick, Quentar, Richiar, Robertinventor, Salix alba, Sardoodledom, Skomorokh, Stephan Schulz, TakuyaMurata, Tarquin, Tentontunic, The Cunctator, TheSmuel, Tim Q. Wells, Titodutta, Toby Bartels, Tomisti, Twin Bird, Urhixidur, Waninoco, Yoda1522, 53 anonymous edits

**Predicate logic** *Source*: http://en.wikipedia.org/w/index.php?oldid=596514694 *Contributors*: Andres, Anonymous Dissident, Bjankuloski06en, BoltonSM3, Brad7777, Byelf2007, CBM, Chester Markel, Cybercobra, DesolateReality, Dessources, Djk3, ESSch, George100, Giftlite, Gregbard, Hypergeek14, Jayde239, JohnBlackburne, Jpbowen, Kumioko (renamed), Leonard G., Logichulk, Materialscientist, MathMartin, Mayur, Mhss, Michael Hardy, Mindmatrix, Nakon, Naudefj, Policron, Satellizer, Sharkface217, Soler97, Stassa, Taxa, Thekohser, TimClicks, Toby Bartels, Tomajohnson, Tomisti, Vanished user g454XxNpUVWvxzlr, Virago250, Wvbailey, Xiaq, Xnn, 39 anonymous edits

**Prenex normal form** *Source*: http://en.wikipedia.org/w/index.php?oldid=587988322 *Contributors*: AugPi, BD2412, CBM, CRGreathouse, Charles Matthews, Coreyoconnor, Dysprosia, Epiglottisz, Esoth, Gandalf61, Greenrd, IsleLaMotte, Jakob.scholbach, Jayme, Joriki, Linas, Lockeownzj00, MattGiuca, Mhss, Michael Hardy, Omnipaedista, Pfortuny, Reetep, The Anome, Toobaz, 26 anonymous edits

**Horn clause** *Source*: http://en.wikipedia.org/w/index.php?oldid=588541676 *Contributors*: A Softer Answer, Altenmann, Angela, BD2412, BLNarayanan, CRGreathouse, Cek, Chalst, Charles Matthews, ChrisGualtieri, Compulogger, Dcoetzee, Doradus, Edward, Elwikipedista, EmilJ, Fieldmethods, Ft1, Gareth Griffith-Jones, Gregbard, Gubbubu, Hannes Eder, Hans Adler, Inquam, Jackee1234, Jacobolus, Jamelan, Jarble, Jochen Burghardt, Karada, Karlheg, Kevin.cohen, Knverma, Kwi, Ldo, Linas, Logperson, Luqui, MIRROR, MattGiuca, Mhss, Michael Hardy, Michael Zeising, NYKevin, Nunoplopes, Osnetwork, Paullakso, RatnimSnave, Rintdusts, Ritchy, Rsimmonds01, Tajko, Tawker, Template namespace initialisation script, Thadk, Theone256, Tijfo098, Tizio, Tom harrison, Vkuncak, Woohookitty, Xmlizer, 36 anonymous edits

# Image Sources, Licenses and Contributors

# License