

Logical

Contents

Articles

Articles with a wide scope and introductions	1
Algebra of sets	1
Boolean algebra (structure)	5
Boolean algebra	17
Field of sets	32
List of logic symbols	36
Logical connective	40
Necessity and sufficiency	46
Propositional calculus	49
Truth function	66
General Information and Vocabulary	74
2-valued morphism	74
Bitwise operation	75
Boolean data type	82
Boolean expression	85
Boolean satisfiability problem	86
Boolean-valued model	96
Booleo	100
Chaff algorithm	101
Correlation immunity	102
Davis–Putnam algorithm	102
DPLL algorithm	103
Formula game	106
Join (sigma algebra)	106
Logic alphabet	107
Logic redundancy	110
Logical matrix	111
Logical value	113
Modal algebra	114
Petrick's method	115
Product term	116
Propositional formula	117

Stone duality	144
Stone functor	148
True quantified Boolean formula	148
Truth value	152
Vector logic	153
Visualization	158
Binary decision diagram	158
Implication graph	162
Karnaugh map	163
Propositional directed acyclic graph	169
Quine–McCluskey algorithm	171
Reed-Muller expansion	174
Truth table	175
Venn diagram	183
Logical Connectives and functions	191
Ampheck	191
Balanced boolean function	193
Bent function	193
Boolean algebras canonically defined	196
Boolean function	208
Boolean-valued function	209
Conditioned disjunction	210
Converse implication	211
Converse nonimplication	212
Evasive Boolean function	215
Exclusive or	216
False (logic)	223
Functional completeness	224
If and only if	227
Inclusion (Boolean algebra)	231
Indicative conditional	232
Indicator function	233
Logical NOR	236
Logical biconditional	238
Logical conjunction	243
Logical disjunction	248

Logical equality	253
Logical implication	255
Logical negation	258
Lupanov representation	262
Majority function	262
Material conditional	263
Material equivalence	267
Material nonimplication	271
Modal operator	272
Negation	273
Parity function	277
Peirce arrow	278
Sheffer stroke	280
Sole sufficient operator	283
Statement (logic)	286
Strict conditional	287
Symmetric Boolean function	289
Symmetric difference	289
Tautology (logic)	292
Zhegalkin polynomial	297
 Syntax	 299
Algebraic normal form	299
Boolean conjunctive query	301
Canonical form (Boolean algebra)	302
Conjunctive normal form	310
Disjunctive normal form	314
Formal system	315
 Normal Forms	 318
Blake canonical form	318
Canonical normal form	318
Herbrand normal form	326
Herbrandization	327
Horn clause	328
Negation normal form	330
Prenex normal form	331
Skolem normal form	334

Theorems and specific laws	337
Absorption law	337
Boole's expansion theorem	338
Boolean prime ideal theorem	339
Compactness theorem	342
Consensus theorem	345
De Morgan's laws	346
Duality (order theory)	352
Laws of classical logic	353
Peirce's law	368
Poretsky's law of forms	370
Stone's representation theorem for Boolean algebras	370
Philosophy	372
Boole's syllogistic	372
Entitative graph	373
Existential graph	374
Implicant	378
Laws of Form	379
Logical graph	393
Examples of Boolean algebras	394
Boolean domain	394
Boolean ring	395
Goodman–Nguyen–van Fraassen algebra	397
Interior algebra	399
Lindenbaum–Tarski algebra	405
Relation algebra	406
Residuated Boolean algebra	412
Robbins algebra	414
Sigma-algebra	415
Topological Boolean algebra	418
Two-element Boolean algebra	418
Extensions and generalizations	421
Complete Boolean algebra	421
Derivative algebra (abstract algebra)	423
First-order logic	424

Free Boolean algebra	444
Heyting algebra	446
Monadic Boolean algebra	456
Skew lattice	457
Technical applications	464
And-inverter graph	464
Boolean analysis	465
Boolean operations in computer-aided design	468
Circuit minimization	469
Espresso heuristic logic minimizer	470
Logic gate	473
People	480
Augustus De Morgan	480
Charles Sanders Peirce	491
George Boole	520
Ivan Ivanovich Zhegalkin	528
John Venn	529
Marshall Harvey Stone	533
William Stanley Jevons	535
Propositional Calculus	545
Clause (logic)	545
Contradiction	546
Deductive closure	550
Formation rule	551
Frege system	552
Frege's propositional calculus	554
Implicational propositional calculus	563
Intermediate logic	566
List of logic systems	568
Literal (mathematical logic)	576
Logical consequence	577
Nicod's axiom	580
Open sentence	580
Predicate (mathematical logic)	582
Principle of distributivity	583

Proof by contrapositive	583
Proposition	584
Propositional proof system	587
Propositional variable	590
Rule of inference	590
Rule of replacement	595
Second-order propositional logic	596
Substitution (logic)	596
Syncategorematic term	598
System L	599
Unsatisfiable core	601
Zeroth-order logic	601
Propositional Fallacies	603
Affirming a disjunct	603
Affirming the consequent	604
Denying the antecedent	605
Rules of Inference	607
List of rules of inference	607
Absorption (logic)	612
Admissible rule	614
Associative property	623
Biconditional elimination	629
Biconditional introduction	630
Commutative property	632
Commutativity of conjunction	639
Conjunction introduction	640
Constructive dilemma	641
Contraposition (traditional logic)	643
Destructive dilemma	645
Disjunction elimination	648
Disjunction introduction	649
Disjunctive syllogism	651
Distributive property	653
Double negative elimination	657
Existential generalization	660
Existential instantiation	661

Exportation (logic)	662
Hypothetical syllogism	664
List of valid argument forms	665
Material implication (rule of inference)	668
Modus non excipiens	670
Modus ponendo tollens	671
Modus ponens	672
Modus tollens	675
Negation as failure	679
Resolution (logic)	681
SLD resolution	685
Simplification	688
Structural rule	690
Tautology (rule of inference)	691
Transposition (logic)	693
Universal generalization	697
Universal instantiation	699

Theorems in Propositional Logic 701

Case analysis	701
Consequentia mirabilis	702
Contraposition	703
Double negation	707
Frege's theorem	708
Idempotency of entailment	708
Law of excluded middle	709
Law of identity	716
Law of noncontradiction	718
Monotonicity of entailment	723
Principle of explosion	724
Proof by contradiction	727
Reductio ad absurdum	729

References

Article Sources and Contributors	731
Image Sources, Licenses and Contributors	744

Article Licenses

License

750

Articles with a wide scope and introductions

Algebra of sets

The **algebra of sets** defines the properties and laws of sets, the set-theoretic operations of union, intersection, and complementation and the relations of set equality and set inclusion. It also provides systematic procedures for evaluating expressions, and performing calculations, involving these operations and relations.

Any set of sets closed under the set-theoretic operations forms a Boolean algebra with the join operator being *union*, the meet operator being *intersection*, and the complement operator being *set complement*.

Fundamentals

The algebra of sets is the set-theoretic analogue of the algebra of numbers. Just as arithmetic addition and multiplication are associative and commutative, so are set union and intersection; just as the arithmetic relation "less than or equal" is reflexive, antisymmetric and transitive, so is the set relation of "subset".

It is the algebra of the set-theoretic operations of union, intersection and complementation, and the relations of equality and inclusion. For a basic introduction to sets see the article on sets, for a fuller account see naive set theory, and for a full rigorous axiomatic treatment see axiomatic set theory.

The fundamental laws of set algebra

The binary operations of set union (\cup) and intersection (\cap) satisfy many identities. Several of these identities or "laws" have well established names.

Commutative laws:

- $A \cup B = B \cup A$
- $A \cap B = B \cap A$

Associative laws:

- $(A \cup B) \cup C = A \cup (B \cup C)$
- $(A \cap B) \cap C = A \cap (B \cap C)$

Distributive laws:

- $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
- $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

The analogy between unions and intersections of sets, and addition and multiplication of numbers, is quite striking. Like addition and multiplication, the operations of union and intersection are commutative and associative, and intersection *distributes* over unions. However, unlike addition and multiplication, union also *distributes* over intersection.

Two additional pairs of laws involve the special sets called the empty set \emptyset and the universal set U ; together with the complement operator. The empty set has no members, and the universal set has all possible members (in a particular context).

Identity laws:

- $A \cup \emptyset = A$
- $A \cap U = A$

Complement laws:

- $A \cup A^C = U$
- $A \cap A^C = \emptyset$

The identity laws (together with the commutative laws) say that, just like 0 and 1 for addition and multiplication, \emptyset and U are the identity elements for union and intersection, respectively.

Unlike addition and multiplication, union and intersection do not have inverse elements. However the complement laws give the fundamental properties of the somewhat inverse-like unary operation of set complementation.

The preceding five pairs of laws—the commutative, associative, distributive, identity and complement laws—encompass all of set algebra, in the sense that every valid proposition in the algebra of sets can be derived from them.

Note that if the complement laws are weakened to the rule $(A^C)^C = A$, then this is exactly the algebra of propositional linear logicWikipedia:Please clarify.

The principle of duality

Each of the identities stated above is one of a pair of identities such that each can be transformed into the other by interchanging \cup and \cap , and also \emptyset and U .

These are examples of an extremely important and powerful property of set algebra, namely, the **principle of duality** for sets, which asserts that for any true statement about sets, the **dual** statement obtained by interchanging unions and intersections, interchanging U and \emptyset and reversing inclusions is also true. A statement is said to be **self-dual** if it is equal to its own dual.

Some additional laws for unions and intersections

The following proposition states six more important laws of set algebra, involving unions and intersections.

PROPOSITION 3: For any subsets A and B of a universal set U , the following identities hold:

idempotent laws:

- $A \cup A = A$
- $A \cap A = A$

domination laws:

- $A \cup U = U$
- $A \cap \emptyset = \emptyset$

absorption laws:

- $A \cup (A \cap B) = A$
- $A \cap (A \cup B) = A$

As noted above each of the laws stated in proposition 3, can be derived from the five fundamental pairs of laws stated in proposition 1 and proposition 2. As an illustration, a proof is given below for the idempotent law for union.

Proof:

$$\begin{aligned}
 A \cup A &= (A \cup A) \cap U && \text{by the identity law of intersection} \\
 &= (A \cup A) \cap (A \cup A^C) && \text{by the complement law for union} \\
 &= A \cup (A \cap A^C) && \text{by the distributive law of union over intersection} \\
 &= A \cup \emptyset && \text{by the complement law for intersection} \\
 &= A && \text{by the identity law for union}
 \end{aligned}$$

The following proof illustrates that the dual of the above proof is the proof of the dual of the idempotent law for union, namely the idempotent law for intersection.

Proof:

$$\begin{aligned}
 A \cap A &= (A \cap A) \cup \emptyset && \text{by the identity law for union} \\
 &= (A \cap A) \cup (A \cap A^C) && \text{by the complement law for intersection} \\
 &= A \cap (A \cup A^C) && \text{by the distributive law of intersection over union} \\
 &= A \cap U && \text{by the complement law for union} \\
 &= A && \text{by the identity law for intersection}
 \end{aligned}$$

Intersection can be expressed in terms of union and set difference :

$$A \cap B = ((A \cup B) \setminus (A \setminus B)) \setminus (B \setminus A)$$

Some additional laws for complements

The following proposition states five more important laws of set algebra, involving complements.

PROPOSITION 4: Let A and B be subsets of a universe \mathbf{U} , then:

De Morgan's laws:

- $(A \cup B)^C = A^C \cap B^C$
- $(A \cap B)^C = A^C \cup B^C$

double complement or Involution law:

- $(A^C)^C = A$

complement laws for the universal set and the empty set:

- $\emptyset^C = \mathbf{U}$
- $\mathbf{U}^C = \emptyset$

Notice that the double complement law is self-dual.

The next proposition, which is also self-dual, says that the complement of a set is the only set that satisfies the complement laws. In other words, complementation is characterized by the complement laws.

PROPOSITION 5: Let A and B be subsets of a universe \mathbf{U} , then:

uniqueness of complements:

- If $A \cup B = \mathbf{U}$, and $A \cap B = \emptyset$, then $B = A^C$

The algebra of inclusion

The following proposition says that inclusion is a partial order.

PROPOSITION 6: If A , B and C are sets then the following hold:

reflexivity:

- $A \subseteq A$

antisymmetry:

- $A \subseteq B$ and $B \subseteq A$ if and only if $A = B$

transitivity:

- If $A \subseteq B$ and $B \subseteq C$, then $A \subseteq C$

The following proposition says that for any set S , the power set of S , ordered by inclusion, is a bounded lattice, and hence together with the distributive and complement laws above, show that it is a Boolean algebra.

PROPOSITION 7: If A , B and C are subsets of a set S then the following hold:

existence of a least element and a greatest element:

- $\emptyset \subseteq A \subseteq S$

existence of joins:

- $A \subseteq A \cup B$

- If $A \subseteq C$ and $B \subseteq C$, then $A \cup B \subseteq C$

existence of meets:

- $A \cap B \subseteq A$

- If $C \subseteq A$ and $C \subseteq B$, then $C \subseteq A \cap B$

The following proposition says that the statement $A \subseteq B$ is equivalent to various other statements involving unions, intersections and complements.

PROPOSITION 8: For any two sets A and B , the following are equivalent:

- $A \subseteq B$
- $A \cap B = A$
- $A \cup B = B$
- $A \setminus B = \emptyset$
- $B^C \subseteq A^C$

The above proposition shows that the relation of set inclusion can be characterized by either of the operations of set union or set intersection, which means that the notion of set inclusion is axiomatically superfluous.

The algebra of relative complements

The following proposition lists several identities concerning relative complements or set-theoretic difference.

PROPOSITION 9: For any universe U and subsets A , B , and C of U , the following identities hold:

- $C \setminus (A \cap B) = (C \setminus A) \cup (C \setminus B)$
- $C \setminus (A \cup B) = (C \setminus A) \cap (C \setminus B)$
- $C \setminus (B \setminus A) = (A \cap C) \cup (C \setminus B)$
- $(B \setminus A) \cap C = (B \cap C) \setminus A = B \cap (C \setminus A)$
- $(B \setminus A) \cup C = (B \cup C) \setminus (A \setminus C)$
- $A \setminus A = \emptyset$
- $\emptyset \setminus A = \emptyset$
- $A \setminus \emptyset = A$

- $B \setminus A = A^C \cap B$
- $(B \setminus A)^C = A \cup B^C$
- $U \setminus A = A^C$
- $A \setminus U = \emptyset$

References

- Stoll, Robert R.; *Set Theory and Logic*, Mineola, N.Y.: Dover Publications (1979) ISBN 0-486-63829-4. "The Algebra of Sets", pp 16—23 [1]
- Courant, Richard, Herbert Robbins, Ian Stewart, *What is mathematics?: An Elementary Approach to Ideas and Methods*, Oxford University Press US, 1996. ISBN 978-0-19-510519-3. "SUPPLEMENT TO CHAPTER II THE ALGEBRA OF SETS" [2]

External links

- Operations on Sets at ProvenMath [3]

References

- [1] <http://books.google.com/books?id=3-nrPB7BQKMC&pg=PA16#v=onepage&q&f=false>
[2] http://books.google.com/books?id=UfdossHPlkgC&pg=PA17-IA8&dq=%22algebra+of+sets%22&hl=en&ei=k8-RTdXoF4K2tgcM-p1v&sa=X&oi=book_result&ct=result&resnum=3&ved=0CDYQ6AEwAg#v=onepage&q=%22algebra%20of%20sets%22&f=false
[3] <http://www.apronus.com/provenmath/btheorems.htm>

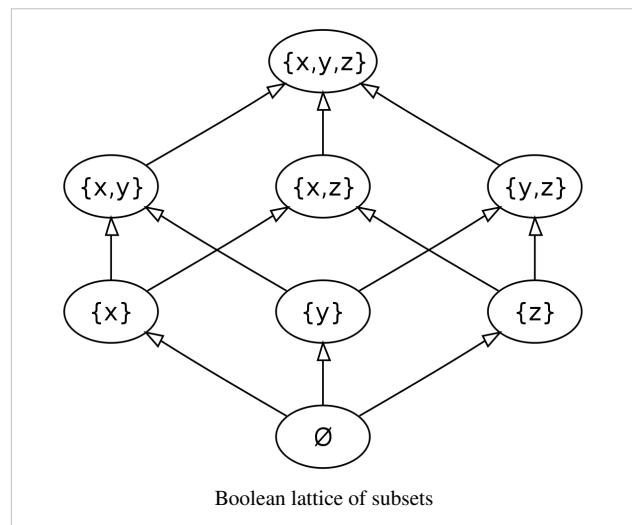
Boolean algebra (structure)

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 \wedge , and ring addition to exclusive disjunction or symmetric difference (not disjunction \vee).

History

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 \wedge (called "meet" or "and"), \vee (called "join" or "or"), a unary operation \neg (called "complement" or "not") and two elements 0 and 1 (called "bottom" and "top", or "least" and "greatest" element, also denoted by the symbols \perp and \top , respectively), such that for all elements a, b and c of A , the following axioms hold:^[1]

$$\begin{array}{lll}
 a \vee (b \vee c) = (a \vee b) \vee c & a \wedge (b \wedge c) = (a \wedge b) \wedge c & \text{associativity} \\
 a \vee b = b \vee a & a \wedge b = b \wedge a & \text{commutativity} \\
 a \vee (a \wedge b) = a & a \wedge (a \vee b) = a & \text{absorption} \\
 a \vee 0 = a & a \wedge 1 = a & \text{identity} \\
 a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) & a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) & \text{distributivity} \\
 a \vee \neg a = 1 & a \wedge \neg a = 0 & \text{complements}
 \end{array}$$

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) that

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

The relation \leq 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 \leq .

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 \vee with \wedge 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**.^[citation needed]

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:

$\begin{array}{ c c c } \hline \top & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 0 & 1 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline \top & 0 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline a & 0 & 1 \\ \hline \neg a & 1 & 0 \\ \hline \end{array}$
--	--	--

- It has applications in logic, interpreting 0 as *false*, 1 as *true*, \wedge as *and*, \vee as *or*, and \neg 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 $\vee := \cup$ (union) and $\wedge := \cap$ (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:

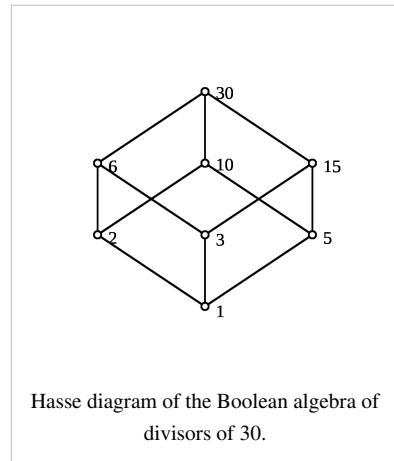
\square	0	a	b	1
0	0	0	0	0
a	0	a	0	a
b	0	0	b	b
1	0	a	b	1

\square	0	a	b	1
0	0	a	b	1
a	a	a	1	1
b	b	1	b	1

x	0	a	b	1
$\neg x$	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 $a \leq b$ 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 $\text{lcm}(a,b)/\text{gcd}(a,b)$. 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.
- 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 $\vee := \cup$ (union) and $\wedge := \cap$ (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, \wedge, \vee) 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 $x \vee y$ 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 $a \wedge \neg a = 0 \in I$ and then $a \in I$ or $\neg a \in I$ 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 $\neg a \notin I$, then $I \cup \{a\}$ or $I \cup \{\neg a\}$ 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	
UId₁ If $x \sqcap o = x$ for all x, then $o = 0$ Proof: If $x \vee o = x$, then 0 $= 0 \vee o$ by assumption $= o \vee 0$ by Cmm ₁ $= o$ by Idn ₁	UId₂ [dual] If $x \wedge i = x$ for all x , then $i = 1$
Idm₁ $x \sqcap x = x$ Proof: $x \vee x$ $= (x \vee x) \wedge 1$ by Idn ₂ $= (x \vee x) \wedge (x \vee \neg x)$ by Cpl ₁ $= x \vee (x \wedge \neg x)$ by Dst ₁ $= x \vee 0$ by Cpl ₂ $= x$ by Idn ₁	Idm₂ [dual] $x \wedge x = x$

<p>Bnd₁ $x \sqcup 1 = 1$</p> <p>Proof: $x \vee 1$</p> $ \begin{aligned} &= (x \vee 1) \wedge 1 && \text{by Idn}_2 \\ &= 1 \wedge (x \vee 1) && \text{by Cmm}_2 \\ &= (x \vee \neg x) \wedge (x \vee 1) && \text{by Cpl}_1 \\ &= x \vee (\neg x \wedge 1) && \text{by Dst}_1 \\ &= x \vee \neg x && \text{by Idn}_2 \\ &= 1 && \text{by Cpl}_1 \end{aligned} $	<p>Bnd₂ [dual] $x \wedge 0 = 0$</p>
<p>Abs₁ $x \sqcup (x \sqcup y) = x$</p> <p>Proof: $x \vee (x \wedge y)$</p> $ \begin{aligned} &= (x \wedge 1) \vee (x \wedge y) && \text{by Idn}_2 \\ &= x \wedge (1 \vee y) && \text{by Dst}_2 \\ &= x \wedge (y \vee 1) && \text{by Cmm}_1 \\ &= x \wedge 1 && \text{by Bnd}_1 \\ &= x && \text{by Idn}_2 \end{aligned} $	<p>Abs₂ [dual] $x \wedge (x \vee y) = x$</p>
<p>UNg If $x \sqcup x_n = 1$ and $x \sqcup x_n = 0$, then $x_n = \neg x$</p> <p>Proof: If $x \vee x_n = 1$ and $x \wedge x_n = 0$, then</p> $ \begin{aligned} &x_n \\ &= x_n \wedge 1 && \text{by Idn}_2 \\ &= 1 \wedge x_n && \text{by Cmm}_2 \\ &= (x \vee \neg x) \wedge x_n && \text{by Cpl}_1 \\ &= x_n \wedge (x \vee \neg x) && \text{by Cmm}_2 \\ &= (x_n \wedge x) \vee (x_n \wedge \neg x) && \text{by Dst}_2 \\ &= (x \wedge x_n) \vee (\neg x \wedge x_n) && \text{by Cmm}_2 \\ &= 0 \vee (\neg x \wedge x_n) && \text{by assumption} \\ &= (x \wedge \neg x) \vee (\neg x \wedge x_n) && \text{by Cpl}_2 \\ &= (\neg x \wedge x) \vee (\neg x \wedge x_n) && \text{by Cmm}_2 \\ &= \neg x \wedge (x \vee x_n) && \text{by Dst}_2 \\ &= \neg x \wedge 1 && \text{by assumption} \\ &= \neg x && \text{by Idn}_2 \end{aligned} $	
<p>DNg $\neg \neg x = x$</p> <p>Proof: $\neg x \vee x = x \vee \neg x = 1$ by Cmm₁, Cpl₁</p> <p>and $\neg x \wedge x = x \wedge \neg x = 0$ by Cmm₂, Cpl₂</p> <p>hence $x = \neg \neg x$ by UNg</p>	

<p>A₁ $x \sqcup (\neg x \sqcup y) = 1$</p> <p>Proof: $x \vee (\neg x \vee y)$</p> $ \begin{aligned} &= (x \vee (\neg x \vee y)) \wedge 1 && \text{by } \mathbf{Idn}_2 \\ &= 1 \wedge (x \vee (\neg x \vee y)) && \text{by } \mathbf{Cmm}_2 \\ &= (x \vee \neg x) \wedge (x \vee (\neg x \vee y)) && \text{by } \mathbf{Cpl}_1 \\ &= x \vee (\neg x \wedge (\neg x \vee y)) && \text{by } \mathbf{Dst}_1 \\ &= x \vee \neg x && \text{by } \mathbf{Xlb} \\ &= 1 && \text{by } \mathbf{Cpl}_1 \end{aligned} $	<p>A₂ [dual] $x \wedge (\neg x \wedge y) = 0$</p>
<p>B₁ $(x \sqcup y) \sqcup (\neg x \sqcup \neg y) = 1$</p> <p>Proof: $(x \vee y) \vee (\neg x \wedge \neg y)$</p> $ \begin{aligned} &= ((x \vee y) \vee \neg x) \wedge ((x \vee y) \vee \neg y) && \text{by } \mathbf{Dst}_1 \\ &= (\neg x \vee (x \vee y)) \wedge (\neg y \vee (y \vee x)) && \text{by } \mathbf{Cmm}_1 \\ &= (\neg x \vee (\neg x \vee y)) \wedge (\neg y \vee (\neg y \vee x)) && \text{by } \mathbf{DNg} \\ &= 1 \wedge 1 && \text{by } \mathbf{A}_1 \\ &= 1 && \text{by } \mathbf{Idn}_2 \end{aligned} $	<p>B₁ [dual] $(x \wedge y) \wedge (\neg x \vee \neg y) = 0$</p>
<p>C₁ $(x \sqcup y) \sqcup (\neg x \sqcup \neg y) = 0$</p> <p>Proof: $(x \vee y) \wedge (\neg x \wedge \neg y)$</p> $ \begin{aligned} &= (\neg x \wedge \neg y) \wedge (x \vee y) && \text{by } \mathbf{Cmm}_2 \\ &= ((\neg x \wedge \neg y) \wedge x) \vee ((\neg x \wedge \neg y) \wedge y) && \text{by } \mathbf{Dst}_2 \\ &= (x \wedge (\neg x \wedge \neg y)) \vee (y \wedge (\neg y \wedge \neg x)) && \text{by } \mathbf{Cmm}_2 \\ &= 0 \vee 0 && \text{by } \mathbf{A}_2 \\ &= 0 && \text{by } \mathbf{Idn}_1 \end{aligned} $	<p>C₂ [dual] $(x \wedge y) \vee (\neg x \vee \neg y) = 1$</p>
<p>DMg₁ $\neg(x \sqcup y) = \neg x \sqcup \neg y$</p> <p>Proof: by B₁, C₁, and UNg</p>	<p>DMg₂ [dual] $\neg(x \wedge y) = \neg x \vee \neg y$</p>
<p>D₁ $(x \sqcup (y \sqcup z)) \sqcup \neg x = 1$</p> <p>Proof: $(x \vee (y \vee z)) \vee \neg x$</p> $ \begin{aligned} &= \neg x \vee (x \vee (y \vee z)) && \text{by } \mathbf{Cmm}_1 \\ &= \neg x \vee (\neg \neg x \vee (y \vee z)) && \text{by } \mathbf{DNg} \\ &= 1 && \text{by } \mathbf{A}_1 \end{aligned} $	<p>D₂ [dual] $(x \wedge (y \wedge z)) \wedge \neg x = 0$</p>

<p>E₁ $y \sqcap (x \sqcap (y \sqcap z)) = y$</p> <p>Proof: $y \wedge (x \vee (y \vee z))$</p> $ \begin{aligned} &= (y \wedge x) \vee (y \wedge (y \vee z)) \text{ by } \mathbf{Dst}_2 \\ &= (y \wedge x) \vee y \text{ by } \mathbf{Abs}_2 \\ &= y \vee (y \wedge x) \text{ by } \mathbf{Cmm}_1 \\ &= y \text{ by } \mathbf{Abs}_1 \end{aligned} $	<p>E₂ [dual] $y \vee (x \wedge (y \wedge z)) = y$</p>
<p>F₁ $(x \sqcap (y \sqcap z)) \sqcap \neg y = 1$</p> <p>Proof: $(x \vee (y \vee z)) \vee \neg y$</p> $ \begin{aligned} &= \neg y \vee (x \vee (y \vee z)) \text{ by } \mathbf{Cmm}_1 \\ &= (\neg y \vee (x \vee (y \vee z))) \wedge 1 \text{ by } \mathbf{Idn}_2 \\ &= 1 \wedge (\neg y \vee (x \vee (y \vee z))) \text{ by } \mathbf{Cmm}_2 \\ &= (y \vee \neg y) \wedge (\neg y \vee (x \vee (y \vee z))) \text{ by } \mathbf{Cpl}_1 \\ &= (\neg y \vee y) \wedge (\neg y \vee (x \vee (y \vee z))) \text{ by } \mathbf{Cmm}_1 \\ &= \neg y \vee (y \wedge (x \vee (y \vee z))) \text{ by } \mathbf{Dst}_1 \\ &= \neg y \vee y \text{ by } \mathbf{E}_1 \\ &= y \vee \neg y \text{ by } \mathbf{Cmm}_1 \\ &= 1 \text{ by } \mathbf{Cpl}_1 \end{aligned} $	<p>F₂ [dual] $(x \wedge (y \wedge z)) \wedge \neg y = 0$</p>
<p>G₁ $(x \sqcap (y \sqcap z)) \sqcap \neg z = 1$</p> <p>Proof: $(x \vee (y \vee z)) \vee \neg z$</p> $ \begin{aligned} &= (x \vee (z \vee y)) \vee \neg z \text{ by } \mathbf{Cmm}_1 \\ &= 1 \text{ by } \mathbf{F}_1 \end{aligned} $	<p>G₂ [dual] $(x \wedge (y \wedge z)) \wedge \neg z = 0$</p>
<p>H₁ $\neg((x \sqcap y) \sqcap z) \sqcap x = 0$</p> <p>Proof: $\neg((x \vee y) \vee z) \wedge x$</p> $ \begin{aligned} &= (\neg(x \vee y) \wedge \neg z) \wedge x \text{ by } \mathbf{DMg}_1 \\ &= ((\neg x \wedge \neg y) \wedge \neg z) \wedge x \text{ by } \mathbf{DMg}_1 \\ &= x \wedge ((\neg x \wedge \neg y) \wedge \neg z) \text{ by } \mathbf{Cmm}_2 \\ &= (x \wedge ((\neg x \wedge \neg y) \wedge \neg z)) \vee 0 \text{ by } \mathbf{Idn}_1 \\ &= 0 \vee (x \wedge ((\neg x \wedge \neg y) \wedge \neg z)) \text{ by } \mathbf{Cmm}_1 \\ &= (x \wedge \neg x) \vee (x \wedge ((\neg x \wedge \neg y) \wedge \neg z)) \text{ by } \mathbf{Cpl}_1 \\ &= x \wedge (\neg x \vee ((\neg x \wedge \neg y) \wedge \neg z)) \text{ by } \mathbf{Dst}_2 \\ &= x \wedge (\neg x \vee (\neg z \wedge (\neg x \wedge \neg y))) \text{ by } \mathbf{Cmm}_2 \\ &= x \wedge \neg x \text{ by } \mathbf{E}_2 \\ &= 0 \text{ by } \mathbf{Cpl}_2 \end{aligned} $	<p>H₂ [dual] $\neg((x \wedge y) \wedge z) \vee x = 1$</p>

<p>I₁ $\neg((x \sqcup y) \sqcup z) \sqcup y = 0$</p> <p>Proof: $\neg((x \vee y) \vee z) \wedge y$</p> $= \neg((y \vee x) \vee z) \wedge y \quad \text{by Cmm}_1$ $= 0 \quad \text{by H}_1$	<p>I₂ [dual] $\neg((x \wedge y) \wedge z) \vee y = 1$</p>
<p>J₁ $\neg((x \sqcup y) \sqcup z) \sqcup z = 0$</p> <p>Proof: $\neg((x \vee y) \vee z) \wedge z$</p> $= (\neg(x \vee y) \wedge \neg z) \wedge z \quad \text{by DMg}_1$ $= z \wedge (\neg(x \vee y) \wedge \neg z) \quad \text{by Cmm}_2$ $= z \wedge (\neg z \wedge \neg(x \vee y)) \quad \text{by Cmm}_2$ $= 0 \quad \text{by A}_2$	<p>J₂ [dual] $\neg((x \wedge y) \wedge z) \vee z = 1$</p>
<p>K₁ $(x \sqcup (y \sqcup z)) \sqcup \neg((x \sqcup y) \sqcup z) = 1$</p> <p>Proof: $(x \vee (y \vee z)) \vee \neg((x \vee y) \vee z)$</p> $= (x \vee (y \vee z)) \vee (\neg(x \vee y) \wedge \neg z) \quad \text{by DMg}_1$ $= (x \vee (y \vee z)) \vee ((\neg x \wedge \neg y) \wedge \neg z) \quad \text{by DMg}_1$ $= ((x \vee (y \vee z)) \vee (\neg x \wedge \neg y)) \wedge ((x \vee (y \vee z)) \vee \neg z) \quad \text{by Dst}_1$ $= (((x \vee (y \vee z)) \vee \neg x) \wedge ((x \vee (y \vee z)) \vee \neg y)) \wedge ((x \vee (y \vee z)) \vee \neg z) \quad \text{by Dst}_1$ $= (1 \wedge 1) \wedge 1 \quad \text{by D}_1, \mathbf{F}_1, \mathbf{G}_1$ $= 1 \quad \text{by Idn}_2$	<p>K₂ [dual] $(x \wedge (y \wedge z)) \wedge \neg((x \wedge y) \wedge z) = 0$</p>
<p>L₁ $(x \sqcup (y \sqcup z)) \sqcup \neg((x \sqcup y) \sqcup z) = 0$</p> <p>Proof: $(x \vee (y \vee z)) \wedge \neg((x \vee y) \vee z)$</p> $= \neg((x \vee y) \vee z) \wedge (x \vee (y \vee z)) \quad \text{by Cmm}_2$ $= (\neg((x \vee y) \vee z) \wedge x) \vee (\neg((x \vee y) \vee z) \wedge (y \vee z)) \quad \text{by Dst}_2$ $= (\neg((x \vee y) \vee z) \wedge x) \vee ((\neg((x \vee y) \vee z) \wedge y) \vee (\neg((x \vee y) \vee z) \wedge z)) \quad \text{by Dst}_2$ $= (0 \vee 0) \vee 0 \quad \text{by H}_1, \mathbf{I}_1, \mathbf{J}_1$ $= 0 \quad \text{by Idn}_1$	<p>L₂ [dual] $(x \wedge (y \wedge z)) \vee \neg((x \wedge y) \wedge z) = 1$</p>
<p>Ass₁ $x \sqcup (y \sqcup z) = (x \sqcup y) \sqcup z$</p> <p>Proof: by K₁, L₁, UNg, DN_g</p>	<p>Ass₂ [dual] $x \wedge (y \wedge z) = (x \wedge y) \wedge z$</p>

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 \wedge , \vee , \neg , 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

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=JIXSlpmISv4C&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

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 of 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*.

Boolean algebra

In mathematics and mathematical logic, **Boolean algebra** is the subarea of algebra in which the values of the variables are the truth values *true* and *false*, usually denoted 1 and 0 respectively. Instead of elementary algebra where the values of the variables are numbers, and the main operations are addition and multiplication, the main operations of Boolean algebra are the conjunction *and*, denoted \wedge , the disjunction *or*, denoted \vee , and the negation *not*, denoted \neg .

Boolean algebra was introduced in 1854 by George Boole in his book *An Investigation of the Laws of Thought*. According to Huntington the term "Boolean algebra" was first suggested by Sheffer in 1913.^[1]

Boolean algebra has been fundamental in the development of computer science and digital logic. It is also used in set theory and statistics.

History

Boole's algebra predicated the modern developments in abstract algebra and mathematical logic; it is however seen as connected to the origins of both fields. In an abstract setting, Boolean algebra was perfected in the late 19th century by Jevons, Schröder, Huntington, and others until it reached the modern conception of an (abstract) mathematical structure. For example, the empirical observation that one can manipulate expressions in the algebra of sets by translating them into expressions in Boole's algebra is explained in modern terms by saying that the algebra of sets is a Boolean algebra (note the indefinite article). In fact, M. H. Stone proved in 1936 that every Boolean algebra is isomorphic to a field of sets.

In the 1930s, while studying switching circuits, Claude Shannon observed that one could also apply the rules of Boole's algebra in this setting, and he introduced **switching algebra** as a way to analyze and design circuits by algebraic means in terms of logic gates. Shannon already had at his disposal the abstract mathematical apparatus, thus he cast his switching algebra as the two-element Boolean algebra. In circuit engineering settings today, there is little need to consider other Boolean algebras, thus "switching algebra" and "Boolean algebra" are often used interchangeably.^[2] Efficient implementation of Boolean functions is a fundamental problem in the design of combinatorial logic circuits. Modern electronic design automation tools for VLSI circuits often rely on an efficient representation of Boolean functions known as (reduced ordered) binary decision diagrams (BDD) for logic synthesis and formal verification.

Logic sentences that can be expressed in classical propositional calculus have an equivalent expression in Boolean algebra. Thus, **Boolean logic** is sometimes used to denote propositional calculus performed in this way. Boolean algebra is not sufficient to capture logic formulas using quantifiers, like those from first order logic. Although the development of mathematical logic did not follow Boole's program, the connection between his algebra and logic was later put on firm ground in the setting of algebraic logic, which also studies the algebraic systems of many other logics. The problem of determining whether the variables of a given Boolean (propositional) formula can be assigned in such a way as to make the formula evaluate to true is called the Boolean satisfiability problem (SAT), and is of importance to theoretical computer science, being the first problem shown to be NP-complete. The closely related model of computation known as a Boolean circuit relates time complexity (of an algorithm) to circuit complexity.

Values

Whereas in elementary algebra expressions denote mainly numbers, in Boolean algebra they denote the truth values *false* and *true*. These values are represented with the bits (or binary digits) being 0 and 1. They do not behave like the integers 0 and 1, for which $1 + 1 = 2$, but may be identified with the elements of the two-element field GF(2), for which $1 + 1 = 0$ with + serving as the Boolean operation XOR.

Boolean algebra also deals with functions which have their values in the set {0, 1}. A sequence of bits is a commonly used such function. Another common example is the subsets of a set E : to a subset F of E is associated the indicator function that takes the value 1 on F and 0 outside F .

As with elementary algebra, the purely equational part of the theory may be developed without considering explicit values for the variables.^[3]

Operations

Note: the truth table consisting of all 16 binary functions is located at [Boolean_algebras_canonically_defined#Truth_tables](#). There is also more information about these functions in [Truth_table](#).

Basic operations

The basic operations of Boolean algebra are the following ones:

- And (conjunction), denoted $x \wedge y$ (sometimes x AND y or K_{xy}), satisfies $x \wedge y = 1$ if $x = y = 1$ and $x \wedge y = 0$ otherwise.
- Or (disjunction), denoted $x \vee y$ (sometimes x OR y or A_{xy}), satisfies $x \vee y = 0$ if $x = y = 0$ and $x \vee y = 1$ otherwise.
- Not (negation), denoted $\neg x$ (sometimes NOT x , Nx or $!x$), satisfies $\neg x = 0$ if $x = 1$ and $\neg x = 1$ if $x = 0$.

If the truth values 0 and 1 are interpreted as integers, these operation may be expressed with the ordinary operations of the arithmetic:

$$x \wedge y = xy,$$

$$x \vee y = x + y - xy,$$

$$\neg x = 1 - x.$$

Alternatively the values of $x \wedge y$, $x \vee y$, and $\neg x$ can be expressed by tabulating their values with truth tables as follows.

x	y	$x \wedge y$	$x \vee y$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	1

+ Figure 1. Truth tables

One may consider that only the negation and one of the two other operations are basic, because of the following identities that allow to define the conjunction in terms of the negation and the disjunction, and vice versa:

$$x \wedge y = \neg(\neg x \vee \neg y)$$

$$x \vee y = \neg(\neg x \wedge \neg y)$$

Derived operations

We have so far seen three Boolean operations. We referred to these as basic, meaning that they can be taken as a basis for other Boolean operations that can be built up from them by **composition**, the manner in which operations are combined or compounded. Here are some examples of operations composed from the basic operations.

$$\begin{aligned}x \rightarrow y &= (\neg x \vee y) \\x \oplus y &= (x \vee y) \wedge \neg(x \wedge y) \\x \equiv y &= \neg(x \oplus y)\end{aligned}$$

These definitions give rise to the following truth tables giving the values of these operations for all four possible inputs.

x	y	$x \rightarrow y$	$x \oplus y$	$x \equiv y$
0	0	1	0	1
1	0	0	1	0
0	1	1	1	0
1	1	1	0	1

The first operation, $x \rightarrow y$, or Cxy , is called **material implication**. If x is true then the value of $x \rightarrow y$ is taken to be that of y . But if x is false then we ignore the value of y ; however we must return *some* truth value and there are only two choices, so we choose the value that entails less, namely *true*. (Relevance logic addresses this by viewing an implication with a false premise as something other than either true or false.)

The second operation, $x \oplus y$, or Jxy , is called **exclusive or** to distinguish it from disjunction as the inclusive kind. It excludes the possibility of both x and y . Defined in terms of arithmetic it is addition mod 2 where $1 + 1 = 0$.

The third operation, the complement of exclusive or, is **equivalence** or Boolean equality: $x \equiv y$, or Exy , is true just when x and y have the same value. Hence $x \oplus y$ as its complement can be understood as $x \neq y$, being true just when x and y are different. Its counterpart in arithmetic mod 2 is $x + y + 1$.

Laws

A **law** of Boolean algebra is an identity such as $x \vee(y \vee z) = (x \vee y) \vee z$ between two Boolean terms, where a **Boolean term** is defined as an expression built up from variables and the constants 0 and 1 using the operations \wedge , \vee , and \neg . The concept can be extended to terms involving other Boolean operations such as \oplus , \rightarrow , and \equiv , but such extensions are unnecessary for the purposes to which the laws are put. Such purposes include the definition of a Boolean algebra as any model of the Boolean laws, and as a means for deriving new laws from old as in the derivation of $x \vee(y \wedge z) = x \vee(z \wedge y)$ from $y \wedge z = z \wedge y$ as treated in the section on axiomatization.

Monotone laws

Boolean algebra satisfies many of the same laws as ordinary algebra when we match up \vee with addition and \wedge with multiplication. In particular the following laws are common to both kinds of algebra:

(Associativity of \vee)	$x \vee (y \vee z) = (x \vee y) \vee z$
(Associativity of \wedge)	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
(Commutativity of \vee)	$x \vee y = y \vee x$
(Commutativity of \wedge)	$x \wedge y = y \wedge x$
(Distributivity of \wedge over \vee)	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
(Identity for \vee)	$x \vee 0 = x$
(Identity for \wedge)	$x \wedge 1 = x$
(Annihilator for \wedge)	$x \wedge 0 = 0$

Boolean algebra however obeys some additional laws, in particular the following:

(Idempotence of \vee)	$x \vee x = x$
(Idempotence of \wedge)	$x \wedge x = x$
(Absorption 1)	$x \wedge (x \vee y) = x$
(Absorption 2)	$x \vee (x \wedge y) = x$
(Distributivity of \vee over \wedge)	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
(Annihilator for \vee)	$x \vee 1 = 1$

A consequence of the first of these laws is $1 \vee 1 = 1$, which is false in ordinary algebra, where $1+1 = 2$. Taking $x = 2$ in the second law shows that it is not an ordinary algebra law either, since $2 \times 2 = 4$. The remaining four laws can be falsified in ordinary algebra by taking all variables to be 1, for example in Absorption Law 1 the left hand side is $1(1+1) = 2$ while the right hand side is 1, and so on.

All of the laws treated so far have been for conjunction and disjunction. These operations have the property that changing either argument either leaves the output unchanged or the output changes in the same way as the input. Equivalently, changing any variable from 0 to 1 never results in the output changing from 1 to 0. Operations with this property are said to be **monotone**. Thus the axioms so far have all been for monotonic Boolean logic. Nonmonotonicity enters via complement \neg as follows.

Nonmonotone laws

The complement operation is defined by the following two laws.

(Complementation 1)	$x \wedge \neg x = 0$
(Complementation 2)	$x \vee \neg x = 1$

All properties of negation including the laws below follow from the above two laws alone.

In both ordinary and Boolean algebra, negation works by exchanging pairs of elements, whence in both algebras it satisfies the double negation law (also called involution law)

$$(\text{Double negation}) \quad \neg \neg x = x.$$

But whereas ordinary algebra satisfies the two laws

$$\begin{aligned} (\neg x)(\neg y) &= xy \\ (\neg x) + (\neg y) &= -(x + y), \end{aligned}$$

Boolean algebra satisfies De Morgan's laws,

$$(\text{De Morgan 1}) \quad (\neg x) \wedge (\neg y) = \neg(x \vee y)$$

$$(\text{De Morgan 2}) \quad (\neg x) \vee (\neg y) = \neg(x \wedge y).$$

Completeness

At this point we can now claim to have defined Boolean algebra, in the sense that the laws we have listed up to now entail the rest of the subject. The laws *Complementation* 1 and 2, together with the monotone laws, suffice for this purpose and can therefore be taken as one possible *complete* set of laws or axiomatization of Boolean algebra. Every law of Boolean algebra follows logically from these axioms. Furthermore Boolean algebras can then be defined as the models of these axioms as treated in the section thereon.

To clarify, writing down further laws of Boolean algebra cannot give rise to any new consequences of these axioms, nor can it rule out any model of them. Had we stopped listing laws too soon, there would have been Boolean laws that did not follow from those on our list, and moreover there would have been models of the listed laws that were not Boolean algebras.

This axiomatization is by no means the only one, or even necessarily the most natural given that we did not pay attention to whether some of the axioms followed from others but simply chose to stop when we noticed we had enough laws, treated further in the section on axiomatizations. Or the intermediate notion of axiom can be sidestepped altogether by defining a Boolean law directly as any **tautology**, understood as an equation that holds for all values of its variables over 0 and 1. All these definitions of Boolean algebra can be shown to be equivalent.

Boolean algebra has the interesting property that $x = y$ can be proved from any non-tautology. This is because the substitution instance of any non-tautology obtained by instantiating its variables with constants 0 or 1 so as to witness its non-tautologyhood reduces by equational reasoning to $0 = 1$. For example the non-tautologyhood of $x \wedge y = x$ is witnessed by $x = 1$ and $y = 0$ and so taking this as an axiom would allow us to infer $1 \wedge 0 = 1$ as a substitution instance of the axiom and hence $0 = 1$. We can then show $x = y$ by the reasoning $x = x \wedge 1 = x \wedge 0 = 0 = 1 = y \vee 1 = y \vee 0 = y$.

Duality principle

There is nothing magical about the choice of symbols for the values of Boolean algebra. We could rename 0 and 1 to say α and β , and as long as we did so consistently throughout it would still be Boolean algebra, albeit with some obvious cosmetic differences.

But suppose we rename 0 and 1 to 1 and 0 respectively. Then it would still be Boolean algebra, and moreover operating on the same values. However it would not be identical to our original Boolean algebra because now we find \vee behaving the way \wedge used to do and vice versa. So there are still some cosmetic differences to show that we've been fiddling with the notation, despite the fact that we're still using 0s and 1s.

But if in addition to interchanging the names of the values we also interchange the names of the two binary operations, *now* there is no trace of what we have done. The end product is completely indistinguishable from what we started with. We might notice that the columns for $x \wedge y$ and $x \vee y$ in the truth tables had changed places, but that switch is immaterial.

When values and operations can be paired up in a way that leaves everything important unchanged when all pairs are switched simultaneously, we call the members of each pair **dual** to each other. Thus 0 and 1 are dual, and \wedge and \vee are dual. The **Duality Principle**, also called De Morgan duality, asserts that Boolean algebra is unchanged when all dual pairs are interchanged.

One change we did not need to make as part of this interchange was to complement. We say that complement is a **self-dual** operation. The identity or do-nothing operation x (copy the input to the output) is also self-dual. A more complicated example of a self-dual operation is $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$. It can be shown that self-dual operations must

take an odd number of arguments; thus there can be no self-dual binary operation.

The principle of duality can be explained from a group theory perspective by fact that there are exactly four functions that are one-to-one mappings (automorphisms) of the set of Boolean polynomials back to itself: the identity function, the complement function, the dual function and the contradual function (complemented dual). These four functions form a group under function composition, isomorphic to the Klein four-group, acting on the set of Boolean polynomials.

Diagrammatic representations

Venn diagrams

A Venn diagram^[4] is a representation of a Boolean operation using shaded overlapping regions. There is one region for each variable, all circular in the examples here. The interior and exterior of region x corresponds respectively to the values 1 (true) and 0 (false) for variable x . The shading indicates the value of the operation for each combination of regions, with dark denoting 1 and light 0 (some authors use the opposite convention).

The three Venn diagrams in the figure below represent respectively conjunction $x \wedge y$, disjunction $x \vee y$, and complement $\neg x$.

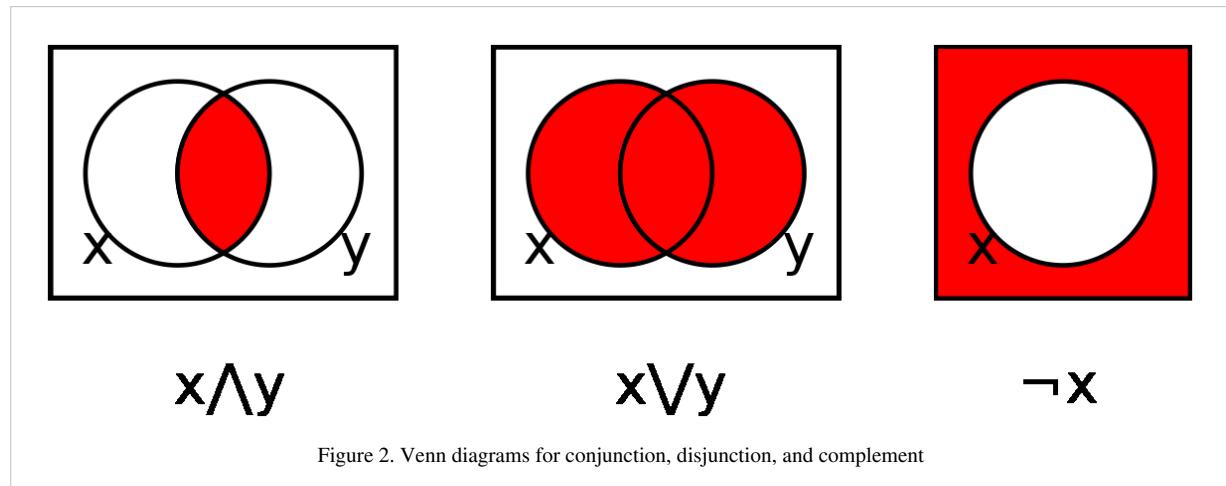


Figure 2. Venn diagrams for conjunction, disjunction, and complement

For conjunction, the region inside both circles is shaded to indicate that $x \wedge y$ is 1 when both variables are 1. The other regions are left unshaded to indicate that $x \wedge y$ is 0 for the other three combinations.

The second diagram represents disjunction $x \vee y$ by shading those regions that lie inside either or both circles. The third diagram represents complement $\neg x$ by shading the region *not* inside the circle.

While we have not shown the Venn diagrams for the constants 0 and 1, they are trivial, being respectively a white box and a dark box, neither one containing a circle. However we could put a circle for x in those boxes, in which case each would denote a function of one argument, x , which returns the same value independently of x , called a constant function. As far as their outputs are concerned, constants and constant functions are indistinguishable; the difference is that a constant takes no arguments, called a *zeroary* or *nullary* operation, while a constant function takes one argument, which it ignores, and is a *unary* operation.

Venn diagrams are helpful in visualizing laws. The commutativity laws for \wedge and \vee can be seen from the symmetry of the diagrams: a binary operation that was not commutative would not have a symmetric diagram because interchanging x and y would have the effect of reflecting the diagram horizontally and any failure of commutativity would then appear as a failure of symmetry.

Idempotence of \wedge and \vee can be visualized by sliding the two circles together and noting that the shaded area then becomes the whole circle, for both \wedge and \vee .

To see the first absorption law, $x \wedge (x \vee y) = x$, start with the diagram in the middle for $x \vee y$ and note that the portion of the shaded area in common with the x circle is the whole of the x circle. For the second absorption law, $x \vee (x \wedge y) = x$, start with the left diagram for $x \wedge y$ and note that shading the whole of the x circle results in just the x circle being shaded, since the previous shading was inside the x circle.

The double negation law can be seen by complementing the shading in the third diagram for $\neg x$, which shades the x circle.

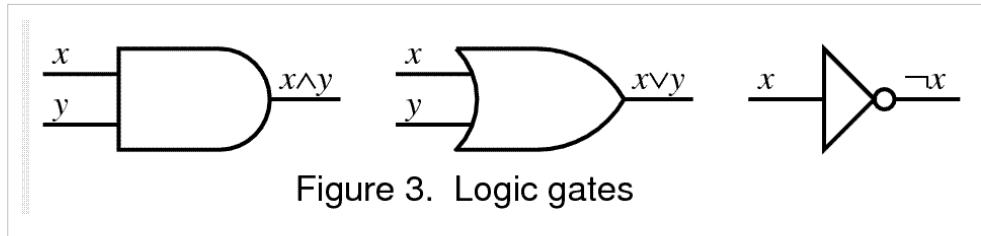
To visualize the first De Morgan's law, $(\neg x) \wedge (\neg y) = \neg(x \vee y)$, start with the middle diagram for $x \vee y$ and complement its shading so that only the region outside both circles is shaded, which is what the right hand side of the law describes. The result is the same as if we shaded that region which is both outside the x circle *and* outside the y circle, i.e. the conjunction of their exteriors, which is what the left hand side of the law describes.

The second De Morgan's law, $(\neg x) \vee (\neg y) = \neg(x \wedge y)$, works the same way with the two diagrams interchanged.

The first complement law, $x \wedge \neg x = 0$, says that the interior and exterior of the x circle have no overlap. The second complement law, $x \vee \neg x = 1$, says that everything is either inside or outside the x circle.

Digital logic gates

Digital logic is the application of the Boolean algebra of 0 and 1 to electronic hardware consisting of logic gates connected to form a circuit diagram. Each gate implements a Boolean operation, and is depicted schematically by a shape indicating the operation. The shapes associated with the gates for conjunction (AND-gates), disjunction (OR-gates), and complement (inverters) are as follows.

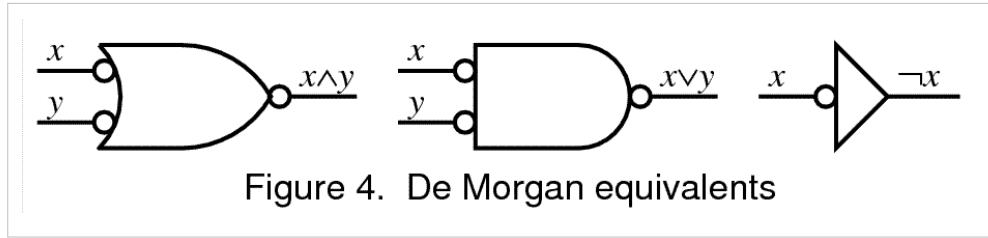


The lines on the left of each gate represent input wires or *ports*. The value of the input is represented by a voltage on the lead. For so-called "active-high" logic 0 is represented by a voltage close to zero or "ground" while 1 is represented by a voltage close to the supply voltage; active-low reverses this. The line on the right of each gate represents the output port, which normally follows the same voltage conventions as the input ports.

Complement is implemented with an inverter gate. The triangle denotes the operation that simply copies the input to the output; the small circle on the output denotes the actual inversion complementing the input. The convention of putting such a circle on any port means that the signal passing through this port is complemented on the way through, whether it is an input or output port.

There being eight ways of labeling the three ports of an AND-gate or OR-gate with inverters, this convention gives a wide range of possible Boolean operations realized as such gates so decorated. Not all combinations are distinct however: any labeling of AND-gate ports with inverters realizes the same Boolean operation as the opposite labeling of OR-gate ports (a given port of the AND-gate is labeled with an inverter if and only if the corresponding port of the OR-gate is not so labeled). This follows from De Morgan's laws.

If we complement all ports on every gate, and interchange AND-gates and OR-gates, as in Figure 4 below, we end up with the same operations as we started with, illustrating both De Morgan's laws and the Duality Principle. Note that we did not need to change the triangle part of the inverter, illustrating self-duality for complement.



Because of the pairwise identification of gates via the Duality Principle, even though 16 schematic symbols can be manufactured from the two basic binary gates AND and OR by furnishing their ports with inverters (circles), they only represent eight Boolean operations, namely those operations with an odd number of ones in their truth table. Altogether there are 16 binary Boolean operations, the other eight being those with an even number of ones in their truth table, namely the following. The constant 0, viewed as a binary operation that ignores both its inputs, has no ones, the six operations x , y , $\neg x$, $\neg y$ (as binary operations that ignore one input), $x \oplus y$, and $x \equiv y$ have two ones, and the constant 1 has four ones.

Boolean algebras

The term "algebra" denotes both a subject, namely the subject of algebra, and an object, namely an algebraic structure. Whereas the foregoing has addressed the subject of Boolean algebra, this section deals with mathematical objects called Boolean algebras, defined in full generality as any model of the Boolean laws. We begin with a special case of the notion definable without reference to the laws, namely concrete Boolean algebras, and then give the formal definition of the general notion.

Concrete Boolean algebras

A **concrete Boolean algebra** or field of sets is any nonempty set of subsets of a given set X closed under the set operations of union, intersection, and complement relative to X .

(As an aside, historically X itself was required to be nonempty as well to exclude the degenerate or one-element Boolean algebra, which is the one exception to the rule that all Boolean algebras satisfy the same equations since the degenerate algebra satisfies every equation. However this exclusion conflicts with the preferred purely equational definition of "Boolean algebra," there being no way to rule out the one-element algebra using only equations— $0 \neq 1$ does not count, being a negated equation. Hence modern authors allow the degenerate Boolean algebra and let X be empty.)

Example 1. The power set 2^X of X , consisting of all subsets of X . Here X may be any set: empty, finite, infinite, or even uncountable.

Example 2. The empty set and X . This two-element algebra shows that a concrete Boolean algebra can be finite even when it consists of subsets of an infinite set. It can be seen that every field of subsets of X must contain the empty set and X . Hence no smaller example is possible, other than the degenerate algebra obtained by taking X to be empty so as to make the empty set and X coincide.

Example 3. The set of finite and cofinite sets of integers, where a cofinite set is one omitting only finitely many integers. This is clearly closed under complement, and is closed under union because the union of a cofinite set with any set is cofinite, while the union of two finite sets is finite. Intersection behaves like union with "finite" and "cofinite" interchanged.

Example 4. For a less trivial example of the point made by Example 2, consider a Venn diagram formed by n closed curves partitioning the diagram into 2^n regions, and let X be the (infinite) set of all points in the plane not on any curve but somewhere within the diagram. The interior of each region is thus an infinite subset of X , and every point in X is in exactly one region. Then the set of all 2^n possible unions of regions (including the empty set obtained as the union of the empty set of regions and X obtained as the union of all 2^n regions) is closed under union,

intersection, and complement relative to X and therefore forms a concrete Boolean algebra. Again we have finitely many subsets of an infinite set forming a concrete Boolean algebra, with Example 2 arising as the case $n = 0$ of no curves.

Subsets as bit vectors

A subset Y of X can be identified with an indexed family of bits with index set X , with the bit indexed by $x \in X$ being 1 or 0 according to whether or not $x \in Y$. (This is the so-called characteristic function notion of a subset.) For example a 32-bit computer word consists of 32 bits indexed by the set $\{0,1,2,\dots,31\}$, with 0 and 31 indexing the low and high order bits respectively. For a smaller example, if $X = \{a,b,c\}$ where a, b, c are viewed as bit positions in that order from left to right, the eight subsets $\{\}, \{c\}, \{b\}, \{b,c\}, \{a\}, \{a,c\}, \{a,b\}$, and $\{a,b,c\}$ of X can be identified with the respective bit vectors 000, 001, 010, 011, 100, 101, 110, and 111. Bit vectors indexed by the set of natural numbers are infinite sequences of bits, while those indexed by the reals in the unit interval $[0,1]$ are packed too densely to be able to write conventionally but nonetheless form well-defined indexed families (imagine coloring every point of the interval $[0,1]$ either black or white independently; the black points then form an arbitrary subset of $[0,1]$).

From this bit vector viewpoint, a concrete Boolean algebra can be defined equivalently as a nonempty set of bit vectors all of the same length (more generally, indexed by the same set) and closed under the bit vector operations of bitwise \wedge , \vee , and \neg , as in $1010 \wedge 0110 = 0010$, $1010 \vee 0110 = 1110$, and $\neg 1010 = 0101$, the bit vector realizations of intersection, union, and complement respectively.

The prototypical Boolean algebra

The set $\{0,1\}$ and its Boolean operations as treated above can be understood as the special case of bit vectors of length one, which by the identification of bit vectors with subsets can also be understood as the two subsets of a one-element set. We call this the **prototypical** Boolean algebra, justified by the following observation.

The laws satisfied by all nondegenerate concrete Boolean algebras coincide with those satisfied by the prototypical Boolean algebra.

This observation is easily proved as follows. Certainly any law satisfied by all concrete Boolean algebras is satisfied by the prototypical one since it is concrete. Conversely any law that fails for some concrete Boolean algebra must have failed at a particular bit position, in which case that position by itself furnishes a one-bit counterexample to that law. Nondegeneracy ensures the existence of at least one bit position because there is only one empty bit vector.

The final goal of the next section can be understood as eliminating "concrete" from the above observation. We shall however reach that goal via the surprisingly stronger observation that, up to isomorphism, all Boolean algebras are concrete.

Boolean algebras: the definition

The Boolean algebras we have seen so far have all been concrete, consisting of bit vectors or equivalently of subsets of some set. Such a Boolean algebra consists of a set and operations on that set which can be *shown* to satisfy the laws of Boolean algebra.

Instead of showing that the Boolean laws are satisfied, we can instead postulate a set X , two binary operations on X , and one unary operation, and *require* that those operations satisfy the laws of Boolean algebra. The elements of X need not be bit vectors or subsets but can be anything at all. This leads to the more general *abstract* definition.

A Boolean algebra is any set with binary operations \wedge and \vee and a unary operation \neg thereon satisfying the Boolean laws.

For the purposes of this definition it is irrelevant how the operations came to satisfy the laws, whether by fiat or proof. All concrete Boolean algebras satisfy the laws (by proof rather than fiat), whence every concrete Boolean

algebra is a Boolean algebra according to our definitions. This axiomatic definition of a Boolean algebra as a set and certain operations satisfying certain laws or axioms *by fiat* is entirely analogous to the abstract definitions of group, ring, field etc. characteristic of modern or abstract algebra.

Given any complete axiomatization of Boolean algebra, such as the axioms for a complemented distributive lattice, a sufficient condition for an algebraic structure of this kind to satisfy all the Boolean laws is that it satisfy just those axioms. The following is therefore an equivalent definition.

A **Boolean algebra** is a complemented distributive lattice.

The section on axiomatization lists other axiomatizations, any of which can be made the basis of an equivalent definition.

Representable Boolean algebras

Although every concrete Boolean algebra is a Boolean algebra, not every Boolean algebra need be concrete. Let n be a square-free positive integer, one not divisible by the square of an integer, for example 30 but not 12. The operations of greatest common divisor, least common multiple, and division into n (that is, $\neg x = n/x$), can be shown to satisfy all the Boolean laws when their arguments range over the positive divisors of n . Hence those divisors form a Boolean algebra. These divisors are not subsets of a set, making the divisors of n a Boolean algebra that is not concrete according to our definitions.

However if we *represent* each divisor of n by the set of its prime factors, we find that this nonconcrete Boolean algebra is isomorphic to the concrete Boolean algebra consisting of all sets of prime factors of n , with union corresponding to least common multiple, intersection to greatest common divisor, and complement to division into n . So this example while not technically concrete is at least "morally" concrete via this representation, called an isomorphism. This example is an instance of the following notion.

A Boolean algebra is called **representable** when it is isomorphic to a concrete Boolean algebra.

The obvious next question is answered positively as follows.

Every Boolean algebra is representable.

That is, up to isomorphism, abstract and concrete Boolean algebras are the same thing. This quite nontrivial result depends on the Boolean prime ideal theorem, a choice principle slightly weaker than the axiom of choice, and is treated in more detail in the article Stone's representation theorem for Boolean algebras. This strong relationship implies a weaker result strengthening the observation in the previous subsection to the following easy consequence of representability.

The laws satisfied by all Boolean algebras coincide with those satisfied by the prototypical Boolean algebra.

It is weaker in the sense that it does not of itself imply representability. Boolean algebras are special here, for example a relation algebra is a Boolean algebra with additional structure but it is not the case that every relation algebra is representable in the sense appropriate to relation algebras.

Axiomatizing Boolean algebra

The above definition of an abstract Boolean algebra as a set and operations satisfying "the" Boolean laws raises the question, what are those laws? A simple-minded answer is "all Boolean laws," which can be defined as all equations that hold for the Boolean algebra of 0 and 1. Since there are infinitely many such laws this is not a terribly satisfactory answer in practice, leading to the next question: does it suffice to require only finitely many laws to hold?

In the case of Boolean algebras the answer is yes. In particular the finitely many equations we have listed above suffice. We say that Boolean algebra is **finitely axiomatizable** or **finitely based**.

Can this list be made shorter yet? Again the answer is yes. To begin with, some of the above laws are implied by some of the others. A sufficient subset of the above laws consists of the pairs of associativity, commutativity, and absorption laws, distributivity of \wedge over \vee (or the other distributivity law—one suffices), and the two complement laws. In fact this is the traditional axiomatization of Boolean algebra as a complemented distributive lattice.

By introducing additional laws not listed above it becomes possible to shorten the list yet further. In 1933 Edward Huntington showed that if the basic operations are taken to be $x \vee y$ and $\neg x$, with $x \wedge y$ considered a derived operation (e.g. via De Morgan's law in the form $x \wedge y = \neg(\neg x \vee \neg y)$), then the equation $\neg(\neg x \vee \neg y) \vee \neg(\neg x \vee y) = x$ along with the two equations expressing associativity and commutativity of \vee completely axiomatized Boolean algebra. When the only basic operation is the binary NAND operation $\neg(x \wedge y)$, Stephen Wolfram has proposed in his book *A New Kind of Science* the single axiom $((xy)z)(x((xz)x)) = z$ as a one-equation axiomatization of Boolean algebra, where for convenience here xy denotes the NAND rather than the AND of x and y .

Propositional logic

Propositional logic is a logical system that is intimately connected to Boolean algebra. Many syntactic concepts of Boolean algebra carry over to propositional logic with only minor changes in notation and terminology, while the semantics of propositional logic are defined via Boolean algebras in a way that the tautologies (theorems) of propositional logic correspond to equational theorems of Boolean algebra.

Syntactically, every Boolean term corresponds to a **propositional formula** of propositional logic. In this translation between Boolean algebra and propositional logic, Boolean variables x, y, \dots become **propositional variables** (or **atoms**) P, Q, \dots , Boolean terms such as $x \vee y$ become propositional formulas $P \vee Q$, 0 becomes *false* or \perp , and 1 becomes *true* or \top . It is convenient when referring to generic propositions to use Greek letters Φ, Ψ, \dots as metavariables (variables outside the language of propositional calculus, used when talking *about* propositional calculus) to denote propositions.

The semantics of propositional logic rely on **truth assignments**. The essential idea of a truth assignment is that the propositional variables are mapped to elements of a fixed Boolean algebra, and then the **truth value** of a propositional formula using these letters is the element of the Boolean algebra that is obtained by computing the value of the Boolean term corresponding to the formula. In classical semantics, only the two-element Boolean algebra is used, while in Boolean-valued semantics arbitrary Boolean algebras are considered. A **tautology** is a propositional formula that is assigned truth value 1 by every truth assignment of its propositional variables to an arbitrary Boolean algebra (or, equivalently, every truth assignment to the two element Boolean algebra).

These semantics permit a translation between tautologies of propositional logic and equational theorems of Boolean algebra. Every tautology Φ of propositional logic can be expressed as the Boolean equation $\Phi = 1$, which will be a theorem of Boolean algebra. Conversely every theorem $\Phi = \Psi$ of Boolean algebra corresponds to the tautologies $(\Phi \vee \neg \Psi) \wedge (\neg \Phi \vee \Psi)$ and $(\Phi \wedge \Psi) \vee (\neg \Phi \wedge \neg \Psi)$. If \rightarrow is in the language these last tautologies can also be written as $(\Phi \rightarrow \Psi) \wedge (\Psi \rightarrow \Phi)$, or as two separate theorems $\Phi \rightarrow \Psi$ and $\Psi \rightarrow \Phi$; if \equiv is available then the single tautology $\Phi \equiv \Psi$ can be used.

Applications

One motivating application of propositional calculus is the analysis of propositions and deductive arguments in natural language. Whereas the proposition "if $x = 3$ then $x+1 = 4$ " depends on the meanings of such symbols as + and 1, the proposition "if $x = 3$ then $x = 3$ " does not; it is true merely by virtue of its structure, and remains true whether " $x = 3$ " is replaced by " $x = 4$ " or "the moon is made of green cheese." The generic or abstract form of this tautology is "if P then P ", or in the language of Boolean algebra, " $P \rightarrow P$ ".

Replacing P by $x = 3$ or any other proposition is called **instantiation** of P by that proposition. The result of instantiating P in an abstract proposition is called an **instance** of the proposition. Thus " $x = 3 \rightarrow x = 3$ " is a tautology by virtue of being an instance of the abstract tautology " $P \rightarrow P$ ". All occurrences of the instantiated variable must be instantiated with the same proposition, to avoid such nonsense as $P \rightarrow x = 3$ or $x = 3 \rightarrow x = 4$.

Propositional calculus restricts attention to abstract propositions, those built up from propositional variables using Boolean operations. Instantiation is still possible within propositional calculus, but only by instantiating propositional variables by abstract propositions, such as instantiating Q by $Q \rightarrow P$ in $P \rightarrow (Q \rightarrow P)$ to yield the instance $P \rightarrow ((Q \rightarrow P) \rightarrow P)$.

(The availability of instantiation as part of the machinery of propositional calculus avoids the need for metavariables within the language of propositional calculus, since ordinary propositional variables can be considered within the language to denote arbitrary propositions. The metavariables themselves are outside the reach of instantiation, not being part of the language of propositional calculus but rather part of the same language for talking about it that this sentence is written in, where we need to be able to distinguish propositional variables and their instantiations as being distinct syntactic entities.)

Deductive systems for propositional logic

An axiomatization of propositional calculus is a set of tautologies called axioms and one or more inference rules for producing new tautologies from old. A *proof* in an axiom system A is a finite nonempty sequence of propositions each of which is either an instance of an axiom of A or follows by some rule of A from propositions appearing earlier in the proof (thereby disallowing circular reasoning). The last proposition is the **theorem** proved by the proof. Every nonempty initial segment of a proof is itself a proof, whence every proposition in a proof is itself a theorem. An axiomatization is **sound** when every theorem is a tautology, and **complete** when every tautology is a theorem.

Sequent calculus

Propositional calculus is commonly organized as a Hilbert system, whose operations are just those of Boolean algebra and whose theorems are Boolean tautologies, those Boolean terms equal to the Boolean constant 1. Another form is sequent calculus, which has two sorts, propositions as in ordinary propositional calculus, and pairs of lists of propositions called sequents, such as $A \vee B, A \wedge C, \dots \vdash A, B \rightarrow C, \dots$. The two halves of a sequent are called the antecedent and the succedent respectively. The customary metavariable denoting an antecedent or part thereof is Γ , and for a succedent Δ ; thus $\Gamma, A \vdash \Delta$ would denote a sequent whose succedent is a list Δ and whose antecedent is a list Γ with an additional proposition A appended after it. The antecedent is interpreted as the conjunction of its propositions, the succedent as the disjunction of its propositions, and the sequent itself as the entailment of the succedent by the antecedent.

Entailment differs from implication in that whereas the latter is a binary *operation* that returns a value in a Boolean algebra, the former is a binary *relation* which either holds or does not hold. In this sense entailment is an *external* form of implication, meaning external to the Boolean algebra, thinking of the reader of the sequent as also being external and interpreting and comparing antecedents and succedents in some Boolean algebra. The natural interpretation of \vdash is as \leq in the partial order of the Boolean algebra defined by $x \leq y$ just when $x \vee y = y$. This ability to mix external implication \vdash and internal implication \rightarrow in the one logic is among the essential differences between sequent calculus and propositional calculus.

Applications

Two-valued logic

Boolean algebra as the calculus of two values is fundamental to digital logic, computer programming, and mathematical logic, and is also used in other areas of mathematics such as set theory and statistics.

Digital logic codes its symbols in various ways: as voltages on wires in high-speed circuits and capacitive storage devices, as orientations of a magnetic domain in ferromagnetic storage devices, as holes in punched cards or paper tape, and so on. Now it is possible to code more than two symbols in any given medium. For example one might use respectively 0, 1, 2, and 3 volts to code a four-symbol alphabet on a wire, or holes of different sizes in a punched card. In practice however the tight constraints of high speed, small size, and low power combine to make noise a major factor. This makes it hard to distinguish between symbols when there are many of them at a single site. Rather than attempting to distinguish between four voltages on one wire, digital designers have settled on two voltages per wire, high and low. To obtain four symbols one uses two wires, and so on.

Programmers programming in machine code, assembly language, and other programming languages that expose the low-level digital structure of the data registers operate on whatever symbols were chosen for the hardware, invariably bit vectors in modern computers for the above reasons. Such languages support both the numeric operations of addition, multiplication, etc. performed on words interpreted as integers, as well as the logical operations of disjunction, conjunction, etc. performed bit-wise on words interpreted as bit vectors. Programmers therefore have the option of working in and applying the laws of either numeric algebra or Boolean algebra as needed. A core differentiating feature is carry propagation with the former but not the latter.

Other areas where two values is a good choice are the law and mathematics. In everyday relaxed conversation, nuanced or complex answers such as "maybe" or "only on the weekend" are acceptable. In more focused situations such as a court of law or theorem-based mathematics however it is deemed advantageous to frame questions so as to admit a simple yes-or-no answer—is the defendant guilty or not guilty, is the proposition true or false—and to disallow any other answer. However much of a straitjacket this might prove in practice for the respondent, the principle of the simple yes-no question has become a central feature of both judicial and mathematical logic, making two-valued logic deserving of organization and study in its own right.

A central concept of set theory is membership. Now an organization may permit multiple degrees of membership, such as novice, associate, and full. With sets however an element is either in or out. The candidates for membership in a set work just like the wires in a digital computer: each candidate is either a member or a nonmember, just as each wire is either high or low.

Algebra being a fundamental tool in any area amenable to mathematical treatment, these considerations combine to make the algebra of two values of fundamental importance to computer hardware, mathematical logic, and set theory.

Two-valued logic can be extended to multi-valued logic, notably by replacing the Boolean domain $\{0, 1\}$ with the unit interval $[0,1]$, in which case rather than only taking values 0 or 1, any value between and including 0 and 1 can be assumed. Algebraically, negation (NOT) is replaced with $1 - x$, conjunction (AND) is replaced with multiplication (xy), and disjunction (OR) is defined via De Morgan's law. Interpreting these values as logical truth values yields a multi-valued logic, which forms the basis for fuzzy logic and probabilistic logic. In these interpretations, a value is interpreted as the "degree" of truth – to what extent a proposition is true, or the probability that the proposition is true.

Boolean operations

The original application for Boolean operations was mathematical logic, where it combines the truth values, true or false, of individual formulas.

Natural languages such as English have words for several Boolean operations, in particular conjunction (*and*), disjunction (*or*), negation (*not*), and implication (*implies*). *But not* is synonymous with *and not*. When used to combine situational assertions such as "the block is on the table" and "cats drink milk," which naively are either true or false, the meanings of these logical connectives often have the meaning of their logical counterparts. However with descriptions of behavior such as "Jim walked through the door", one starts to notice differences such as failure of commutativity, for example the conjunction of "Jim opened the door" with "Jim walked through the door" in that order is not equivalent to their conjunction in the other order, since *and* usually means *and then* in such cases. Questions can be similar: the order "Is the sky blue, and why is the sky blue?" makes more sense than the reverse order. Conjunctive commands about behavior are like behavioral assertions, as in *get dressed and go to school*. Disjunctive commands such *love me or leave me* or *fish or cut bait* tend to be asymmetric via the implication that one alternative is less preferable. Conjoined nouns such as *tea and milk* generally describe aggregation as with set union while *tea or milk* is a choice. However context can reverse these senses, as in *your choices are coffee and tea* which usually means the same as *your choices are coffee or tea* (alternatives). Double negation as in "I don't not like milk" rarely means literally "I do like milk" but rather conveys some sort of hedging, as though to imply that there is a third possibility. "Not not P" can be loosely interpreted as "surely P", and although *P* necessarily implies "not not *P*" the converse is suspect in English, much as with intuitionistic logic. In view of the highly idiosyncratic usage of conjunctions in natural languages, Boolean algebra cannot be considered a reliable framework for interpreting them.

Boolean operations are used in digital logic to combine the bits carried on individual wires, thereby interpreting them over $\{0,1\}$. When a vector of n identical binary gates are used to combine two bit vectors each of n bits, the individual bit operations can be understood collectively as a single operation on values from a Boolean algebra with 2^n elements.

Naive set theory interprets Boolean operations as acting on subsets of a given set X . As we saw earlier this behavior exactly parallels the coordinate-wise combinations of bit vectors, with the union of two sets corresponding to the disjunction of two bit vectors and so on.

The 256-element free Boolean algebra on three generators is deployed in computer displays based on raster graphics, which use bit blit to manipulate whole regions consisting of pixels, relying on Boolean operations to specify how the source region should be combined with the destination, typically with the help of a third region called the mask. Modern video cards offer all $2^3 = 256$ ternary operations for this purpose, with the choice of operation being a one-byte (8-bit) parameter. The constants SRC = 0xaa or 10101010, DST = 0xcc or 11001100, and MSK = 0xf0 or 11110000 allow Boolean operations such as (SRC \wedge DST)&MSK (meaning XOR the source and destination and then AND the result with the mask) to be written directly as a constant denoting a byte calculated at compile time, 0x60 in the (SRC \wedge DST)&MSK example, 0x66 if just SRC \wedge DST, etc. At run time the video card interprets the byte as the raster operation indicated by the original expression in a uniform way that requires remarkably little hardware and which takes time completely independent of the complexity of the expression.

Solid modeling systems for computer aided design offer a variety of methods for building objects from other objects, combination by Boolean operations being one of them. In this method the space in which objects exist is understood as a set S of voxels (the three-dimensional analogue of pixels in two-dimensional graphics) and shapes are defined as subsets of S , allowing objects to be combined as sets via union, intersection, etc. One obvious use is in building a complex shape from simple shapes simply as the union of the latter. Another use is in sculpting understood as removal of material: any grinding, milling, routing, or drilling operation that can be performed with physical machinery on physical materials can be simulated on the computer with the Boolean operation $x \wedge \neg y$ or $x - y$, which in set theory is set difference, remove the elements of y from those of x . Thus given two shapes one to be machined and the other the material to be removed, the result of machining the former to remove the latter is described simply

as their set difference.

Boolean searches

Search engine queries also employ Boolean logic. For this application, each web page on the Internet may be considered to be an "element" of a "set". The following examples use a syntax supported by Google.^[5]

- Doublequotes are used to combine whitespace-separated words into a single search term.^[6]
- Whitespace is used to specify logical AND, as it is the default operator for joining search terms:

```
"Search term 1" "Search term 2"
```

- The OR keyword is used for logical OR:

```
"Search term 1" OR "Search term 2"
```

- The minus sign is used for logical NOT (AND NOT):

```
"Search term 1" - "Search term 2"
```

References

- [1] cf footnote on page 278: "* The name Boolean algebra (or Boolean "algebras") for the calculus originated by Boole, extended by Schröder, and perfected by Whitehead seems to have been first suggested by Sheffer, in 1913" quoted from E. V. Huntington January 1933, "NEW SETS OF INDEPENDENT POSTULATES FOR THE ALGEBRA OF LOGIC, WITH SPECIAL REFERENCE TO WHITEHEAD AND RUSSELL'S PRINCIPIA MATHEMATICA", [http://www.ams.org/journals/tran/1933-035-01/S0002-9947-1933-1501684-X.pdf](http://www.ams.org/journals/tran/1933-035-01/S0002-9947-1933-1501684-X/S0002-9947-1933-1501684-X.pdf)
- [2] , online sample (<http://www.wiley.com/college/engin/balabanian293512/pdf/ch02.pdf>)
- [3] Halmos, Paul (1963). Lectures on Boolean Algebras. van Nostrand.
- [4] J. Venn, *On the Diagrammatic and Mechanical Representation of Propositions and Reasonings*, Philosophical Magazine and Journal of Science, Series 5, vol. **10**, No. 59, July 1880.
- [5] Not all search engines support the same query syntax. Additionally, some organizations (such as Google) provide "specialized" search engines that support alternate or extended syntax. (See e.g., Syntax cheatsheet (<http://www.google.com/help/cheatsheet.html>), Google codesearch supports regular expressions (http://www.google.com/intl/en/help/faq_codesearch.html#regexp)).
- [6] Doublequote-delimited search terms are called "exact phrase" searches in the Google documentation.

Mano, Morris; Ciletti, Michael D. (2013). *Digital Design*. Pearson. ISBN 978-0-13-277420-8.

Further reading

- J. Eldon Whitesitt (1995). *Boolean algebra and its applications*. Courier Dover Publications. ISBN 978-0-486-68483-3. Suitable introduction for students in applied fields.
- Dwinger, Philip (1971). *Introduction to Boolean algebras*. Würzburg: Physica Verlag.
- Sikorski, Roman (1969). *Boolean Algebras* (3/e ed.). Berlin: Springer-Verlag. ISBN 978-0-387-04469-9.
- Bocheński, Józef Maria (1959). *A Précis of Mathematical Logic*. Translated from the French and German editions by Otto Bird. Dordrecht, South Holland: D. Reidel.

Historical perspective

- George Boole (1848). " The Calculus of Logic, (<http://www.maths.tcd.ie/pub/HistMath/People/Boole/CalcLogic/CalcLogic.html>)" *Cambridge and Dublin Mathematical Journal III: 183–98*.
- Theodore Hailperin (1986). *Boole's logic and probability: a critical exposition from the standpoint of contemporary algebra, logic, and probability theory* (2nd ed.). Elsevier. ISBN 978-0-444-87952-3.
- Dov M. Gabbay, John Woods, ed. (2004). *The rise of modern logic: from Leibniz to Frege*. Handbook of the History of Logic **3**. Elsevier. ISBN 978-0-444-51611-4., several relevant chapters by Hailperin, Valencia, and Grattan-Guinesss
- Calixto Badesa (2004). *The birth of model theory: Löwenheim's theorem in the frame of the theory of relatives*. Princeton University Press. ISBN 978-0-691-05853-5., chapter 1, "Algebra of Classes and Propositional

Calculus"

- Burris, Stanley, 2009. The Algebra of Logic Tradition (<http://plato.stanford.edu/entries/algebra-logic-tradition/>). Stanford Encyclopedia of Philosophy.
- Radomir S. Stankovic; Jaakko Astola (2011). *From Boolean Logic to Switching Circuits and Automata: Towards Modern Information Technology* (<http://books.google.com/books?id=uagvEc2jGTIC>). Springer. ISBN 978-3-642-11681-0.

External links

- How Stuff Works – Boolean Logic (<http://computer.howstuffworks.com/boolean.htm>)
- Science and Technology - Boolean Algebra (<http://oscience.info/mathematics/boolean-algebra-2/>) contains a list and proof of Boolean theorems and laws.

Field of sets

In mathematics a **field of sets** is a pair $\langle X, \mathcal{F} \rangle$ where X is a set and \mathcal{F} is an **algebra over X** i.e., a non-empty subset of the power set of X closed under the intersection and union of pairs of sets and under complements of individual sets. In other words \mathcal{F} forms a subalgebra of the power set Boolean algebra of X . (Many authors refer to \mathcal{F} itself as a field of sets. The word "field" in "field of sets" is not used with the meaning of field from field theory.) Elements of X are called **points** and those of \mathcal{F} are called **complexes** and are said to be the **admissible sets of X** .

Fields of sets play an essential role in the representation theory of Boolean algebras. Every Boolean algebra can be represented as a field of sets.

Fields of sets in the representation theory of Boolean algebras

Stone representation

Every finite Boolean algebra can be represented as a whole power set - the power set of its set of atoms; each element of the Boolean algebra corresponds to the set of atoms below it (the join of which is the element). This **power set representation** can be constructed more generally for any complete atomic Boolean algebra.

In the case of Boolean algebras which are not complete and atomic we can still generalize the power set representation by considering fields of sets instead of whole power sets. To do this we first observe that the atoms of a finite Boolean algebra correspond to its ultrafilters and that an atom is below an element of a finite Boolean algebra if and only if that element is contained in the ultrafilter corresponding to the atom. This leads us to construct a representation of a Boolean algebra by taking its set of ultrafilters and forming complexes by associating with each element of the Boolean algebra the set of ultrafilters containing that element. This construction does indeed produce a representation of the Boolean algebra as a field of sets and is known as the **Stone representation**. It is the basis of Stone's representation theorem for Boolean algebras and an example of a completion procedure in order theory based on ideals or filters, similar to Dedekind cuts.

Alternatively one can consider the set of homomorphisms onto the two element Boolean algebra and form complexes by associating each element of the Boolean algebra with the set of such homomorphisms that map it to the top element. (The approach is equivalent as the ultrafilters of a Boolean algebra are precisely the pre-images of the top elements under these homomorphisms.) With this approach one sees that Stone representation can also be regarded as a generalization of the representation of finite Boolean algebras by truth tables.

Separative and compact fields of sets: towards Stone duality

- A field of sets is called **separative** (or **differentiated**) if and only if for every pair of distinct points there is a complex containing one and not the other.
- A field of sets is called **compact** if and only if for every proper filter over X the intersection of all the complexes contained in the filter is non-empty.

These definitions arise from considering the topology generated by the complexes of a field of sets. Given a field of sets $\mathbf{X} = \langle X, \mathcal{F} \rangle$ the complexes form a base for a topology, we denote the corresponding topological space by $T(\mathbf{X})$. Then

- $T(\mathbf{X})$ is always a zero-dimensional space.
- $T(\mathbf{X})$ is a Hausdorff space if and only if \mathbf{X} is separative.
- $T(\mathbf{X})$ is a compact space with compact open sets \mathcal{F} if and only if \mathbf{X} is compact.
- $T(\mathbf{X})$ is a Boolean space with clopen sets \mathcal{F} if and only if \mathbf{X} is both separative and compact (in which case it is described as being **descriptive**)

The Stone representation of a Boolean algebra is always separative and compact; the corresponding Boolean space is known as the Stone space of the Boolean algebra. The clopen sets of the Stone space are then precisely the complexes of the Stone representation. The area of mathematics known as Stone duality is founded on the fact that the Stone representation of a Boolean algebra can be recovered purely from the corresponding Stone space whence a duality exists between Boolean algebras and Boolean spaces.

Fields of sets with additional structure

Sigma algebras and measure spaces

If an algebra over a set is closed under countable intersections and countable unions, it is called a sigma algebra and the corresponding field of sets is called a **measurable space**. The complexes of a measurable space are called **measurable sets**.

A **measure space** is a triple $\langle X, \mathcal{F}, \mu \rangle$ where $\langle X, \mathcal{F} \rangle$ is a measurable space and μ is a measure defined on it.

If μ is in fact a probability measure we speak of a **probability space** and call its underlying measurable space a **sample space**. The points of a sample space are called **samples** and represent potential outcomes while the measurable sets (complexes) are called **events** and represent properties of outcomes for which we wish to assign probabilities. (Many use the term **sample space** simply for the underlying set of a probability space, particularly in the case where every subset is an event.) Measure spaces and probability spaces play a foundational role in measure theory and probability theory respectively.

The Loomis-Sikorski theorem provides a Stone-type duality between abstract sigma algebras and measurable spaces.

Topological fields of sets

A **topological field of sets** is a triple $\langle X, T, \mathcal{F} \rangle$ where $\langle X, T \rangle$ is a topological space and $\langle X, \mathcal{F} \rangle$ is a field of sets which is closed under the closure operator of T or equivalently under the interior operator i.e. the closure and interior of every complex is also a complex. In other words \mathcal{F} forms a subalgebra of the power set interior algebra on $\langle X, T \rangle$.

Every interior algebra can be represented as a topological field of sets with its interior and closure operators corresponding to those of the topological space.

Given a topological space the clopen sets trivially form a topological field of sets as each clopen set is its own interior and closure. The Stone representation of a Boolean algebra can be regarded as such a topological field of sets.

Algebraic fields of sets and Stone fields

A topological field of sets is called **algebraic** if and only if there is a base for its topology consisting of complexes.

If a topological field of sets is both compact and algebraic then its topology is compact and its compact open sets are precisely the open complexes. Moreover the open complexes form a base for the topology.

Topological fields of sets that are separative, compact and algebraic are called **Stone fields** and provide a generalization of the Stone representation of Boolean algebras. Given an interior algebra we can form the Stone representation of its underlying Boolean algebra and then extend this to a topological field of sets by taking the topology generated by the complexes corresponding to the open elements of the interior algebra (which form a base for a topology). These complexes are then precisely the open complexes and the construction produces a Stone field representing the interior algebra - the **Stone representation**.

Preorder fields

A **preorder field** is a triple $\langle X, \leq, \mathcal{F} \rangle$ where $\langle X, \leq \rangle$ is a preordered set and $\langle X, \mathcal{F} \rangle$ is a field of sets.

Like the topological fields of sets, preorder fields play an important role in the representation theory of interior algebras. Every interior algebra can be represented as a preorder field with its interior and closure operators corresponding to those of the Alexandrov topology induced by the preorder. In other words

$$\text{Int}(S) = \{x \in X : \text{there exists a } y \in S \text{ with } y \leq x\} \text{ and}$$

$$\text{Cl}(S) = \{x \in X : \text{there exists a } y \in S \text{ with } x \leq y\} \text{ for all } S \in \mathcal{F}$$

Preorder fields arise naturally in modal logic where the points represent the *possible worlds* in the Kripke semantics of a theory in the modal logic *S4* (a formal mathematical abstraction of epistemic logic), the preorder represents the accessibility relation on these possible worlds in this semantics, and the complexes represent sets of possible worlds in which individual sentences in the theory hold, providing a representation of the Lindenbaum-Tarski algebra of the theory.

Algebraic and canonical preorder fields

A preorder field is called **algebraic** if and only if it has a set of complexes \mathcal{A} which determines the preorder in the following manner: $x \leq y$ if and only if for every complex $S \in \mathcal{A}$, $x \in S$ implies $y \in S$. The preorder fields obtained from *S4* theories are always algebraic, the complexes determining the preorder being the sets of possible worlds in which the sentences of the theory closed under necessity hold.

A separative compact algebraic preorder field is said to be **canonical**. Given an interior algebra, by replacing the topology of its Stone representation with the corresponding canonical preorder (specialization preorder) we obtain a representation of the interior algebra as a canonical preorder field. By replacing the preorder by its corresponding Alexandrov topology we obtain an alternative representation of the interior algebra as a topological field of sets. (The topology of this "**Alexandrov representation**" is just the Alexandrov bi-coreflection of the topology of the Stone representation.)

Complex algebras and fields of sets on relational structures

The representation of interior algebras by preorder fields can be generalized to a representation theorem for arbitrary (normal) Boolean algebras with operators. For this we consider structures $\langle X, (R_i)_I, \mathcal{F} \rangle$ where $\langle X, (R_i)_I \rangle$ is a relational structure i.e. a set with an indexed family of relations defined on it, and $\langle X, \mathcal{F} \rangle$ is a field of sets. The **complex algebra** (or **algebra of complexes**) determined by a field of sets $\mathbf{X} = \langle X, (R_i)_I, \mathcal{F} \rangle$ on a relational structure, is the Boolean algebra with operators

$$\mathcal{C}(\mathbf{X}) = \langle \mathcal{F}, \cap, \cup, \iota, \emptyset, X, (f_i)_I \rangle$$

where for all $i \in I$, if R_i is a relation of arity $n + 1$, then f_i is an operator of arity n and for all $S_1, \dots, S_n \in \mathcal{F}$

$$f_i(S_1, \dots, S_n) = \{x \in X : \text{there exist } x_1 \in S_1, \dots, x_n \in S_n \text{ such that } R_i(x_1, \dots, x_n, x)\}$$

This construction can be generalized to fields of sets on arbitrary algebraic structures having both operators and relations as operators can be viewed as a special case of relations. If \mathcal{F} is the whole power set of X then $\mathcal{C}(\mathbf{X})$ is called a **full complex algebra or power algebra**.

Every (normal) Boolean algebra with operators can be represented as a field of sets on a relational structure in the sense that it is isomorphic to the complex algebra corresponding to the field.

(Historically the term **complex** was first used in the case where the algebraic structure was a group and has its origins in 19th century group theory where a subset of a group was called a **complex**.)

References

- Goldblatt, R., *Algebraic Polymodal Logic: A Survey*, Logic Journal of the IGPL, Volume 8, Issue 4, p. 393-450, July 2000
- Goldblatt, R., *Varieties of complex algebras*, Annals of Pure and Applied Logic, 44, p. 173-242, 1989
- Johnstone, Peter T. (1982). *Stone spaces* (3rd edition ed.). Cambridge: Cambridge University Press. ISBN 0-521-33779-8.
- Naturman, C.A., *Interior Algebras and Topology*, Ph.D. thesis, University of Cape Town Department of Mathematics, 1991
- Patrick Blackburn, Johan F.A.K. van Benthem, Frank Wolter ed., *Handbook of Modal Logic, Volume 3 of Studies in Logic and Practical Reasoning*, Elsevier, 2006

External links

- Hazewinkel, Michiel, ed. (2001), "Algebra of sets" [1], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

References

[1] <http://www.encyclopediaofmath.org/index.php?title=p/a011400>

List of logic symbols

In logic, a set of symbols is commonly used to express logical representation. As logicians are familiar with these symbols, they are not explained each time they are used. So, for students of logic, the following table lists many common symbols together with their name, pronunciation, and the related field of mathematics. Additionally, the third column contains an informal definition, and the fourth column gives a short example.

Be aware that, outside of logic, different symbols have the same meaning, and the same symbol has, depending on the context, different meanings.

Basic logic symbols

Symbol	Name	Explanation	Examples	Unicode Value	HTML Entity	LaTeX symbol
	Should be read as					
	Category					
\Rightarrow \rightarrow \supset	material implication	$A \Rightarrow B$ is true just in the case that either A is false or B is true, or both. \rightarrow may mean the same as \Rightarrow (the symbol may also indicate the domain and codomain of a function; see table of mathematical symbols). \supset may mean the same as \Rightarrow (the symbol may also mean superset).	$x = 2 \Rightarrow x^2 = 4$ is true, but $x^2 = 4 \Rightarrow x = 2$ is in general false (since x could be -2).	U+21D2 U+2192 U+2283	⇒ → ⊃	$\Rightarrow \backslash Rightarrow$ $\rightarrow \backslash to$ $\supset \backslash supset$ $\implies \backslash implies$
	implies; if .. then					
	propositional logic, Heyting algebra					
\Leftrightarrow \equiv \leftrightarrow	material equivalence	$A \Leftrightarrow B$ is true just in case either both A and B are false, or both A and B are true.	$x + 5 = y + 2 \Leftrightarrow x + 3 = y$	U+21D4 U+2261 U+2194	⇔ ≡ ↔	$\Leftrightarrow \backslash Leftrightarrow$ $\equiv \backslash equiv$ $\leftrightarrow \backslash Leftrightarrow$ $\iff \backslash iff$
	if and only if; iff; means the same as					
	propositional logic					
\neg \sim $!$	negation	The statement $\neg A$ is true if and only if A is false.	$\neg(\neg A) \Leftrightarrow A$	U+00AC U+02DC	¬ ˜ ~	$\neg \backslash not$ or $\backslash neg$ $\sim \backslash sim$
	not	A slash placed through another operator is the same as " \neg " placed in front.	$x \neq y \Leftrightarrow \neg(x = y)$			
	propositional logic					
\wedge \bullet $\&$	logical conjunction	The statement $A \wedge B$ is true if A and B are both true; else it is false.	$n < 4 \wedge n > 2 \Leftrightarrow n = 3$ when n is a natural number.	U+2227 U+0026	∧ &	$\wedge \backslash wedge$ or $\backslash land$ $\& \backslash [1]$
	and					
	propositional logic					

∨	logical disjunction	The statement $A \vee B$ is true if A or B (or both) are true; if both are false, the statement is false.	$n \geq 4 \vee n \leq 2 \Leftrightarrow n \neq 3$ when n is a natural number.	U+2228	∨	$\vee \backslash\text{or}$ or $\backslash\text{vee}$
+	or					
 	propositional logic					
⊕	exclusive disjunction	The statement $A \oplus B$ is true when either A or B , but not both, are true. $A \sqcup B$ means the same.	$(\neg A) \oplus A$ is always true, $A \oplus A$ is always false.	U+2295 U+22BB	⊕	$\oplus \backslash\text{oplus}$ $\vee \backslash\text{veebar}$
⊤	Tautology	The statement \top is unconditionally true.	$A \Rightarrow \top$ is always true.	U+22A4	T	$\top \backslash\text{top}$
⊤	top, verum					
1	propositional logic, Boolean algebra					
⊥	Contradiction	The statement \perp is unconditionally false.	$\perp \Rightarrow A$ is always true.	U+22A5	⊥ F	$\perp \backslash\text{bot}$
F	bottom, falsum					
0	propositional logic, Boolean algebra					
∀	universal quantification	$\forall x: P(x)$ or $(x) P(x)$ means $P(x)$ is true for all x .	$\forall n \in \mathbb{N}: n^2 \geq n$.	U+2200	∀	$\forall \backslash\text{forall}$
(for all; for any; for each					
)	first-order logic					
∃	existential quantification	$\exists x: P(x)$ means there is at least one x such that $P(x)$ is true.	$\exists n \in \mathbb{N}: n$ is even.	U+2203	∃	$\exists \backslash\text{exists}$
	there exists					
	first-order logic					
∃!	uniqueness quantification	$\exists! x: P(x)$ means there is exactly one x such that $P(x)$ is true.	$\exists! n \in \mathbb{N}: n + 5 = 2n$.	U+2203 U+0021	∃!;	$\exists! \backslash\text{exists} !$
	there exists exactly one					
	first-order logic					
:=	definition	$x := y$ or $x \equiv y$ means x is defined to be another name for y (but note that \equiv can also mean other things, such as congruence). $P := Q$ means P is defined to be logically equivalent to Q .	$\cosh x := (1/2)(\exp x + \exp(-x))$ $A \text{ XOR } B := (A \vee B) \wedge \neg(A \wedge B)$	U+2254 (U+003A U+003D) U+2261 U+003A U+229C	:= : ≡ ⇔	$:=:=$ $\equiv \backslash\text{equiv}$ $\Leftrightarrow \backslash\text{Leftrightarrow}$
≡	is defined as					
↔	everywhere					

()	precedence grouping parentheses, brackets everywhere	Perform the operations inside the parentheses first.	$(8 \div 4) \div 2 = 2 \div 2 = 1$, but $8 \div (4 \div 2) = 8 \div 2 = 4$.	U+0028 U+0029	()	()()
⊦	Turnstile	$x \vdash y$ means y is provable from x (in some specified formal system).	$A \rightarrow B \vdash \neg B \rightarrow \neg A$	U+22A2	⊢	⊦ \vdash
	provable					
	propositional logic, first-order logic					
⊧	double turnstile	$x \Vdash y$ means x semantically entails y	$A \rightarrow B \Vdash \neg B \rightarrow \neg A$	U+22A8	⊨	⊧ \models
	entails					
	propositional logic, first-order logic					

Advanced and rarely used logical symbols

These symbols are sorted by their Unicode value:

- U+00B7 · middle dot, an outdated way for denoting AND^[citation needed], still in use in electronics; for example "A·B" is the same as "A&B"
- · : Center dot with a line above it. Outdated way for denoting NAND, for example "A·B" is the same as "A NAND B" or "A|B" or " $\neg(A \& B)$ ". See also Unicode U+22C5 · dot operator.
- U+0305 ¯ combining overline, used as abbreviation for standard numerals. For example, using HTML style "Ā" is a shorthand for the standard numeral "SSSSO".
- Overline, is also a rarely used format for denoting Gödel numbers, for example "ĀVB" says the Gödel number of "(AVB)"
- Overline is also an outdated way for denoting negation, still in use in electronics; for example "ĀVB" is the same as " $\neg(\overline{A} \& B)$ "
- U+2191 ↑ upwards arrow or U+007C | vertical line: Sheffer stroke, the sign for the NAND operator.
- U+2201 ⊥ complement
- U+2204 ⊮ there does not exist: strike out existential quantifier same as " $\neg\exists$ "
- U+2234 ∴ therefore
- U+2235 ⊥ because
- U+22A7 ⊨ models: is a model of
- U+22A8 ⊤ true: is true of
- U+22AC ⊥ does not prove: negated ⊨, the sign for "does not prove", for example $T \bot P$ says " P is not a theorem of T "
- U+22AD ⊥ not true: is not true of
- U+22BC ⊥ nand: another NAND operator, can also be rendered as \wedge
- U+22BD ⊥ nor: another NOR operator, can also be rendered as \vee
- U+22C4 ⊦ diamond operator: modal operator for "it is possible that", "it is not necessarily not" or rarely "it is not provable not" (in most modal logics it is defined as " $\neg\Box\neg$ ")
- U+22C6 ★ star operator: usually used for ad-hoc operators
- U+22A5 ⊥ up tack or U+2193 ↓ downwards arrow: Webb-operator or Peirce arrow, the sign for NOR. Confusingly, " \perp " is also the sign for contradiction or absurdity.

- U+2310 ⌐ reversed not sign
- U+231C ⌑ top left corner and U+231D ⌒ top right corner: corner quotes, also called "Quine quotes"; for quasi-quotation, i.e. quoting specific context of unspecified ("variable") expressions^[2]; also the standard symbol^[citation needed] used for denoting Gödel number; for example "⌑G⌒" denotes the Gödel number of G. (Typographical note: although the quotes appears as a "pair" in unicode (231C and 231D), they are not symmetrical in some fonts. And in some fonts (for example Arial) they are only symmetrical in certain sizes. Alternatively the quotes can be rendered as ⌑ and ⌒ (U+2308 and U+2309) or by using a negation symbol and a reversed negation symbol ⌐ ¬ in superscript mode.)
- U+25FB ☐ white medium square or U+25A1 ☐ white square: modal operator for "it is necessary that" (in modal logic), or "it is provable that" (in provability logic), or "it is obligatory that" (in deontic logic), or "it is believed that" (in doxastic logic).

Note that the following operators are rarely supported by natively installed fonts. If you wish to use these in a web page, you should always embed the necessary fonts so the page viewer can see the web page without having the necessary fonts installed in their computer.

- U+27E1 ⌈ white concave-sided diamond
- U+27E2 ⌉ white concave-sided diamond with leftwards tick: modal operator for was never
- U+27E3 ⌊ white concave-sided diamond with rightwards tick: modal operator for will never be
- U+27E4 ⌋ white square with leftwards tick: modal operator for was always
- U+27E5 ⌌ white square with rightwards tick: modal operator for will always be
- U+297D ⌍ right fish tail: sometimes used for "relation", also used for denoting various ad hoc relations (for example, for denoting "witnessing" in the context of Rosser's trick) The fish hook is also used as strict implication by C.I.Lewis $p \sqcap q \equiv \square(p \rightarrow q)$, the corresponding LaTeX macro is \strictif. See here^[3] for an image of glyph. Added to Unicode 3.2.0.

Notes

[1] Although this character is available in LaTeX, the Mediawiki TeX system doesn't support this character.

[2] Quine, W.V. (1981): *Mathematical Logic*, §6

[3] <http://www.fileformat.info/info/unicode/char/297d/index.htm>

External links

- Named character entities (<http://www.w3.org/TR/WD-html40-970708/sgml/entities.html>) in HTML 4.0

Logical connective

In logic, a **logical connective** (also called a **logical operator**) is a symbol or word used to connect two or more sentences (of either a formal or a natural language) in a grammatically valid way, such that the sense of the compound sentence produced depends only on the original sentences.

The most common logical connectives are **binary connectives** (also called **dyadic connectives**) which join two sentences which can be thought of as the function's operands. Also commonly, negation is considered to be a **unary connective**.

Logical connectives along with quantifiers are the two main types of logical constants used in formal systems such as propositional logic and predicate logic. Semantics of a logical connective is often, but not always, presented as a truth function.

In language

Natural language

In the grammar of natural languages two sentences may be joined by a grammatical conjunction to form a *grammatically* compound sentence. Some but not all such grammatical conjunctions are truth functions. For example, consider the following sentences:

- A: Jack went up the hill.
- B: Jill went up the hill.
- C: Jack went up the hill *and* Jill went up the hill.
- D: Jack went up the hill *so* Jill went up the hill.

The words *and* and *so* are *grammatical* conjunctions joining the sentences (A) and (B) to form the compound sentences (C) and (D). The *and* in (C) is a *logical* connective, since the truth of (C) is completely determined by (A) and (B): it would make no sense to affirm (A) and (B) but deny (C). However *so* in (D) is not a logical connective, since it would be quite reasonable to affirm (A) and (B) but deny (D): perhaps, after all, Jill went up the hill to fetch a pail of water, not because Jack had gone up the hill at all.

Various English words and word pairs express logical connectives, and some of them are synonymous. Examples (with the name of the relationship in parentheses) are:

- "and" (conjunction)
- "or" (disjunction)
- "either...or" (exclusive disjunction)
- "implies" (implication)
- "if...then" (implication)
- "if and only if" (equivalence)
- "only if" (implication)
- "just in case" (biconditional)
- "but" (conjunction)
- "however" (conjunction)
- "not both" (alternative denial)
- "neither...nor" (joint denial)

The word "not" (negation) and the phrases "it is false that" (negation) and "it is not the case that" (negation) also express a logical connective – even though they are applied to a single statement, and do not connect two statements.

Formal languages

In formal languages, truth functions are represented by unambiguous symbols. These symbols are called "logical connectives", "logical operators", "propositional operators", or, in classical logic, "truth-functional connectives". See well-formed formula for the rules which allow new well-formed formulas to be constructed by joining other well-formed formulas using truth-functional connectives.

Logical connectives can be used to link more than two statements, so one can speak about "n-ary logical connective".

Common logical connectives

Name / Symbol		Truth table	Venn diagram
		$P = \begin{matrix} 0 & 1 \end{matrix}$	
Truth/Tautology	T	1 1	
Proposition P		0 1	
False/Contradiction	⊥	0 0	
Negation	¬	1 0	
Binary connectives		$P = \begin{matrix} 0 & 0 & 1 & 1 \end{matrix}$	
		$Q = \begin{matrix} 0 & 1 & 0 & 1 \end{matrix}$	
Conjunction	∧	0 0 0 1	
Alternative denial	↑	1 1 1 0	
Disjunction	∨	0 1 1 1	
Joint denial	↓	1 0 0 0	
Material conditional	→	1 1 0 1	
Exclusive or	↔	0 1 1 0	
Biconditional	↔	1 0 0 1	
Converse implication	←	1 0 1 1	
Proposition P		0 0 1 1	
Proposition Q		0 1 0 1	

[More information](#)

List of common logical connectives

Commonly used logical connectives include:

- Negation (not): \neg , Np , \sim
- Conjunction (and): \wedge , Kpq , $\&$, \cdot
- Disjunction (or): \vee , Apq
- Material implication (if...then): \rightarrow , Cpq , \Rightarrow , \supset
- Biconditional (if and only if): \leftrightarrow , Epq , \equiv , $=$

Alternative names for biconditional are "iff", "xnor" and "bi-implication".

For example, the meaning of the statements *it is raining* and *I am indoors* is transformed when the two are combined with logical connectives:

- It is raining **and** I am indoors ($P \wedge Q$)
- **If** it is raining, **then** I am indoors ($P \rightarrow Q$)
- **If** I am indoors, **then** it is raining ($Q \rightarrow P$)
- I am indoors **if and only if** it is raining ($P \leftrightarrow Q$)
- It is **not** raining ($\neg P$)

For statement $P = \text{It is raining}$ and $Q = \text{I am indoors}$.

It is also common to consider the *always true* formula and the *always false* formula to be connective:

- True formula (T , 1, Vpq , or T)
- False formula (\perp , 0, Opq , or F)

History of notations

- Negation: the symbol \neg appeared in Heyting in 1929.^{[1][2]} (compare to Frege's symbol $\neg\neg$ A in his *Begriffsschrift*); the symbol \sim appeared in Russell in 1908;^[3] an alternative notation is to add an horizontal line on top of the formula, as in \overline{P} ; another alternative notation is to use a prime symbol as in P' .
- Conjunction: the symbol \wedge appeared in Heyting in 1929 (compare to Peano's use of the set-theoretic notation of intersection \cap ^[4]); $\&$ appeared at least in Schönfinkel in 1924.^[5] comes from Boole's interpretation of logic as an elementary algebra.
- Disjunction: the symbol \vee appeared in Russell in 1908 (compare to Peano's use of the set-theoretic notation of union \cup); the symbol $+$ is also used, in spite of the ambiguity coming from the fact that the $+$ of ordinary elementary algebra is an exclusive or when interpreted logically in a two-element ring; punctually in the history a $+$ together with a dot in the lower right corner has been used by Peirce.^[6]
- Implication: the symbol \rightarrow can be seen in Hilbert in 1917;^[7] \supset was used by Russell in 1908 (compare to Peano's inverted C notation); \Rightarrow was used in Vax.^[8]
- Biconditional: the symbol \equiv was used at least by Russell in 1908; \leftrightarrow was used at least by Tarski in 1940;^[9] \Leftrightarrow was used in Vax; other symbols appeared punctually in the history such as $\supset C$ in Gentzen,^[10] \sim in Schönfinkel or $\supset\supset$ in Chazal.^[11]
- True: the symbol 1 comes from Boole's interpretation of logic as an elementary algebra over the two-element Boolean algebra; other notations include \bigwedge to be found in Peano.
- False: the symbol 0 comes also from Boole's interpretation of logic as a ring; other notations include \bigvee to be found in Peano.

Some authors used letters for connectives at some time of the history: **u.** for conjunction (German's "und" for "and") and **o.** for disjunction (German's "oder" for "or") in earlier works by Hilbert (1904); **Np** for negation, **Kpq** for conjunction, **Apq** for disjunction, **Cpq** for implication, **Epq** for biconditional in Łukasiewicz (1929).^[12]

Redundancy

Such logical connective as converse implication \leftarrow is actually the same as material conditional with swapped arguments, so the symbol for converse implication is redundant. In some logical calculi (notably, in classical logic) certain essentially different compound statements are logically equivalent. Less trivial example of a redundancy is a classical equivalence between $\neg P \vee Q$ and $P \rightarrow Q$. Therefore, a classical-based logical system does not need the conditional operator " \rightarrow " if " \neg " (not) and " \vee " (or) are already in use, or may use the " \rightarrow " only as a syntactic sugar for a compound having one negation and one disjunction.

There are sixteen Boolean functions associating the input truth values P and Q with four-digit binary outputs. These correspond to possible choices of binary logical connectives for classical logic. Different implementation of classical logic can choose different functionally complete subsets of connectives.

One approach is to choose a *minimal* set, and define other connectives by some logical form, like in the example with material conditional above. The following are the minimal functionally complete sets of operators in classical logic whose arities do not exceed 2:

One element

$$\{\uparrow\}, \{\downarrow\}.$$

Two elements

$$\{\vee, \neg\}, \{\wedge, \neg\}, \{\rightarrow, \neg\}, \{\leftarrow, \neg\}, \{\rightarrow, \perp\}, \{\leftarrow, \perp\}, \{\rightarrow, \leftrightarrow\}, \{\leftarrow, \leftrightarrow\}, \{\rightarrow, \Rightarrow\}, \{\rightarrow, \Leftarrow\}, \{\leftarrow, \Rightarrow\}, \{\leftarrow, \Leftarrow\}, \{\Rightarrow, \neg\}, \{\Leftarrow, \neg\}, \{\Rightarrow, \top\}, \{\Leftarrow, \top\}, \{\Rightarrow, \leftrightarrow\}, \{\Leftarrow, \leftrightarrow\}.$$

Three elements

$$\{\vee, \leftrightarrow, \perp\}, \{\vee, \leftrightarrow, \leftrightarrow\}, \{\vee, \leftrightarrow, \top\}, \{\wedge, \leftrightarrow, \perp\}, \{\wedge, \leftrightarrow, \leftrightarrow\}, \{\wedge, \leftrightarrow, \top\}.$$

See more details about functional completeness in classical logic at Truth function#Functional completeness.

Another approach is to use on equal rights connectives of a certain convenient and functionally complete, but *not minimal* set. This approach requires more propositional axioms and each equivalence between logical forms must be either an axiom or provable as a theorem.

But intuitionistic logic has the situation more complicated. Of its five connectives $\{\wedge, \vee, \rightarrow, \neg, \perp\}$ only negation \neg has to be reduced to other connectives (see details). Neither of conjunction, disjunction and material conditional has an equivalent form constructed of other four logical connectives.

Properties

Some logical connectives possess properties which may be expressed in the theorems containing the connective. Some of those properties that a logical connective may have are:

- **Associativity:** Within an expression containing two or more of the same associative connectives in a row, the order of the operations does not matter as long as the sequence of the operands is not changed.
- **Commutativity:** The operands of the connective may be swapped preserving logical equivalence to the original expression.
- **Distributivity:** A connective denoted by \cdot distributes over another connective denoted by $+$, if $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all operands a, b, c .
- **Idempotence:** Whenever the operands of the operation are the same, the compound is logically equivalent to the operand.
- **Absorption:** A pair of connectives \wedge, \vee satisfies the absorption law if $a \wedge (a \vee b) = a$ for all operands a, b .
- **Monotonicity:** If $f(a_1, \dots, a_n) \leq f(b_1, \dots, b_n)$ for all $a_1, \dots, a_n, b_1, \dots, b_n \in \{0,1\}$ such that $a_1 \leq b_1, a_2 \leq b_2, \dots, a_n \leq b_n$. E.g., $\vee, \wedge, \top, \perp$.

- **Affinity:** Each variable always makes a difference in the truth-value of the operation or it never makes a difference. E.g., \neg , \leftrightarrow , $\leftrightarrow\leftrightarrow$, \top , \perp .
- **Duality:** To read the truth-value assignments for the operation from top to bottom on its truth table is the same as taking the complement of reading the table of the same or another connective from bottom to top. Without resorting to truth tables it may be formulated as $\tilde{g}(\neg a_1, \dots, \neg a_n) = \neg g(a_1, \dots, a_n)$. E.g., \neg .
- **Truth-preserving:** The compound all those argument are tautologies is a tautology itself. E.g., \vee , \wedge , \top , \rightarrow , \leftrightarrow , \subset . (see validity)
- **Falsehood-preserving:** The compound all those argument are contradictions is a contradiction itself. E.g., \vee , \wedge , \leftrightarrow , \perp , \subset , $\not\subset$. (see validity)
- **Involutivity** (for unary connectives): $f(f(a)) = a$. E.g. negation in classical logic.

For classical and intuitionistic logic, the " $=$ " symbol means that corresponding implications " $\dots \rightarrow \dots$ " and " $\dots \leftarrow \dots$ " for logical compounds can be both proved as theorems, and the " \leq " symbol means that " $\dots \rightarrow \dots$ " for logical compounds is a consequence of corresponding " $\dots \rightarrow \dots$ " connectives for propositional variables. Some of many-valued logics may have incompatible definitions of equivalence and order (entailment).

Both conjunction and disjunction are associative, commutative and idempotent in classical logic, most varieties of many-valued logic and intuitionistic logic. The same is true about distributivity of conjunction over disjunction and disjunction over conjunction, as well as for the absorption law.

In classical logic and some varieties of many-valued logic, conjunction and disjunction are dual, and negation is self-dual, the latter is also self-dual in intuitionistic logic.

Order of precedence

As a way of reducing the number of necessary parentheses, one may introduce precedence rules: \neg has higher precedence than \wedge , \wedge higher than \vee , and \vee higher than \rightarrow . So for example, $P \vee Q \wedge \neg R \rightarrow S$ is short for $(P \vee (Q \wedge (\neg R))) \rightarrow S$.

Here is a table that shows a commonly used precedence of logical operators.

Operator	Precedence
\neg	1
\wedge	2
\vee	3
\rightarrow	4
\leftrightarrow	5

The order of precedence determines which connective is the "main connective" when interpreting a non-atomic formula.

Computer science

Truth-functional approach to logical operators is implemented as logic gates in digital circuits. Practically all digital circuits (the major exception is DRAM) are built up from NAND, NOR, NOT, and transmission gates; see more details in Truth function#Computer science. Logical operators over bit vectors (corresponding to finite Boolean algebras) are bitwise operations.

But not every usage of a logical connective in computer programming has a Boolean semantic. For example, lazy evaluation is sometimes implemented for $P \wedge Q$ and $P \vee Q$, so these connectives are not commutative if some of expressions P, Q has side effects. Also, a conditional, which in some sense corresponds to the material conditional connective, is essentially non-Boolean because for `if (P) then Q;` the consequent Q is not executed if the antecedent P is false (although a compound as a whole is successful \approx "true" in such case). This is closer to intuitionist and constructivist views on the material conditional, rather than to classical logic's ones.

Notes

- [1] Heyting (1929) *Die formalen Regeln der intuitionistischen Logik*.
- [2] Denis Roegel (2002), *Petit panorama des notations logiques du 20e siècle* (<http://www.loria.fr/~roegel/cours/symboles-logiques.pdf>) (see chart on page 2).
- [3] Russell (1908) *Mathematical logic as based on the theory of types* (American Journal of Mathematics 30, p222–262, also in From Frege to Gödel edited by van Heijenoort).
- [4] Peano (1889) *Arithmetices principia, nova methodo exposita*.
- [5] Schönfinkel (1924) *Über die Bausteine der mathematischen Logik*, translated as *On the building blocks of mathematical logic* in From Frege to Gödel edited by van Heijenoort.
- [6] Peirce (1867) *On an improvement in Boole's calculus of logic*.
- [7] Hilbert (1917/1918) *Prinzipien der Mathematik* (Bernays' course notes).
- [8] Vax (1982) *Lexique logique*, Presses Universitaires de France.
- [9] Tarski (1940) *Introduction to logic and to the methodology of deductive sciences*.
- [10] Gentzen (1934) *Untersuchungen über das logische Schließen*.
- [11] Chazal (1996) : *Éléments de logique formelle*.
- [12] See Roegel

References

- Bocheński, Józef Maria (1959), *A Précis of Mathematical Logic*, translated from the French and German editions by Otto Bird, D. Reidel, Dordrecht, South Holland.
- Enderton, Herbert (2001), *A Mathematical Introduction to Logic* (2nd ed.), Boston, MA: Academic Press, ISBN 978-0-12-238452-3
- Gamut, L.T.F (1991), "Chapter 2", *Logic, Language and Meaning 1*, University of Chicago Press, pp. 54–64, OCLC 21372380 (<http://www.worldcat.org/oclc/21372380>)

Further reading

- Lloyd Humberstone (2011). *The Connectives*. MIT Press. ISBN 978-0-262-01654-4.

External links

- Hazewinkel, Michiel, ed. (2001), "Propositional connective" (<http://www.encyclopediaofmath.org/index.php?title=p/p075490>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Lloyd Humberstone (2010), " Sentence Connectives in Formal Logic (<http://plato.stanford.edu/entries/connectives-logic/>)", Stanford Encyclopedia of Philosophy (An abstract algebraic logic approach to connectives.)
- John MacFarlane (2005), " Logical constants (<http://plato.stanford.edu/entries/logical-constants/>)", Stanford Encyclopedia of Philosophy.

Necessity and sufficiency

In logic, **necessity** and **sufficiency** are implicational relationships between statements. The assertion that one statement is a *necessary and sufficient* condition of another means that the former statement is true if and only if the latter is true.

Definitions

A true **necessary** condition in a conditional statement makes the statement true. In formal terms, a consequent N is a necessary condition for an antecedent S , in the conditional statement, " N if S ", " N is implied by S ", or $N \Leftarrow S$. In common words, we would also say " N is weaker than S " or " S cannot occur without N ". For example, it is necessary to be Named, to be called "Socrates".

A true **sufficient** condition in a conditional statement ties the statement's truth to its consequent. In formal terms, an antecedent S is a sufficient condition for a consequent N , in the conditional statement, "if S , then N ", " S implies N ", or $S \Rightarrow N$. In common words, we would also say " S is stronger than N " or " S guarantees N ". For example, "Socrates" suffices for a Name.

S implies N

S	N	$S \Rightarrow N$
T	T	T
T	F	F
F	T	T
F	F	T

Necessity

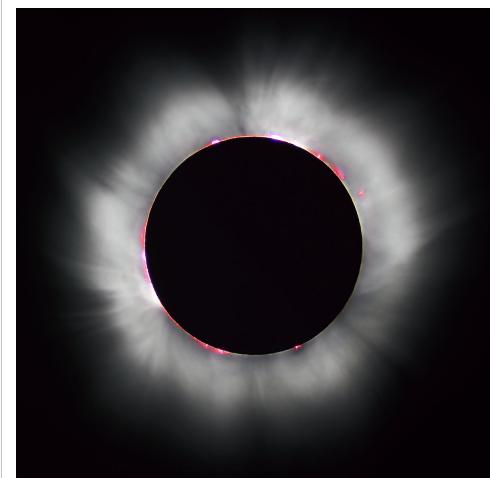
The assertion that Q is necessary for P is colloquially equivalent to " P cannot be true unless Q is true," or "if Q is false then P is false." By contraposition, this is the same thing as "whenever P is true, so is Q ". The logical relation between them is expressed as "If P then Q " and denoted " $P \Rightarrow Q$ " (P implies Q), and may also be expressed as any of " Q , if P "; " Q whenever P "; and " Q when P ." One often finds, in mathematical prose for instance, several necessary conditions that, taken together, constitute a sufficient condition, as shown in Example 5.

Example 1: In order for it to be true that "John is a bachelor," it is necessary that it be also true that he is

1. unmarried
2. male
3. adult

since to state "John is a bachelor" implies John has each of those three additional predicates.

Example 2: For the whole numbers greater than two, being odd is necessary to being prime, since two is the only whole number that is both even and prime.



The sun being above the horizon is a necessary condition for direct sunlight; but it is not a sufficient condition as something else may be casting a shadow, e.g. in the case of an eclipse.

Example 3: Consider thunder, in the technical sense, the acoustic quality demonstrated by the shock wave that inevitably results from any lightning bolt in the atmosphere. It may fairly be said that thunder is necessary for lightning, since lightning cannot occur without thunder, too, occurring. That is, if lightning does occur, then there is thunder.

Example 4: Being at least 30 years old is necessary of serving in the U.S. Senate. If you are under 30 years old then it is impossible for you to be a senator. That is, if you are a senator, it follows that you are at least 30 years old.

Example 5: In algebra, in order for some set S together with an operation \star to form a group, it is necessary that \star be associative. It is also necessary that S include a special element e such that for every x in S it is the case that $e \star x$ and $x \star e$ both equal x . It is also necessary that for every x in S there exist a corresponding element x'' such that both $x \star x''$ and $x'' \star x$ equal the special element e . None of these three necessary conditions by itself is sufficient, but the conjunction of the three is.

Sufficiency

To say that P is sufficient for Q is to say that, in and of itself, knowing P to be true is adequate grounds to conclude that Q is true. (It is to say, at the same time, that knowing P not to be true does not, in and of itself, provide adequate grounds to conclude that Q is not true, either.) The logical relation is expressed as "If P then Q " or " $P \Rightarrow Q$," and may also be expressed as " P implies Q ." Several sufficient conditions may, taken together, constitute a single necessary condition, as illustrated in example 5.

Example 1: Stating that "John is a bachelor" implies that John is male. So knowing that it is true that John is a bachelor is sufficient to know that he is a male.

Example 2: A number's being divisible by 4 is sufficient (but not necessary) for its being even, but being divisible by 2 is both sufficient and necessary.

Example 3: An occurrence of thunder is a sufficient condition for the occurrence of lightning in the sense that hearing thunder, and unambiguously recognizing it as such, justifies concluding that there has been a lightning bolt.

Example 4: A U.S. president's signing a bill that Congress passed is sufficient to make the bill law. Note that the case whereby the president did not sign the bill, e.g. through exercising a presidential veto, does not mean that the bill has not become law (it could still have become law through a congressional override).

Example 5: That the center of a playing card should be marked with a single large spade (\spadesuit) is sufficient for the card to be an ace. Three other sufficient conditions are that the center of the card be marked with a diamond (\diamondsuit), heart (\heartsuit), or club (\clubsuit), respectively. None of these conditions is necessary to the card's being an ace, but their disjunction is, since no card can be an ace without fulfilling at least (in fact, exactly) one of the conditions.

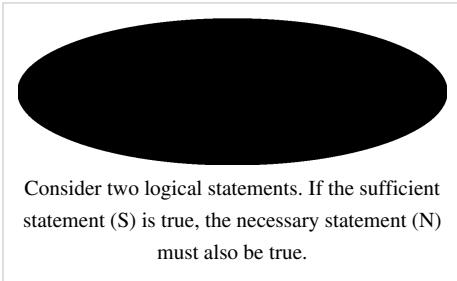


That a train runs on schedule can be a sufficient condition for arriving on time (if one boards the train and it departs on time, then one will arrive on time); but it is not always a necessary condition, since there are other ways to travel (if the train does not run to time, one could still arrive on time through other means of transport).

Relationship between necessity and sufficiency

A condition can be either necessary or sufficient without being the other. For instance, *being a mammal* (N) is necessary but not sufficient to *being human* (S), and that a number x is *rational* (S) is sufficient but not necessary to x 's *being a real number* (N) (since there are real numbers that are not rational).

A condition can be both necessary and sufficient. For example, at present, "today is the Fourth of July" is a necessary and sufficient condition for "today is Independence Day in the United States." Similarly, a necessary and sufficient condition for invertibility of a matrix M is that M has a nonzero determinant.



Consider two logical statements. If the sufficient statement (S) is true, the necessary statement (N) must also be true.

Mathematically speaking, necessity and sufficiency are dual to one another. For any statements S and N , the assertion that " N is necessary for S " is equivalent to the assertion that " S is sufficient for N ." Another facet of this duality is that, as illustrated above, conjunctions (using "and") of necessary conditions may achieve sufficiency, while disjunctions (using "or") of sufficient conditions may achieve necessity. For a third facet, identify every mathematical predicate N with the set $T(N)$ of objects, events, or statements for which N holds true; then asserting the necessity of N for S is equivalent to claiming that $T(N)$ is a superset of $T(S)$, while asserting the sufficiency of S for N is equivalent to claiming that $T(S)$ is a subset of $T(N)$.

Simultaneous necessity and sufficiency

To say that P is necessary and sufficient for Q is to say two things, that P is necessary for Q and that P is sufficient for Q . Of course, it may instead be understood to say *a different* two things, namely that each of P and Q is necessary for the other. And it may be understood in a third equivalent way: as saying that each is sufficient for the other. One may summarize any—and thus all—of these cases by the statement " P if and only if Q ," which is denoted by $P \Leftrightarrow Q$.

For example, in graph theory a graph G is called bipartite if it is possible to assign to each of its vertices the color *black* or *white* in such a way that every edge of G has one endpoint of each color. And for any graph to be bipartite, it is a necessary and sufficient condition that it contain no odd-length cycles. Thus, discovering whether a graph has any odd cycles tells one whether it is bipartite and vice versa. A philosopher^[1] might characterize this state of affairs thus: "Although the concepts of bipartiteness and absence of odd cycles differ in intension, they have identical extension."^[2]

References

- [1] Stanford University primer, 2006 (<http://plato.stanford.edu/entries/logic-intensional/>)
- [2] "Meanings, in this sense, are often called *intensions*, and things designated, *extensions*. Contexts in which extension is all that matters are, naturally, called *extensional*, while contexts in which extension is not enough are *intensional*. Mathematics is typically extensional throughout." Stanford University primer, 2006 (<http://plato.stanford.edu/entries/logic-intensional/>)

External links

- Critical thinking web tutorial: *Necessary and Sufficient Conditions* (<http://philosophy.hku.hk/think/meaning/nsc.php>)
- Simon Fraser University: Concepts with examples (<http://www.sfu.ca/~swartz/conditions1.htm>)

Propositional calculus

In mathematical logic, a **propositional calculus** or **logic** (also called **sentential calculus** or **sentential logic**) is a formal system in which formulas of a formal language may be interpreted to represent propositions. A system of inference rules and axioms allows certain formulas to be derived. These derived formulas are called theorems and may be interpreted to be true propositions. Such a constructed sequence of formulas is known as a *derivation* or *proof* and the last formula of the sequence is the theorem. The derivation may be interpreted as proof of the proposition represented by the theorem.

Usually in **Truth-functional propositional logic**, formulas are interpreted as having either a truth value of *true* or a truth value of *false*. Wikipedia:Please clarify Truth-functional propositional logic and systems isomorphic to it, are considered to be **zeroth-order logic**.

History

Although propositional logic (which is interchangeable with propositional calculus) had been hinted by earlier philosophers, it was developed into a formal logic by Chrysippus^[1] and expanded by the Stoics. The logic was focused on propositions. This advancement was different from the traditional syllogistic logic which was focused on terms. However, later in antiquity, the propositional logic developed by the Stoics was no longer understood. Consequently, the system was essentially reinvented by Peter Abelard.

Propositional logic was eventually refined using symbolic logic. Gottfried Leibniz has been credited with being the founder of symbolic logic for his work with the calculus ratiocinator. Although his work was the first of its kind, it was unknown to the larger logical community. Consequently, many of the advances achieved by Leibniz were reacheived by logicians like George Boole and Augustus De Morgan completely independent of Leibniz.^[2]

Just as propositional logic can be considered an advancement from the earlier syllogistic logic, Gottlob Frege's predicate logic was an advancement from the earlier propositional logic. Predicate logic has been described to be combining "the distinctive features of syllogistic logic and propositional logic." Consequently, it ushered a new era in the history of logic. However, advances in propositional logic were still made after Frege. These include Natural Deduction, Truth-Trees and Truth-Tables. Natural deduction was invented by Gerhard Gentzen and Jan Łukasiewicz. Truth-Trees were invented by Evert Willem Beth.^[3] The invention of truth-tables, however, is of controversial attribution.

The ideas preceding truth tables have been found in both Frege^[4] and Bertrand Russell^[5] whereas the actual 'tabular structure' (i.e. being formed in a table format) is generally credited to either Ludwig Wittgenstein, Emil Post or both (independently of one another). Besides Frege and Russell, others credited for having preceding ideas of truth-tables include Philo, Boole, Charles Sanders Peirce, and Ernst Schröder. And besides Post and Wittgenstein, others credited with the tabular structure include Łukasiewicz, Schröder, Alfred North Whitehead, William Stanley Jevons, John Venn, and Clarence Irving Lewis. Ultimately, some, like John Shosky, have concluded "It is far from clear that any one person should be given the title of 'inventor' of truth-tables."

Terminology

In general terms, a calculus is a formal system that consists of a set of syntactic expressions (*well-formed formulae* or *wffs*), a distinguished subset of these expressions (axioms), plus a set of formal rules that define a specific binary relation, intended to be interpreted to be logical equivalence, on the space of expressions.

When the formal system is intended to be a logical system, the expressions are meant to be interpreted to be statements, and the rules, known to be *inference rules*, are typically intended to be truth-preserving. In this setting, the rules (which may include axioms) can then be used to derive ("infer") formulae representing true statements from given formulae representing true statements.

The set of axioms may be empty, a nonempty finite set, a countably infinite set, or be given by axiom schemata. A formal grammar recursively defines the expressions and well-formed formulae (wffs) of the language. In addition a semantics may be given which defines truth and valuations (or interpretations).

The language of a propositional calculus consists of

1. a set of primitive symbols, variously referred to be *atomic formulae*, *placeholders*, *proposition letters*, or *variables*, and
2. a set of operator symbols, variously interpreted to be *logical operators* or *logical connectives*.

A *well-formed formula* (wff) is any atomic formula, or any formula that can be built up from atomic formulae by means of operator symbols according to the rules of the grammar.

Mathematicians sometimes distinguish between propositional constants, propositional variables, and schemata. Propositional constants represent some particular proposition, while propositional variables range over the set of all atomic propositions. Schemata, however, range over all propositions. It is common to represent propositional constants by A , B , and C , propositional variables by P , Q , and R , and schematic letters are often Greek letters, most often φ , ψ , and χ .

Basic concepts

The following outlines a standard propositional calculus. Many different formulations exist which are all more or less equivalent but differ in the details of

1. their language, that is, the particular collection of primitive symbols and operator symbols,
2. the set of axioms, or distinguished formulae, and
3. the set of inference rules.

We may represent any given proposition with a letter which we call a propositional constant, analogous to representing a number by a letter in mathematics, for instance, $a = 5$. We require that all propositions have exactly one of two truth-values: true or false. To take an example, let P be the proposition that it is raining outside. This will be true if it is raining outside and false otherwise.

- We then define truth-functional operators, beginning with negation. We write $\neg P$ to represent the negation of P , which can be thought of to be the denial of P . In the example above, $\neg P$ expresses that it is not raining outside, or by a more standard reading: "It is not the case that it is raining outside." When P is true, $\neg P$ is false; and when P is false, $\neg P$ is true. $\neg\neg P$ always has the same truth-value like P .
- Conjunction is a truth-functional connective which forms a proposition out of two simpler propositions, for example, P and Q . The conjunction of P and Q is written $P \wedge Q$, and expresses that each are true. We read $P \wedge Q$ for " P and Q ". For any two propositions, there are four possible assignments of truth values:
 1. P is true and Q is true
 2. P is true and Q is false
 3. P is false and Q is true
 4. P is false and Q is false

The conjunction of P and Q is true in case 1 and is false otherwise. Where P is the proposition that it is raining outside and Q is the proposition that a cold-front is over Kansas, $P \wedge Q$ is true when it is raining outside and there is a cold-front over Kansas. If it is not raining outside, then $P \wedge Q$ is false; and if there is no cold-front over Kansas, then $P \wedge Q$ is false.

- Disjunction resembles conjunction in that it forms a proposition out of two simpler propositions. We write it $P \vee Q$, and it is read " P or Q ". It expresses that either P or Q is true. Thus, in the cases listed above, the disjunction of P and Q is true in all cases except 4. Using the example above, the disjunction expresses that it is either raining outside or there is a cold front over Kansas. (Note, this use of disjunction is supposed to resemble the use of the English word "or". However, it is most like the English inclusive "or", which can be used to express the truth of at least one of two propositions. It is not like the English exclusive "or", which expresses the truth of exactly one of two propositions. That is to say, the exclusive "or" is false when both P and Q are true (case 1). An example of the exclusive or is: You may have a bagel or a pastry, but not both. Often in natural language, given the appropriate context, the addendum "but not both" is omitted but implied. In mathematics, however, "or" is always inclusive or; if exclusive or is meant it will be specified, possibly by "xor".)
- Material conditional also joins two simpler propositions, and we write $P \rightarrow Q$, which is read "if P then Q ". The proposition to the left of the arrow is called the antecedent and the proposition to the right is called the consequent. (There is no such designation for conjunction or disjunction, since they are commutative operations.) It expresses that Q is true whenever P is true. Thus it is true in every case above except case 2, because this is the only case when P is true but Q is not. Using the example, if P then Q expresses that if it is raining outside then there is a cold-front over Kansas. The material conditional is often confused with physical causation. The material conditional, however, only relates two propositions by their truth-values—which is not the relation of cause and effect. It is contentious in the literature whether the material implication represents logical causation.
- Biconditional joins two simpler propositions, and we write $P \leftrightarrow Q$, which is read " P if and only if Q ". It expresses that P and Q have the same truth-value, thus P if and only if Q is true in cases 1 and 4, and false otherwise.

It is extremely helpful to look at the truth tables for these different operators, as well as the method of analytic tableaux.

Closure under operations

Propositional logic is closed under truth-functional connectives. That is to say, for any proposition φ , $\neg\varphi$ is also a proposition. Likewise, for any propositions φ and ψ , $\varphi \wedge \psi$ is a proposition, and similarly for disjunction, conditional, and biconditional. This implies that, for instance, $P \wedge Q$ is a proposition, and so it can be conjoined with another proposition. In order to represent this, we need to use parentheses to indicate which proposition is conjoined with which. For instance, $P \wedge Q \wedge R$ is not a well-formed formula, because we do not know if we are conjoining $P \wedge Q$ with R or if we are conjoining P with $Q \wedge R$. Thus we must write either $(P \wedge Q) \wedge R$ to represent the former, or $P \wedge (Q \wedge R)$ to represent the latter. By evaluating the truth conditions, we see that both expressions have the same truth conditions (will be true in the same cases), and moreover that any proposition formed by arbitrary conjunctions will have the same truth conditions, regardless of the location of the parentheses. This means that conjunction is associative, however, one should not assume that parentheses never serve a purpose. For instance, the sentence $P \wedge (Q \vee R)$ does not have the same truth conditions of $(P \wedge Q) \vee R$, so they are different sentences distinguished only by the parentheses. One can verify this by the truth-table method referenced above.
Note: For any arbitrary number of propositional constants, we can form a finite number of cases which list their possible truth-values. A simple way to generate this is by truth-tables, in which one writes P, Q, \dots, Z for any list of k propositional constants—that is to say, any list of propositional constants with k entries. Below this list, one writes 2^k rows, and below P one fills in the first half of the rows with true (or T) and the second half with false

(or F). Below Q one fills in one-quarter of the rows with T, then one-quarter with F, then one-quarter with T and the last quarter with F. The next column alternates between true and false for each eighth of the rows, then sixteenths, and so on, until the last propositional constant varies between T and F for each row. This will give a complete listing of cases or truth-value assignments possible for those propositional constants.

Argument

The propositional calculus then defines an *argument* to be a set of propositions. A valid argument is a set of propositions, the last of which follows from—or is implied by—the rest. All other arguments are invalid. The simplest valid argument is modus ponens, one instance of which is the following set of propositions:

$$\begin{array}{c} 1. \ P \rightarrow Q \\ 2. \ P \\ \hline \therefore Q \end{array}$$

This is a set of three propositions, each line is a proposition, and the last follows from the rest. The first two lines are called premises, and the last line the conclusion. We say that any proposition C follows from any set of propositions (P_1, \dots, P_n) , if C must be true whenever every member of the set (P_1, \dots, P_n) is true. In the argument above, for any P and Q , whenever $P \rightarrow Q$ and P are true, necessarily Q is true. Notice that, when P is true, we cannot consider cases 3 and 4 (from the truth table). When $P \rightarrow Q$ is true, we cannot consider case 2. This leaves only case 1, in which Q is also true. Thus Q is implied by the premises. This generalizes schematically. Thus, where φ and ψ may be any propositions at all,

$$\begin{array}{c} 1. \ \varphi \rightarrow \psi \\ 2. \ \varphi \\ \hline \therefore \psi \end{array}$$

Other argument forms are convenient, but not necessary. Given a complete set of axioms (see below for one such set), modus ponens is sufficient to prove all other argument forms in propositional logic, thus they may be considered to be a derivative. Note, this is not true of the extension of propositional logic to other logics like first-order logic. First-order logic requires at least one additional rule of inference in order to obtain completeness.

The significance of argument in formal logic is that one may obtain new truths from established truths. In the first example above, given the two premises, the truth of Q is not yet known or stated. After the argument is made, Q is deduced. In this way, we define a deduction system to be a set of all propositions that may be deduced from another set of propositions. For instance, given the set of propositions $A = \{P \vee Q, \neg Q \wedge R, (P \vee Q) \rightarrow R\}$, we can define a deduction system, Γ , which is the set of all propositions which follow from A . Reiteration is always assumed, so $P \vee Q, \neg Q \wedge R, (P \vee Q) \rightarrow R \in \Gamma$. Also, from the first element of A , last element, as well as modus ponens, R is a consequence, and so $R \in \Gamma$. Because we have not included sufficiently complete axioms, though, nothing else may be deduced. Thus, even though most deduction systems studied in propositional logic are able to deduce $(P \vee Q) \leftrightarrow (\neg P \rightarrow Q)$, this one is too weak to prove such a proposition.

Generic description of a propositional calculus

A **propositional calculus** is a formal system $\mathcal{L} = \mathcal{L}(A, \Omega, Z, I)$, where:

- The *alpha set* A is a finite set of elements called *proposition symbols* or *propositional variables*. Syntactically speaking, these are the most basic elements of the formal language \mathcal{L} , otherwise referred to as *atomic formulae* or *terminal elements*. In the examples to follow, the elements of A are typically the letters p, q, r , and so on.
- The *omega set* Ω is a finite set of elements called *operator symbols* or *logical connectives*. The set Ω is partitioned into disjoint subsets as follows:

$$\Omega = \Omega_0 \cup \Omega_1 \cup \dots \cup \Omega_j \cup \dots \cup \Omega_m.$$

In this partition, Ω_j is the set of operator symbols of *arity* j .

In the more familiar propositional calculi, Ω is typically partitioned as follows:

$$\Omega_1 = \{\neg\},$$

$$\Omega_2 \subseteq \{\wedge, \vee, \rightarrow, \leftrightarrow\}.$$

A frequently adopted convention treats the constant logical values as operators of arity zero, thus:

$$\Omega_0 = \{0, 1\}.$$

Some writers use the tilde (\sim), or N , instead of \neg ; and some use the ampersand ($\&$), the prefixed K , or \cdot instead of \wedge . Notation varies even more for the set of logical values, with symbols like {false, true}, {F, T}, or $\{\perp, \top\}$ all being seen in various contexts instead of {0, 1}.

- The *zeta set* Z is a finite set of *transformation rules* that are called *inference rules* when they acquire logical applications.
- The *iota set* I is a finite set of *initial points* that are called *axioms* when they receive logical interpretations.

The *language* of \mathcal{L} , also known as its set of *formulae*, *well-formed formulas* or *wffs*, is inductively defined by the following rules:

- Base: Any element of the alpha set A is a formula of \mathcal{L} .
- If p_1, p_2, \dots, p_j are formulae and f is in Ω_j , then $(f(p_1, p_2, \dots, p_j))$ is a formula.
- Closed: Nothing else is a formula of \mathcal{L} .

Repeated applications of these rules permits the construction of complex formulae. For example:

- By rule 1, p is a formula.
- By rule 2, $\neg p$ is a formula.
- By rule 1, q is a formula.
- By rule 2, $(\neg p \vee q)$ is a formula.

Example 1. Simple axiom system

Let $\mathcal{L}_1 = \mathcal{L}(A, \Omega, Z, I)$, where A, Ω, Z, I are defined as follows:

- The alpha set A , is a finite set of symbols that is large enough to supply the needs of a given discussion, for example:

$$A = \{p, q, r, s, t, u\}.$$

- Of the three connectives for conjunction, disjunction, and implication (\wedge, \vee , and \rightarrow), one can be taken as primitive and the other two can be defined in terms of it and negation (\neg).^[6] Indeed, all of the logical connectives can be defined in terms of a sole sufficient operator. The biconditional (\leftrightarrow) can of course be defined in terms of conjunction and implication, with $a \leftrightarrow b$ defined as $(a \rightarrow b) \wedge (b \rightarrow a)$.

Adopting negation and implication as the two primitive operations of a propositional calculus is tantamount to having the omega set $\Omega = \Omega_1 \cup \Omega_2$ partition as follows:

$$\Omega_1 = \{\neg\},$$

$$\Omega_2 = \{\rightarrow\}.$$

- An axiom system discovered by Jan Łukasiewicz formulates a propositional calculus in this language as follows. The axioms are all substitution instances of:
 - $(p \rightarrow (q \rightarrow p))$
 - $((p \rightarrow (q \rightarrow r)) \rightarrow ((p \rightarrow q) \rightarrow (p \rightarrow r)))$
 - $((\neg p \rightarrow \neg q) \rightarrow (q \rightarrow p))$
- The rule of inference is modus ponens (i.e., from p and $(p \rightarrow q)$, infer q). Then $a \vee b$ is defined as $\neg a \rightarrow b$, and $a \wedge b$ is defined as $\neg(a \rightarrow \neg b)$.

Example 2. Natural deduction system

Let $\mathcal{L}_2 = \mathcal{L}(A, \Omega, Z, I)$, where A , Ω , Z , I are defined as follows:

- The alpha set A , is a finite set of symbols that is large enough to supply the needs of a given discussion, for example:

$$A = \{p, q, r, s, t, u\}.$$

- The omega set $\Omega = \Omega_1 \cup \Omega_2$ partitions as follows:

$$\Omega_1 = \{\neg\},$$

$$\Omega_2 = \{\wedge, \vee, \rightarrow, \leftrightarrow\}.$$

In the following example of a propositional calculus, the transformation rules are intended to be interpreted as the inference rules of a so-called *natural deduction system*. The particular system presented here has no initial points, which means that its interpretation for logical applications derives its theorems from an empty axiom set.

- The set of initial points is empty, that is, $I = \emptyset$.
- The set of transformation rules, Z , is described as follows:

Our propositional calculus has ten inference rules. These rules allow us to derive other true formulae given a set of formulae that are assumed to be true. The first nine simply state that we can infer certain wffs from other wffs. The last rule however uses hypothetical reasoning in the sense that in the premise of the rule we temporarily assume an (unproven) hypothesis to be part of the set of inferred formulae to see if we can infer a certain other formula. Since the first nine rules don't do this they are usually described as *non-hypothetical* rules, and the last one as a *hypothetical* rule.

In describing the transformation rules, we may introduce a metalanguage symbol \vdash . It is basically a convenient shorthand for saying "infer that". The format is $\Gamma \vdash \psi$, in which Γ is a (possibly empty) set of formulae called premises, and ψ is a formula called conclusion. The transformation rule $\Gamma \vdash \psi$ means that if every proposition in Γ is a theorem (or has the same truth value as the axioms), then ψ is also a theorem. Note that considering the following rule Conjunction introduction, we will know whenever Γ has more than one formula, we can always safely reduce it into one formula using conjunction. So for short, from that time on we may represent Γ as one formula instead of a set. Another omission for convenience is when Γ is an empty set, in which case Γ may not appear.

Negation introduction

From $(p \rightarrow q)$ and $(p \rightarrow \neg q)$, infer $\neg p$.

That is, $\{(p \rightarrow q), (p \rightarrow \neg q)\} \vdash \neg p$.

Negation elimination

From $\neg p$, infer $(p \rightarrow r)$.

That is, $\{\neg p\} \vdash (p \rightarrow r)$.

Double negative elimination

From $\neg\neg p$, infer p .

That is, $\neg\neg p \vdash p$.

Conjunction introduction

From p and q , infer $(p \wedge q)$.

That is, $\{p, q\} \vdash (p \wedge q)$.

Conjunction elimination

From $(p \wedge q)$, infer p .

From $(p \wedge q)$, infer q .

That is, $(p \wedge q) \vdash p$ and $(p \wedge q) \vdash q$.

Disjunction introduction

From p , infer $(p \vee q)$.

From q , infer $(p \vee q)$.

That is, $p \vdash (p \vee q)$ and $q \vdash (p \vee q)$.

Disjunction elimination

From $(p \vee q)$ and $(p \rightarrow r)$ and $(q \rightarrow r)$, infer r .

That is, $\{p \vee q, p \rightarrow r, q \rightarrow r\} \vdash r$.

Biconditional introduction

From $(p \rightarrow q)$ and $(q \rightarrow p)$, infer $(p \leftrightarrow q)$.

That is, $\{p \rightarrow q, q \rightarrow p\} \vdash (p \leftrightarrow q)$.

Biconditional elimination

From $(p \leftrightarrow q)$, infer $(p \rightarrow q)$.

From $(p \leftrightarrow q)$, infer $(q \rightarrow p)$.

That is, $(p \leftrightarrow q) \vdash (p \rightarrow q)$ and $(p \leftrightarrow q) \vdash (q \rightarrow p)$.

Modus ponens (conditional elimination)

From p and $(p \rightarrow q)$, infer q .

That is, $\{p, p \rightarrow q\} \vdash q$.

Conditional proof (conditional introduction)

From [accepting p allows a proof of q], infer $(p \rightarrow q)$.

That is, $(p \vdash q) \vdash (\vdash (p \rightarrow q))$.

Basic and derived argument forms

Basic and Derived Argument Forms		
Name	Sequent	Description
Modus Ponens	$((p \rightarrow q) \wedge p) \vdash q$	If p then q ; p ; therefore q
Modus Tollens	$((p \rightarrow q) \wedge \neg q) \vdash \neg p$	If p then q ; not q ; therefore not p
Hypothetical Syllogism	$((p \rightarrow q) \wedge (q \rightarrow r)) \vdash (p \rightarrow r)$	If p then q ; if q then r ; therefore, if p then r
Disjunctive Syllogism	$((p \vee q) \wedge \neg p) \vdash q$	Either p or q , or both; not p ; therefore, q
Constructive Dilemma	$((p \rightarrow q) \wedge (r \rightarrow s) \wedge (p \vee r)) \vdash (q \vee s)$	If p then q ; and if r then s ; but p or r ; therefore q or s
Destructive Dilemma	$((p \rightarrow q) \wedge (r \rightarrow s) \wedge (\neg q \vee \neg s)) \vdash (\neg p \vee \neg r)$	If p then q ; and if r then s ; but not q or not s ; therefore not p or not r
Bidirectional Dilemma	$((p \rightarrow q) \wedge (r \rightarrow s) \wedge (p \vee \neg s)) \vdash (q \vee \neg r)$	If p then q ; and if r then s ; but p or not s ; therefore q or not r
Simplification	$(p \wedge q) \vdash p$	p and q are true; therefore p is true
Conjunction	$p, q \vdash (p \wedge q)$	p and q are true separately; therefore they are true conjointly
Addition	$p \vdash (p \vee q)$	p is true; therefore the disjunction (p or q) is true
Composition	$((p \rightarrow q) \wedge (p \rightarrow r)) \vdash (p \rightarrow (q \wedge r))$	If p then q ; and if p then r ; therefore if p is true then q and r are true
De Morgan's Theorem (1)	$\neg(p \wedge q) \vdash (\neg p \vee \neg q)$	The negation of (p and q) is equiv. to (not p or not q)
De Morgan's Theorem (2)	$\neg(p \vee q) \vdash (\neg p \wedge \neg q)$	The negation of (p or q) is equiv. to (not p and not q)
Commutation (1)	$(p \vee q) \vdash (q \vee p)$	(p or q) is equiv. to (q or p)
Commutation (2)	$(p \wedge q) \vdash (q \wedge p)$	(p and q) is equiv. to (q and p)
Commutation (3)	$(p \leftrightarrow q) \vdash (q \leftrightarrow p)$	(p is equiv. to q) is equiv. to (q is equiv. to p)
Association (1)	$(p \vee (q \vee r)) \vdash ((p \vee q) \vee r)$	p or (q or r) is equiv. to (p or q) or r
Association (2)	$(p \wedge (q \wedge r)) \vdash ((p \wedge q) \wedge r)$	p and (q and r) is equiv. to (p and q) and r
Distribution (1)	$(p \wedge (q \vee r)) \vdash ((p \wedge q) \vee (p \wedge r))$	p and (q or r) is equiv. to (p and q) or (p and r)
Distribution (2)	$(p \vee (q \wedge r)) \vdash ((p \vee q) \wedge (p \vee r))$	p or (q and r) is equiv. to (p or q) and (p or r)
Double Negation	$p \vdash \neg\neg p$	p is equivalent to the negation of not p
Transposition	$(p \rightarrow q) \vdash (\neg q \rightarrow \neg p)$	If p then q is equiv. to if not q then not p
Material Implication	$(p \rightarrow q) \vdash (\neg p \vee q)$	If p then q is equiv. to not p or q
Material Equivalence (1)	$(p \leftrightarrow q) \vdash ((p \rightarrow q) \wedge (q \rightarrow p))$	(p iff q) is equiv. to (if p is true then q is true) and (if q is true then p is true)
Material Equivalence (2)	$(p \leftrightarrow q) \vdash ((p \wedge q) \vee (\neg p \wedge \neg q))$	(p iff q) is equiv. to either (p and q are true) or (both p and q are false)
Material Equivalence (3)	$(p \leftrightarrow q) \vdash ((p \vee \neg q) \wedge (\neg p \vee q))$	(p iff q) is equiv. to., both (p or not q is true) and (not p or q is true)
Exportation	$((p \wedge q) \rightarrow r) \vdash (p \rightarrow (q \rightarrow r))$	from (if p and q are true then r is true) we can prove (if q is true then r is true, if p is true)
Importation	$(p \rightarrow (q \rightarrow r)) \vdash ((p \wedge q) \rightarrow r)$	If p then (if q then r) is equivalent to if p and q then r

Tautology (1)	$p \vdash (p \vee p)$	p is true is equiv. to p is true or \bar{p} is true
Tautology (2)	$p \vdash (p \wedge p)$	p is true is equiv. to p is true and \bar{p} is true
Tertium non datur (Law of Excluded Middle)	$\vdash (p \vee \neg p)$	p or not p is true
Law of Non-Contradiction	$\vdash \neg(p \wedge \neg p)$	p and not p is false, is a true statement

Proofs in propositional calculus

One of the main uses of a propositional calculus, when interpreted for logical applications, is to determine relations of logical equivalence between propositional formulae. These relationships are determined by means of the available transformation rules, sequences of which are called *derivations* or *proofs*.

In the discussion to follow, a proof is presented as a sequence of numbered lines, with each line consisting of a single formula followed by a *reason* or *justification* for introducing that formula. Each premise of the argument, that is, an assumption introduced as an hypothesis of the argument, is listed at the beginning of the sequence and is marked as a "premise" in lieu of other justification. The conclusion is listed on the last line. A proof is complete if every line follows from the previous ones by the correct application of a transformation rule. (For a contrasting approach, see proof-trees).

Example of a proof

- To be shown that $A \rightarrow A$.
- One possible proof of this (which, though valid, happens to contain more steps than are necessary) may be arranged as follows:

Example of a Proof		
Number	Formula	Reason
1	A	premise
2	$A \vee A$	From (1) by disjunction introduction
3	$(A \vee A) \wedge A$	From (1) and (2) by conjunction introduction
4	A	From (3) by conjunction elimination
5	$A \vdash A$	Summary of (1) through (4)
6	$\vdash A \rightarrow A$	From (5) by conditional proof

Interpret $A \vdash A$ as "Assuming A , infer A ". Read $\vdash A \rightarrow A$ as "Assuming nothing, infer that A implies A ", or "It is a tautology that A implies A ", or "It is always true that A implies A ".

Soundness and completeness of the rules

The crucial properties of this set of rules are that they are *sound* and *complete*. Informally this means that the rules are correct and that no other rules are required. These claims can be made more formal as follows.

We define a *truth assignment* as a function that maps propositional variables to **true** or **false**. Informally such a truth assignment can be understood as the description of a possible state of affairs (or possible world) where certain statements are true and others are not. The semantics of formulae can then be formalized by defining for which "state of affairs" they are considered to be true, which is what is done by the following definition.

We define when such a truth assignment A satisfies a certain wff with the following rules:

- A satisfies the propositional variable P if and only if $A(P) = \text{true}$

- A satisfies $\neg\phi$ if and only if A does not satisfy ϕ
- A satisfies $(\phi \wedge \psi)$ if and only if A satisfies both ϕ and ψ
- A satisfies $(\phi \vee \psi)$ if and only if A satisfies at least one of either ϕ or ψ
- A satisfies $(\phi \rightarrow \psi)$ if and only if it is not the case that A satisfies ϕ but not ψ
- A satisfies $(\phi \leftrightarrow \psi)$ if and only if A satisfies both ϕ and ψ or satisfies neither one of them

With this definition we can now formalize what it means for a formula ϕ to be implied by a certain set S of formulae. Informally this is true if in all worlds that are possible given the set of formulae S the formula ϕ also holds. This leads to the following formal definition: We say that a set S of wffs *semantically entails* (or *implies*) a certain wff ϕ if all truth assignments that satisfy all the formulae in S also satisfy ϕ .

Finally we define *syntactical entailment* such that ϕ is syntactically entailed by S if and only if we can derive it with the inference rules that were presented above in a finite number of steps. This allows us to formulate exactly what it means for the set of inference rules to be sound and complete:

Soundness

If the set of wffs S syntactically entails wff ϕ then S semantically entails ϕ

Completeness

If the set of wffs S semantically entails wff ϕ then S syntactically entails ϕ

For the above set of rules this is indeed the case.

Sketch of a soundness proof

(For most logical systems, this is the comparatively "simple" direction of proof)

Notational conventions: Let G be a variable ranging over sets of sentences. Let A , B , and C range over sentences. For " G syntactically entails A " we write " G proves A ". For " G semantically entails A " we write " G implies A ".

We want to show: (A) (G) (if G proves A , then G implies A).

We note that " G proves A " has an inductive definition, and that gives us the immediate resources for demonstrating claims of the form "If G proves A , then ...". So our proof proceeds by induction.

I. Basis. Show: If A is a member of G , then G implies A .

II. Basis. Show: If A is an axiom, then G implies A .

III. Inductive step (induction on n , the length of the proof):

- Assume for arbitrary G and A that if G proves A in n or fewer steps, then G implies A .
- For each possible application of a rule of inference at step $n + 1$, leading to a new theorem B , show that G implies B .

Notice that Basis Step II can be omitted for natural deduction systems because they have no axioms. When used, Step II involves showing that each of the axioms is a (semantic) logical truth.

The Basis step(s) demonstrate(s) that the simplest provable sentences from G are also implied by G , for any G .

(The is simple, since the semantic fact that a set implies any of its members, is also trivial.) The Inductive step will systematically cover all the further sentences that might be provable—by considering each case where we might reach a logical conclusion using an inference rule—and shows that if a new sentence is provable, it is also logically implied. (For example, we might have a rule telling us that from " A " we can derive " A or B ". In III.a We assume that if A is provable it is implied. We also know that if A is provable then " A or B " is provable. We have to show that then " A or B " too is implied. We do so by appeal to the semantic definition and the assumption we just made. A is provable from G , we assume. So it is also implied by G . So any semantic valuation making all of G true makes A true. But any valuation making A true makes " A or B " true, by the defined semantics for "or". So any valuation which makes all of G true makes " A or B " true. So " A or B " is

implied.) Generally, the Inductive step will consist of a lengthy but simple case-by-case analysis of all the rules of inference, showing that each "preserves" semantic implication.

By the definition of provability, there are no sentences provable other than by being a member of G , an axiom, or following by a rule; so if all of those are semantically implied, the deduction calculus is sound.

Sketch of completeness proof

(This is usually the much harder direction of proof.)

We adopt the same notational conventions as above.

We want to show: If G implies A , then G proves A . We proceed by contraposition: We show instead that if G does **not** prove A then G does **not** imply A .

I. G does not prove A . (Assumption)

II. If G does not prove A , then we can construct an (infinite) "Maximal Set", G^* , which is a superset of G and which also does not prove A .

1. Place an "ordering" on all the sentences in the language (e.g., shortest first, and equally long ones in extended alphabetical ordering), and number them E_1, E_2, \dots

2. Define a series G_n of sets (G_0, G_1, \dots) inductively:

- i. $G_0 = G$

- ii. If $G_k \cup \{E_{k+1}\}$ proves A , then $G_{k+1} = G_k$

- iii. If $G_k \cup \{E_{k+1}\}$ does **not** prove A , then $G_{k+1} = G_k \cup \{E_{k+1}\}$

3. Define G^* as the union of all the G_n . (That is, G^* is the set of all the sentences that are in any G_n .)

4. It can be easily shown that

- i. G^* contains (is a superset of) G (by (b.i));

- ii. G^* does not prove A (because if it proves A then some sentence was added to some G_n which caused it to prove ' A '; but this was ruled out by definition); and

- iii. G^* is a "Maximal Set" (with respect to A): If *any* more sentences whatever were added to G^* , it would prove A . (Because if it were possible to add any more sentences, they should have been added when they were encountered during the construction of the G_n , again by definition)

III. If G^* is a Maximal Set (wrt A), then it is "truth-like". This means that it contains the sentence " C " only if it does *not* contain the sentence not- C ; If it contains " C " and contains "If C then B " then it also contains " B "; and so forth.

IV. If G^* is truth-like there is a " G^* -Canonical" valuation of the language: one that makes every sentence in G^* true and everything outside G^* false while still obeying the laws of semantic composition in the language.

V. A G^* -canonical valuation will make our original set G all true, and make A false.

VI. If there is a valuation on which G are true and A is false, then G does not (semantically) imply A .

QED

Another outline for a completeness proof

If a formula is a tautology, then there is a truth table for it which shows that each valuation yields the value true for the formula. Consider such a valuation. By mathematical induction on the length of the subformulae, show that the truth or falsity of the subformula follows from the truth or falsity (as appropriate for the valuation) of each propositional variable in the subformula. Then combine the lines of the truth table together two at a time by using "(P is true implies S) implies ((P is false implies S) implies S)". Keep repeating this until all dependencies on propositional variables have been eliminated. The result is that we have proved the given tautology. Since every tautology is provable, the logic is complete.

Interpretation of a truth-functional propositional calculus

An **interpretation of a truth-functional propositional calculus** \mathcal{P} is an assignment to each propositional symbol of \mathcal{P} of one or the other (but not both) of the truth values truth (**T**) and falsity (**F**), and an assignment to the connective symbols of \mathcal{P} of their usual truth-functional meanings. An interpretation of a truth-functional propositional calculus may also be expressed in terms of truth tables.

For n distinct propositional symbols there are 2^n distinct possible interpretations. For any particular symbol a , for example, there are $2^1 = 2$ possible interpretations:

1. a is assigned **T**, or
2. a is assigned **F**.

For the pair a, b there are $2^2 = 4$ possible interpretations:

1. both are assigned **T**,
2. both are assigned **F**,
3. a is assigned **T** and b is assigned **F**, or
4. a is assigned **F** and b is assigned **T**.

Since \mathcal{P} has \aleph_0 , that is, denumerably many propositional symbols, there are $2^{\aleph_0} = \mathfrak{c}$, and therefore uncountably many distinct possible interpretations of \mathcal{P} .

Interpretation of a sentence of truth-functional propositional logic

If ϕ and ψ are formulae of \mathcal{P} and \mathcal{I} is an interpretation of \mathcal{P} then:

- A sentence of propositional logic is *true under an interpretation* \mathcal{I} iff \mathcal{I} assigns the truth value **T** to that sentence. If a sentence is true under an interpretation, then that interpretation is called a *model* of that sentence.
- ϕ is *false under an interpretation* \mathcal{I} iff ϕ is not true under \mathcal{I} .
- A sentence of propositional logic is *logically valid* iff it is true under every interpretation

$\models \phi$ means that ϕ is logically valid

- A sentence ψ of propositional logic is a *semantic consequence* of a sentence ϕ iff there is no interpretation under which ϕ is true and ψ is false.
- A sentence of propositional logic is *consistent* iff it is true under at least one interpretation. It is inconsistent if it is not consistent.

Some consequences of these definitions:

- For any given interpretation a given formula is either true or false.
- No formula is both true and false under the same interpretation.
- ϕ is false for a given interpretation iff $\neg\phi$ is true for that interpretation; and ϕ is true under an interpretation iff $\neg\phi$ is false under that interpretation.
- If ϕ and $(\phi \rightarrow \psi)$ are both true under a given interpretation, then ψ is true under that interpretation.
- If $\models_P \phi$ and $\models_P (\phi \rightarrow \psi)$, then $\models_P \psi$.
- $\neg\phi$ is true under \mathcal{I} iff ϕ is not true under \mathcal{I} .
- $(\phi \rightarrow \psi)$ is true under \mathcal{I} iff either ϕ is not true under \mathcal{I} or ψ is true under \mathcal{I} .
- A sentence ψ of propositional logic is a semantic consequence of a sentence ϕ iff $(\phi \rightarrow \psi)$ is logically valid, that is, $\phi \models_P \psi$ iff $\models_P (\phi \rightarrow \psi)$.

Alternative calculus

It is possible to define another version of propositional calculus, which defines most of the syntax of the logical operators by means of axioms, and which uses only one inference rule.

Axioms

Let ϕ , χ and ψ stand for well-formed formulæ. (The wffs themselves would not contain any Greek letters, but only capital Roman letters, connective operators, and parentheses.) Then the axioms are as follows:

Axioms		
Name	Axiom Schema	Description
THEN-1	$\phi \rightarrow (\chi \rightarrow \phi)$	Add hypothesis χ , implication introduction
THEN-2	$(\phi \rightarrow (\chi \rightarrow \psi)) \rightarrow ((\phi \rightarrow \chi) \rightarrow (\phi \rightarrow \psi))$	Distribute hypothesis ϕ over implication
AND-1	$\phi \wedge \chi \rightarrow \phi$	Eliminate conjunction
AND-2	$\phi \wedge \chi \rightarrow \chi$	
AND-3	$\phi \rightarrow (\chi \rightarrow (\phi \wedge \chi))$	Introduce conjunction
OR-1	$\phi \rightarrow \phi \vee \chi$	Introduce disjunction
OR-2	$\chi \rightarrow \phi \vee \chi$	
OR-3	$(\phi \rightarrow \psi) \rightarrow ((\chi \rightarrow \psi) \rightarrow (\phi \vee \chi \rightarrow \psi))$	Eliminate disjunction
NOT-1	$(\phi \rightarrow \chi) \rightarrow ((\phi \rightarrow \neg \chi) \rightarrow \neg \phi)$	Introduce negation
NOT-2	$\phi \rightarrow (\neg \phi \rightarrow \chi)$	Eliminate negation
NOT-3	$\phi \vee \neg \phi$	Excluded middle, classical logic
IFF-1	$(\phi \leftrightarrow \chi) \rightarrow (\phi \rightarrow \chi)$	Eliminate equivalence
IFF-2	$(\phi \leftrightarrow \chi) \rightarrow (\chi \rightarrow \phi)$	
IFF-3	$(\phi \rightarrow \chi) \rightarrow ((\chi \rightarrow \phi) \rightarrow (\phi \leftrightarrow \chi))$	Introduce equivalence

- Axiom THEN-2 may be considered to be a "distributive property of implication with respect to implication."
- Axioms AND-1 and AND-2 correspond to "conjunction elimination". The relation between AND-1 and AND-2 reflects the commutativity of the conjunction operator.
- Axiom AND-3 corresponds to "conjunction introduction."
- Axioms OR-1 and OR-2 correspond to "disjunction introduction." The relation between OR-1 and OR-2 reflects the commutativity of the disjunction operator.
- Axiom NOT-1 corresponds to "reductio ad absurdum."
- Axiom NOT-2 says that "anything can be deduced from a contradiction."
- Axiom NOT-3 is called "tertium non datur" (Latin: "a third is not given") and reflects the semantic valuation of propositional formulae: a formula can have a truth-value of either true or false. There is no third truth-value, at least not in classical logic. Intuitionistic logicians do not accept the axiom NOT-3.

Inference rule

The inference rule is modus ponens:

$$\phi, \phi \rightarrow \chi \vdash \chi.$$

Meta-inference rule

Let a demonstration be represented by a sequence, with hypotheses to the left of the turnstile and the conclusion to the right of the turnstile. Then the deduction theorem can be stated as follows:

If the sequence

$$\phi_1, \phi_2, \dots, \phi_n, \chi \vdash \psi$$

has been demonstrated, then it is also possible to demonstrate the sequence

$$\phi_1, \phi_2, \dots, \phi_n \vdash \chi \rightarrow \psi.$$

This deduction theorem (DT) is not itself formulated with propositional calculus: it is not a theorem of propositional calculus, but a theorem about propositional calculus. In this sense, it is a meta-theorem, comparable to theorems about the soundness or completeness of propositional calculus.

On the other hand, DT is so useful for simplifying the syntactical proof process that it can be considered and used as another inference rule, accompanying modus ponens. In this sense, DT corresponds to the natural conditional proof inference rule which is part of the first version of propositional calculus introduced in this article.

The converse of DT is also valid:

If the sequence

$$\phi_1, \phi_2, \dots, \phi_n \vdash \chi \rightarrow \psi$$

has been demonstrated, then it is also possible to demonstrate the sequence

$$\phi_1, \phi_2, \dots, \phi_n, \chi \vdash \psi$$

in fact, the validity of the converse of DT is almost trivial compared to that of DT:

If

$$\phi_1, \dots, \phi_n \vdash \chi \rightarrow \psi$$

then

$$1: \phi_1, \dots, \phi_n, \chi \vdash \chi \rightarrow \psi$$

$$2: \phi_1, \dots, \phi_n, \chi \vdash \chi$$

and from (1) and (2) can be deduced

$$3: \phi_1, \dots, \phi_n, \chi \vdash \psi$$

by means of modus ponens, Q.E.D.

The converse of DT has powerful implications: it can be used to convert an axiom into an inference rule. For example, the axiom AND-1,

$$\vdash \phi \wedge \chi \rightarrow \phi$$

can be transformed by means of the converse of the deduction theorem into the inference rule

$$\phi \wedge \chi \vdash \phi$$

which is conjunction elimination, one of the ten inference rules used in the first version (in this article) of the propositional calculus.

Example of a proof

The following is an example of a (syntactical) demonstration, involving only axioms THEN-1 and THEN-2:

Prove: $A \rightarrow A$ (Reflexivity of implication).

Proof:

1. $(A \rightarrow ((B \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A))$

Axiom THEN-2 with $\phi = A$, $\chi = B \rightarrow A$, $\psi = A$

2. $A \rightarrow ((B \rightarrow A) \rightarrow A)$

Axiom THEN-1 with $\phi = A$, $\chi = B \rightarrow A$

3. $(A \rightarrow (B \rightarrow A)) \rightarrow (A \rightarrow A)$

From (1) and (2) by modus ponens.

4. $A \rightarrow (B \rightarrow A)$

Axiom THEN-1 with $\phi = A$, $\chi = B$

5. $A \rightarrow A$

From (3) and (4) by modus ponens.

Equivalence to equational logics

The preceding alternative calculus is an example of a Hilbert-style deduction system. In the case of propositional systems the axioms are terms built with logical connectives and the only inference rule is modus ponens. Equational logic as standardly used informally in high school algebra is a different kind of calculus from Hilbert systems. Its theorems are equations and its inference rules express the properties of equality, namely that it is a congruence on terms that admits substitution.

Classical propositional calculus as described above is equivalent to Boolean algebra, while intuitionistic propositional calculus is equivalent to Heyting algebra. The equivalence is shown by translation in each direction of the theorems of the respective systems. Theorems ϕ of classical or intuitionistic propositional calculus are translated as equations $\phi = 1$ of Boolean or Heyting algebra respectively. Conversely theorems $x = y$ of Boolean or Heyting algebra are translated as theorems $(x \rightarrow y) \wedge (y \rightarrow x)$ of classical or propositional calculus respectively, for which $x \equiv y$ is a standard abbreviation. In the case of Boolean algebra $x = y$ can also be translated as $(x \wedge y) \vee (\neg x \wedge \neg y)$, but this translation is incorrect intuitionistically.

In both Boolean and Heyting algebra, inequality $x \leq y$ can be used in place of equality. The equality $x = y$ is expressible as a pair of inequalities $x \leq y$ and $y \leq x$. Conversely the inequality $x \leq y$ is expressible as the equality $x \wedge y = x$, or as $x \vee y = y$. The significance of inequality for Hilbert-style systems is that it corresponds to the latter's deduction or entailment symbol \vdash . An entailment

$$\phi_1, \phi_2, \dots, \phi_n \vdash \psi$$

is translated in the inequality version of the algebraic framework as

$$\phi_1 \wedge \phi_2 \wedge \dots \wedge \phi_n \leq \psi$$

Conversely the algebraic inequality $x \leq y$ is translated as the entailment

$$x \vdash y.$$

The difference between implication $x \rightarrow y$ and inequality or entailment $x \leq y$ or $x \vdash y$ is that the former is internal to the logic while the latter is external. Internal implication between two terms is another term of the same kind. Entailment as external implication between two terms expresses a metatruth outside the language of the logic, and is considered part of the metalanguage. Even when the logic under study is intuitionistic, entailment is ordinarily understood classically as two-valued: either the left side entails, or is less-or-equal to, the right side, or it is not.

Similar but more complex translations to and from algebraic logics are possible for natural deduction systems as described above and for the sequent calculus. The entailments of the latter can be interpreted as two-valued, but a more insightful interpretation is as a set, the elements of which can be understood as abstract proofs organized as the morphisms of a category. In this interpretation the cut rule of the sequent calculus corresponds to composition in the category. Boolean and Heyting algebras enter this picture as special categories having at most one morphism per homset, i.e., one proof per entailment, corresponding to the idea that existence of proofs is all that matters: any proof will do and there is no point in distinguishing them.

Graphical calculi

It is possible to generalize the definition of a formal language from a set of finite sequences over a finite basis to include many other sets of mathematical structures, so long as they are built up by finitary means from finite materials. What's more, many of these families of formal structures are especially well-suited for use in logic.

For example, there are many families of graphs that are close enough analogues of formal languages that the concept of a calculus is quite easily and naturally extended to them. Indeed, many species of graphs arise as *parse graphs* in the syntactic analysis of the corresponding families of text structures. The exigencies of practical computation on formal languages frequently demand that text strings be converted into pointer structure renditions of parse graphs, simply as a matter of checking whether strings are wffs or not. Once this is done, there are many advantages to be gained from developing the graphical analogue of the calculus on strings. The mapping from strings to parse graphs is called *parsing* and the inverse mapping from parse graphs to strings is achieved by an operation that is called *traversing* the graph.

Other logical calculi

Propositional calculus is about the simplest kind of logical calculus in current use. It can be extended in several ways. (Aristotelian "syllogistic" calculus, which is largely supplanted in modern logic, is in *some* ways simpler – but in other ways more complex – than propositional calculus.) The most immediate way to develop a more complex logical calculus is to introduce rules that are sensitive to more fine-grained details of the sentences being used.

First-order logic (aka first-order predicate logic) results when the "atomic sentences" of propositional logic are broken up into terms, variables, predicates, and quantifiers, all keeping the rules of propositional logic with some new ones introduced. (For example, from "All dogs are mammals" we may infer "If Rover is a dog then Rover is a mammal".) With the tools of first-order logic it is possible to formulate a number of theories, either with explicit axioms or by rules of inference, that can themselves be treated as logical calculi. Arithmetic is the best known of these; others include set theory and mereology. Second-order logic and other higher-order logics are formal extensions of first-order logic. Thus, it makes sense to refer to propositional logic as "*zeroth-order logic*", when comparing it with these logics.

Modal logic also offers a variety of inferences that cannot be captured in propositional calculus. For example, from "Necessarily p " we may infer that p . From p we may infer "It is possible that p ". The translation between modal logics and algebraic logics concerns classical and intuitionistic logics but with the introduction of a unary operator on Boolean or Heyting algebras, different from the Boolean operations, interpreting the possibility modality, and in the case of Heyting algebra a second operator interpreting necessity (for Boolean algebra this is redundant since necessity is the De Morgan dual of possibility). The first operator preserves 0 and disjunction while the second preserves 1 and conjunction.

Many-valued logics are those allowing sentences to have values other than *true* and *false*. (For example, *neither* and *both* are standard "extra values"; "continuum logic" allows each sentence to have any of an infinite number of "degrees of truth" between *true* and *false*.) These logics often require calculational devices quite distinct from propositional calculus. When the values form a Boolean algebra (which may have more than two or even infinitely

many values), many-valued logic reduces to classical logic; many-valued logics are therefore only of independent interest when the values form an algebra that is not Boolean.

Solvers

Finding solutions to propositional logic formulae is an NP-complete problem. However, practical methods exist (e.g., DPLL algorithm, 1962; Chaff algorithm, 2001) that are very fast for many useful cases. Recent work has extended the SAT solver algorithms to work with propositions containing arithmetic expressions; these are the SMT solvers.

References

- [1] Ancient Logic (Stanford Encyclopedia of Philosophy) (<http://plato.stanford.edu/entries/logic-ancient/>)
- [2] Leibniz's Influence on 19th Century Logic (<http://plato.stanford.edu/entries/leibniz-logic-influence/#Con>)
- [3] Beth, Evert W., "Semantic entailment and formal derivability", series: Mededelingen van de Koninklijke Nederlandse Akademie van Wetenschappen, Afdeling Letterkunde, Nieuwe Reeks, vol. 18, no. 13, Noord-Hollandsche Uitg. Mij., Amsterdam, 1955, pp. 309–42. Reprinted in Jaakko Intikka (ed.) *The Philosophy of Mathematics*, Oxford University Press, 1969
- [4] Truth in Frege (<http://frege.brown.edu/heck/pdf/unpublished/TruthInFrege.pdf>)
- [5] Russell's Use of Truth-Tables (<http://digitalcommons.mcmaster.ca/cgi/viewcontent.cgi?article=1219&context=russelljournal>)
- [6] Wernick, William (1942) "Complete Sets of Logical Functions," *Transactions of the American Mathematical Society* **51**, pp. 117–132.

Further reading

- Brown, Frank Markham (2003), *Boolean Reasoning: The Logic of Boolean Equations*, 1st edition, Kluwer Academic Publishers, Norwell, MA. 2nd edition, Dover Publications, Mineola, NY.
- Chang, C.C. and Keisler, H.J. (1973), *Model Theory*, North-Holland, Amsterdam, Netherlands.
- Kohavi, Zvi (1978), *Switching and Finite Automata Theory*, 1st edition, McGraw–Hill, 1970. 2nd edition, McGraw–Hill, 1978.
- Korfage, Robert R. (1974), *Discrete Computational Structures*, Academic Press, New York, NY.
- Lambek, J. and Scott, P.J. (1986), *Introduction to Higher Order Categorical Logic*, Cambridge University Press, Cambridge, UK.
- Mendelson, Elliot (1964), *Introduction to Mathematical Logic*, D. Van Nostrand Company.

Related works

- Hofstadter, Douglas (1979). *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books. ISBN 978-0-465-02656-2.

External links

- Klement, Kevin C. (2006), "Propositional Logic", in James Fieser and Bradley Dowden (eds.), *Internet Encyclopedia of Philosophy*, Eprint (<http://www.iep.utm.edu/p/prop-log.htm>).
- Introduction to Mathematical Logic (<http://www.ltn.lv/~podnieks/mlog/ml2.htm>) by V. Detlovs and K. Podnieks
- Formal Predicate Calculus (http://www.qedeq.org/current/doc/math/qedeq_formal_logic_v1_en.pdf), contains a systematic formal development along the lines of Alternative calculus
- *forall x: an introduction to formal logic* (<http://www.fecundity.com/logic/>), by P.D. Magnus, covers formal semantics and proof theory for sentential logic.
- Propositional Logic on PlanetMath (GFDLed)
- Category:Propositional Calculus (http://www.proofwiki.org/wiki/Category:Propositional_Calculus) on ProofWiki (GFDLed)

Truth function

In mathematical logic, a **truth function** is a function from a set of truth values Wikipedia:Please clarify to truth values. Classically the domain and range of a truth function are $\{\text{truth}, \text{falsehood}\}$, but they may have any number of truth values, including an infinity of these^[citation needed].

A logical connective is truth-functional if the truth-value of a compound sentence is a function of the truth-value of its sub-sentences. A class of connectives is truth-functional if each of its members is. For example, the connective "and" is truth-functional since a sentence like "*Apples are fruits and carrots are vegetables*" is true just in case each of its sub-sentences "*apples are fruits*" and "*carrots are vegetables*" is true, and it is false otherwise. Not all connectives of a natural language, such as English, are truth-functional.

Connectives of the form "*x believes that ...*" are typical examples of connectives that are not truth-functional. If e.g. Mary mistakenly believes that Al Gore was President of the USA on April 20, 2000, but she does not believe that the moon is made of green cheese, then the sentence

"Mary believes that Al Gore was President of the USA on April 20, 2000"

is true while

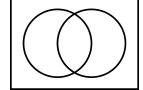
"Mary believes that the moon is made of green cheese"

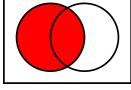
is false. In both cases, each component sentence (i.e. "*Al Gore was president of the USA on April 20, 2000*" and "*the moon is made of green cheese*") is false, but each compound sentence formed by prefixing the phrase "*Mary believes that*" differs in truth-value. That is, the truth-value of a sentence of the form "*Mary believes that...*" is not determined solely by the truth-value of its component sentence, and hence the (unary) connective (or simply *operator* since it is unary) is non-truth-functional.

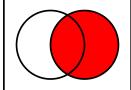
The class of classical logic connectives (e.g. $\&$, \rightarrow) used in the construction of formulas is truth-functional. Their values for various truth-values as argument are usually given by truth tables. Truth-functional propositional calculus is a formal system whose formulas may be interpreted as either true or false.

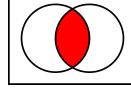
Table of binary truth functions

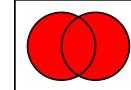
In two-valued logic, there are sixteen possible truth functions, also called Boolean functions, of two inputs P and Q . Any of these functions corresponds to a truth table of a certain logical connective in classical logic, including several degenerate cases such as a function not depending on one or both of its arguments. Truth and falsehood is denoted as 1 and 0 in the following truth tables, respectively, for sake of brevity.

Contradiction/False			
Notation	Equivalent formulas	Truth table	Venn diagram
\perp "bottom"	$P \wedge \neg P$ Opq	Q 0 1 0 0 0 P 1 0 0	

Proposition P			
Notation	Equivalent formulas	Truth table	Venn diagram
P	p Ipq	Q 0 1 0 0 0 P 1 1 1	

Proposition Q			
Notation	Equivalent formulas	Truth table	Venn diagram
Q	q Hpq	Q 0 1 0 0 1 P 1 0 1	

Conjunction			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \wedge Q$ $P \& Q$ $P \cdot Q$ $P \text{ AND } Q$	$P \not\rightarrow \neg Q$ $\neg P \not\leftarrow Q$ $\neg P \downarrow \neg Q$ Kpq	Q 0 1 0 0 0 P 1 0 1	

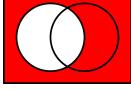
Disjunction			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \vee Q$ $P \text{ OR } Q$	$P \leftarrow \neg Q$ $\neg P \rightarrow Q$ $\neg P \uparrow \neg Q$ $\neg(\neg P \wedge \neg Q)$ Apq	Q 0 1 0 0 1 P 1 1 1	

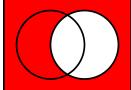
Material nonimplication			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \not\rightarrow Q$ $P \not\nearrow Q$	$P \wedge \neg Q$ $\neg P \downarrow Q$ $\neg P \not\rightarrow$ $\neg Q$ Lpq	Q 0 1 P 0 0 0 1 1 0	

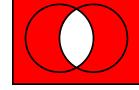
Converse nonimplication			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \not\leftarrow Q$ $P \not\subset Q$	$P \downarrow \neg Q$ $\neg P \wedge Q$ $\neg P \not\rightarrow$ $\neg Q$ Mpq	Q 0 1 P 0 0 1 1 0 0	

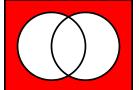
Exclusive disjunction			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \not\leftrightarrow Q$ $P \not\equiv Q$ $P \oplus Q$ $P \text{ XOR } Q$	$P \leftrightarrow \neg Q$ $\neg P \leftrightarrow Q$ $\neg P \not\leftrightarrow$ $\neg Q$ Jpq	Q 0 1 P 0 0 1 1 1 0	

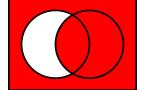
Tautology/True			
Notation	Equivalent formulas	Truth table	Venn diagram
\top "top"	$P \vee \neg P$ Vpq	Q 0 1 P 0 1 1 1 1 1	

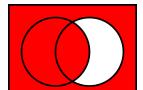
Negation of P			
Notation	Equivalent formulas	Truth table	Venn diagram
$\neg P$ $\sim P$	$\text{N}p$ $\text{F}pq$	Q 0 1 P 0 1 1 1 0 0	

Negation of Q			
Notation	Equivalent formulas	Truth table	Venn diagram
$\neg Q$ $\sim Q$	$\text{N}q$ $\text{G}pq$	Q 0 1 P 0 1 0 1 1 0	

Alternative denial			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \uparrow Q$ $P \downarrow Q$ $P \text{ NAND } Q$	$P \rightarrow \neg Q$ $\neg P \leftarrow Q$ $\neg P \vee \neg Q$ $\text{D}pq$	Q 0 1 P 0 1 1 1 1 0	

Joint denial			
Notation	Equivalent formulas	Truth table	Venn diagram
$P \downarrow Q$ $P \text{ NOR } Q$	$P \not\leftarrow \neg Q$ $\neg P \not\rightarrow Q$ $\neg P \wedge \neg Q$ $\text{X}pq$	Q 0 1 P 0 1 0 1 0 0	

Material implication												
Notation	Equivalent formulas	Truth table	Venn diagram									
$P \rightarrow Q$ $P \supset Q$	$P \uparrow \neg Q$ $\neg P \vee Q$ $\neg P \leftarrow \neg Q$ Cpq	Q <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td></td><td>0</td><td>1</td></tr> <tr><td></td><td>0</td><td>1</td></tr> <tr><td>P</td><td>1</td><td>0</td></tr> </table>		0	1		0	1	P	1	0	
	0	1										
	0	1										
P	1	0										

Converse implication												
Notation	Equivalent formulas	Truth table	Venn diagram									
$P \leftarrow Q$ $P \subset Q$	$P \vee \neg Q$ $\neg P \uparrow Q$ $\neg P \rightarrow \neg Q$ Bpq	Q <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td></td><td>0</td><td>1</td></tr> <tr><td></td><td>0</td><td>1</td></tr> <tr><td>P</td><td>1</td><td>1</td></tr> </table>		0	1		0	1	P	1	1	
	0	1										
	0	1										
P	1	1										

Biconditional												
Notation	Equivalent formulas	Truth table	Venn diagram									
$P \leftrightarrow Q$ $P \equiv Q$ $P \text{ XNOR } Q$ $P \text{ IFF } Q$	$P \not\leftrightarrow \neg Q$ $\neg P \not\leftrightarrow Q$ $\neg P \leftrightarrow \neg Q$ $\neg Q$ Epq	Q <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td></td><td>0</td><td>1</td></tr> <tr><td></td><td>0</td><td>1</td></tr> <tr><td>P</td><td>1</td><td>0</td></tr> </table>		0	1		0	1	P	1	0	
	0	1										
	0	1										
P	1	0										

Functional completeness

Because a function may be expressed as a composition, a truth-functional logical calculus does not need to have dedicated symbols for all of the above-mentioned functions to be functionally complete. This is expressed in a propositional calculus as logical equivalence of certain compound statements. For example, classical logic has $\neg P \vee Q$ equivalent to $P \rightarrow Q$. The conditional operator " \rightarrow " is therefore not necessary for a classical-based logical system if " \neg " (not) and " \vee " (or) are already in use.

A minimal set of operators that can express every statement expressible in the propositional calculus is called a *minimal functionally complete set*. A minimally complete set of operators is achieved by NAND alone $\{\uparrow\}$ and NOR alone $\{\downarrow\}$.

The following are the minimal functionally complete sets of operators whose arities do not exceed 2:^[1]

One element

$\{\uparrow\}, \{\downarrow\}$.

Two elements

$$\{\vee, \neg\}, \{\wedge, \neg\}, \{\rightarrow, \neg\}, \{\leftarrow, \neg\}, \{\rightarrow, \perp\}, \{\leftarrow, \perp\}, \{\rightarrow, \not\leftrightarrow\}, \{\leftarrow, \not\leftrightarrow\}, \{\rightarrow, \not\rightarrow\}, \{\leftarrow, \not\rightarrow\}, \{\leftarrow, \not\leftrightarrow\}, \{\not\rightarrow, \neg\}, \{\not\leftrightarrow, \neg\}, \{\not\rightarrow, \top\}, \{\not\leftrightarrow, \top\}, \{\not\rightarrow, \leftrightarrow\}, \{\not\leftrightarrow, \leftrightarrow\}.$$

Three elements

$$\{\vee, \leftrightarrow, \perp\}, \{\vee, \leftrightarrow, \not\leftrightarrow\}, \{\vee, \not\leftrightarrow, \top\}, \{\wedge, \leftrightarrow, \perp\}, \{\wedge, \leftrightarrow, \not\leftrightarrow\}, \{\wedge, \not\leftrightarrow, \top\}.$$

Algebraic properties

Some truth functions possess properties which may be expressed in the theorems containing the corresponding connective. Some of those properties that a binary truth function (or a corresponding logical connective) may have are:

- **Associativity:** Within an expression containing two or more of the same associative connectives in a row, the order of the operations does not matter as long as the sequence of the operands is not changed.
- **Commutativity:** The operands of the connective may be swapped without affecting the truth-value of the expression.
- **Distributivity:** A connective denoted by \cdot distributes over another connective denoted by $+$, if $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ for all operands a, b, c .
- **Idempotence:** Whenever the operands of the operation are the same, the connective gives the operand as the result. In other words, the operation is both truth-preserving and falsehood-preserving (see below).
- **Absorption:** A pair of connectives \wedge, \vee satisfies the absorption law if $a \wedge (a \vee b) = a$ for all operands a, b .

A set of truth functions is functionally complete if and only if for each of the following five properties it contains at least one member lacking it:

- **monotonic:** If $f(a_1, \dots, a_n) \leq f(b_1, \dots, b_n)$ for all $a_1, \dots, a_n, b_1, \dots, b_n \in \{0,1\}$ such that $a_1 \leq b_1, a_2 \leq b_2, \dots, a_n \leq b_n$. E.g., $\vee, \wedge, \top, \perp$.
- **affine:** Each variable always makes a difference in the truth-value of the operation or it never makes a difference. E.g., $\neg, \leftrightarrow, \not\leftrightarrow, \top, \perp$.
- **self dual:** To read the truth-value assignments for the operation from top to bottom on its truth table is the same as taking the complement of reading it from bottom to top; in other words, $f(\neg a_1, \dots, \neg a_n) = \neg f(a_1, \dots, a_n)$. E.g., \neg .
- **truth-preserving:** The interpretation under which all variables are assigned a truth value of 'true' produces a truth value of 'true' as a result of these operations. E.g., $\vee, \wedge, \top, \rightarrow, \leftrightarrow, C$. (see validity)
- **falsehood-preserving:** The interpretation under which all variables are assigned a truth value of 'false' produces a truth value of 'false' as a result of these operations. E.g., $\vee, \wedge, \not\leftrightarrow, \perp, \neg, D$. (see validity)

Arity

A concrete function may be also referred to as an *operator*. In two-valued logic there are 2 nullary operators (constants), 4 unary operators, 16 binary operators, 256 ternary operators, and 2^{2^n} n -ary operators. In three-valued logic there are 3 nullary operators (constants), 27 unary operators, 19683 binary operators, 7625597484987 ternary operators, and 3^{3^n} n -ary operators. In k -valued logic, there are k nullary operators, k^k unary operators, k^{k^2} binary operators, k^{k^3} ternary operators, and k^{k^n} n -ary operators. An n -ary operator in k -valued logic is a function from $\mathbb{Z}_k^n \rightarrow \mathbb{Z}_k$. Therefore the number of such operators is $|\mathbb{Z}_k|^{|\mathbb{Z}_k^n|} = k^{k^n}$, which is how the above numbers were derived.

However, some of the operators of a particular arity are actually degenerate forms that perform a lower-arity operation on some of the inputs and ignores the rest of the inputs. Out of the 256 ternary boolean operators cited above, $\binom{3}{2} \cdot 16 - \binom{3}{1} \cdot 4 + \binom{3}{0} \cdot 2$ of them are such degenerate forms of binary or lower-arity operators,

using the inclusion-exclusion principle. The ternary operator $f(x, y, z) = \neg x$ is one such operator which is actually a unary operator applied to one input, and ignoring the other two inputs.

"Not" is a unary operator, it takes a single term ($\neg P$). The rest are binary operators, taking two terms to make a compound statement ($P \wedge Q, P \vee Q, P \rightarrow Q, P \leftrightarrow Q$).

The set of logical operators Ω may be partitioned into disjoint subsets as follows:

$$\Omega = \Omega_0 \cup \Omega_1 \cup \dots \cup \Omega_j \cup \dots \cup \Omega_m.$$

In this partition, Ω_j is the set of operator symbols of *arity j*.

In the more familiar propositional calculi, Ω is typically partitioned as follows:

$$\text{nullary operators: } \Omega_0 = \{\perp, \top\}$$

$$\text{unary operators: } \Omega_1 = \{\neg\}$$

$$\text{binary operators: } \Omega_2 \subseteq \{\wedge, \vee, \rightarrow, \leftrightarrow\}$$

Principle of compositionality

Instead of using truth tables, logical connective symbols can be interpreted by means of an interpretation function and a functionally complete set of truth-functions (Gamut 1991), as detailed by the principle of compositionality of meaning. Let I be an interpretation function, let Φ, Ψ be any two sentences and let the truth function f_{nand} be defined as:

- $f_{\text{nand}}(\text{T}, \text{T}) = \text{F}; f_{\text{nand}}(\text{T}, \text{F}) = f_{\text{nand}}(\text{F}, \text{T}) = f_{\text{nand}}(\text{F}, \text{F}) = \text{T}$

Then, for convenience, $f_{\text{not}}, f_{\text{or}}, f_{\text{and}}$ and so on are defined by means of f_{nand} :

- $f_{\text{not}}(x) = f_{\text{nand}}(x, x)$
- $f_{\text{or}}(x, y) = f_{\text{nand}}(f_{\text{not}}(x), f_{\text{not}}(y))$
- $f_{\text{and}}(x, y) = f_{\text{not}}(f_{\text{nand}}(x, y))$

or, alternatively $f_{\text{not}}, f_{\text{or}}, f_{\text{and}}$ and so on are defined directly:

- $f_{\text{not}}(\text{T}) = \text{F}; f_{\text{not}}(\text{F}) = \text{T};$
- $f_{\text{or}}(\text{T}, \text{T}) = f_{\text{or}}(\text{T}, \text{F}) = f_{\text{or}}(\text{F}, \text{T}) = \text{T}; f_{\text{or}}(\text{F}, \text{F}) = \text{F}$
- $f_{\text{and}}(\text{T}, \text{T}) = \text{T}; f_{\text{and}}(\text{T}, \text{F}) = f_{\text{and}}(\text{F}, \text{T}) = f_{\text{and}}(\text{F}, \text{F}) = \text{F}$

Then

- $I(\neg) = I(\neg) = f_{\text{not}}$
- $I(\&) = I(\wedge) = I(\&) = f_{\text{and}}$
- $I(\vee) = I(\vee) = f_{\text{or}}$
- $I(\neg\Phi) = I(\neg\Phi) = I(\neg)(I(\Phi)) = f_{\text{not}}(I(\Phi))$
- $I(\Phi \& \Psi) = I(\&)(I(\Phi), I(\Psi)) = f_{\text{and}}(I(\Phi), I(\Psi))$

etc.

Thus if S is a sentence that is a string of symbols consisting of logical symbols $v_1 \dots v_n$ representing logical connectives, and non-logical symbols $c_1 \dots c_n$, then if and only if $I(v_1) \dots I(v_n)$ have been provided interpreting v_1 to v_n by means of f_{nand} (or any other set of functionally complete truth-functions) then the truth-value of $I(s)$ is determined entirely by the truth-values of $c_1 \dots c_n$ i.e. of $I(c_1) \dots I(c_n)$. In other words, as expected and required, S is true or false only under an interpretation of all its non-logical symbols.

Computer science

Logical operators are implemented as logic gates in digital circuits. Practically all digital circuits (the major exception is DRAM) are built up from NAND, NOR, NOT, and transmission gates. NAND and NOR gates with 3 or more inputs rather than the usual 2 inputs are fairly common, although they are logically equivalent to a cascade of 2-input gates. All other operators are implemented by breaking them down into a logically equivalent combination of 2 or more of the above logic gates.

The "logical equivalence" of "NAND alone", "NOR alone", and "NOT and AND" is similar to Turing equivalence.

That fact that all truth functions can be expressed with NOR alone is demonstrated by the Apollo guidance computer.

Notes

- [1] Wernick, William (1942) "Complete Sets of Logical Functions," *Transactions of the American Mathematical Society* 51: 117–32. In his list on the last page of the article, Wernick does not distinguish between \leftarrow and \rightarrow , or between UNIQ-math-0-a68bc4e06c66b481-QINU and UNIQ-math-1-a68bc4e06c66b481-QINU .

Further reading

- Józef Maria Bocheński (1959), *A Précis of Mathematical Logic*, translated from the French and German versions by Otto Bird, Dordrecht, South Holland: D. Reidel.
- Alonzo Church (1944), *Introduction to Mathematical Logic*, Princeton, NJ: Princeton University Press. See the Introduction for a history of the truth function concept.

References

- *This article incorporates material from TruthFunction on PlanetMath, which is licensed under the Creative Commons Attribution/Share-Alike License.*

General Information and Vocabulary

2-valued morphism

2-valued morphism is a term used in mathematics^[citation needed] to describe a morphism that sends a Boolean algebra B onto a two-element Boolean algebra $\mathbf{2} = \{0,1\}$. It is essentially the same thing as an ultrafilter on B .

A 2-valued morphism can be interpreted as representing a particular state of B . All propositions of B which are mapped to 1 are considered true, all propositions mapped to 0 are considered false. Since this morphism conserves the Boolean operators (negation, conjunction, etc.), the set of true propositions will not be inconsistent but will correspond to a particular maximal conjunction of propositions, denoting the (atomic) state.

The transition between two states s_1 and s_2 of B , represented by 2-valued morphisms, can then be represented by an automorphism f from B to B , such that $s_2 \circ f = s_1$.

The possible states of different objects defined in this way can be conceived as representing potential events. The set of events can then be structured in the same way as invariance of causal structure, or local-to-global causal connections or even formal properties of global causal connections.

The morphisms between (non-trivial) objects could be viewed as representing causal connections leading from one event to another one. For example, the morphism f above leads from event s_1 to event s_2 . The sequences or "paths" of morphisms for which there is no inverse morphism, could then be interpreted as defining horisomatic or chronological precedence relations. These relations would then determine a temporal order, a topology, and possibly a metric.

According to, "A minimal realization of such a relationally determined space-time structure can be found". In this model there are, however, no explicit distinctions. This is equivalent to a model where each object is characterized by only one distinction: (presence, absence) or (existence, non-existence) of an event. In this manner, "the 'arrows' or the 'structural language' can then be interpreted as morphisms which conserve this unique distinction".

If more than one distinction is considered, however, the model becomes much more complex, and the interpretation of distinctional states as events, or morphisms as processes, is much less straightforward.

References

External links

- "Representation and Change - A metarepresentational framework for the foundations of physical and cognitive science" (<http://pcp.vub.ac.be/books/Rep&Change.pdf>)

Bitwise operation

A **bitwise operation** operates on one or more bit patterns or binary numerals at the level of their individual bits. It is a fast, primitive action directly supported by the processor, and is used to manipulate values for comparisons and calculations. On simple low-cost processors, typically, bitwise operations are substantially faster than division, several times faster than multiplication, and sometimes significantly faster than addition. While modern processors usually perform addition and multiplication just as fast as bitwise operations due to their longer instruction pipelines and other architectural design choices, bitwise operations do commonly use less power because of the reduced use of resources.

Bitwise operators

Note that in the explanations below, any indication of a bit's position is counted from the right (least significant) side, advancing left. For example, the binary value 0001 (decimal 1) has zeroes at every position but the first one.

NOT

The **bitwise NOT**, or **complement**, is an unary operation that performs logical negation on each bit, forming the ones' complement of the given binary value. Bits that are 0 become 1, and those that are 1 become 0. For example:

```
NOT 0111 (decimal 7)
= 1000 (decimal 8)
```

The bitwise complement is equal to the two's complement of the value minus one. If two's complement arithmetic is used, then

$$\text{NOT } x = -x - 1.$$

For unsigned integers, the bitwise complement of a number is the "mirror reflection" of the number across the half-way point of the unsigned integer's range. For example, for 8-bit unsigned integers, $\text{NOT } x = 255 - x$, which can be visualized on a graph as a downward line that effectively "flips" an increasing range from 0 to 255, to a decreasing range from 255 to 0. A simple but illustrative example use is to invert a grayscale image where each pixel is stored as an unsigned integer.

AND

A **bitwise AND** takes two binary representations of equal length and performs the logical AND operation on each pair of corresponding bits. The result in each position is 1 if the first bit is 1 *and* the second bit is 1; otherwise, the result is 0. In this, we perform the multiplication of two bits; i.e., $1 \times 0 = 0$ and $1 \times 1 = 1$. For example:

```
0101 (decimal 5)
AND 0011 (decimal 3)
= 0001 (decimal 1)
```

This may be used to determine whether a particular bit is *set* (1) or *clear* (0). For example, given a bit pattern 0011 (decimal 3), to determine whether the second bit is set we use a bitwise AND with a bit pattern containing 1 only in the second bit:

```
0011 (decimal 3)
AND 0010 (decimal 2)
= 0010 (decimal 2)
```

Because the result 0010 is non-zero, we know the second bit in the original pattern was set. This is often called *bit masking*. (By analogy, the use of masking tape covers, or *masks*, portions that should not be altered or portions that are not of interest. In this case, the 0 values mask the bits that are not of interest.)

If we store the result, this may be used to clear selected bits in a register. Given the example 0110 (decimal 6), the second bit may be cleared by using a bitwise AND with the pattern that has a zero only in the second bit:

```
0110 (decimal 6)
AND 1101 (decimal 13)
= 0100 (decimal 4)
```

Because of this property, it becomes easy to check the parity of a binary number by checking the value of the lowest valued bit. Using the example above:

```
0110 (decimal 6)
AND 0001 (decimal 1)
= 0000 (decimal 0)
```

Therefore 6 is divisible by two and even.

OR

A **bitwise OR** takes two bit patterns of equal length and performs the logical inclusive OR operation on each pair of corresponding bits. The result in each position is 1 if the first bit is 1 *or* the second bit is 1 *or* both bits are 1; otherwise, the result is 0. For example:

```
0101 (decimal 5)
OR 0011 (decimal 3)
= 0111 (decimal 7)
```

The bitwise OR may be used to set selected bits, such as a specific bit (or flag) in a register where each bit represents an individual Boolean state. For example 0010 (decimal 2) can be considered a set of four flags, where the first, third, and fourth flags are clear (0) and the second flag is set (1). The fourth flag may be set by performing a bitwise OR between this value and a bit pattern with only the fourth bit set:

```
0010 (decimal 2)
OR 1000 (decimal 8)
= 1010 (decimal 10)
```

This technique is an efficient way to store a number of Boolean values using as little memory as possible.

XOR

A **bitwise XOR** takes two bit patterns of equal length and performs the logical exclusive OR operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 *or* only the second bit is 1, but will be 0 if both are 0 or both are 1. In this we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same. For example:

```
0101 (decimal 5)
XOR 0011 (decimal 3)
= 0110 (decimal 6)
```

The bitwise XOR may be used to invert selected bits in a register (also called toggle or flip). Any bit may be toggled by XORing it with 1. For example, given the bit pattern 0010 (decimal 2) the second and fourth bits may be toggled by a bitwise XOR with a bit pattern containing 1 in the second and fourth positions:

```

0010 (decimal 2)
XOR 1010 (decimal 10)
= 1000 (decimal 8)

```

This technique may be used to manipulate bit patterns representing sets of Boolean states.

Assembly language programmers sometimes use XOR as a short-cut to setting the value of a register to zero. Performing XOR on a value against itself always yields zero, and on many architectures this operation requires fewer clock cycles and/or memory than loading a zero value and saving it to the register.

Atomic inputs

For the examples above often 3 and 5 (binary 0011 and 0101) are used as inputs. They correspond to the unchanged statements among the 2-ary logical connectives. For the 3-ary case 15, 51 and 85 would be used. These numbers are found in the number triangle [OEIS A211344](#):

	1			01		
	3	5		0011	0101	
15	51	85		00001111	00110011	01010101

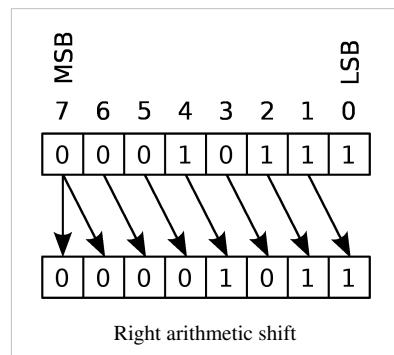
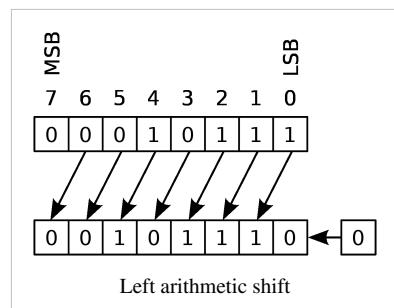
Bit shifts

The **bit shifts** are sometimes considered bitwise operations, because they operate on the binary representation of an integer instead of its numerical value; however, the bit shifts do not operate on pairs of corresponding bits, and therefore cannot properly be called *bit-wise*. In these operations the digits are moved, or *shifted*, to the left or right. Registers in a computer processor have a fixed width, so some bits will be "shifted out" of the register at one end, while the same number of bits are "shifted in" from the other end; the differences between bit shift operators lie in how they determine the values of the shifted-in bits.

Arithmetic shift

In an *arithmetic shift*, the bits that are shifted out of either end are discarded. In a left arithmetic shift, zeros are shifted in on the right; in a right arithmetic shift, the sign bit is shifted in on the left, thus preserving the sign of the operand. Further on, while shifting right, the empty spaces will be filled up with a copy of the most significant bit (MSB). Meaning by shifting the second arithmetic shift register (ASR#2), with a MSB=1, you fill up with 1.

This example uses an 8-bit register:



```

00010111 (decimal +23) LEFT-SHIFT
=
00101110 (decimal +46)

10010111 (decimal -105) RIGHT-SHIFT
=
11001011 (decimal -53)

```

In the first case, the leftmost digit was shifted past the end of the register, and a new 0 was shifted into the rightmost position. In the second case, the rightmost 1 was shifted out (perhaps into the carry flag), and a new 1 was copied into the leftmost position, preserving the sign of the number. Multiple shifts are sometimes shortened to a single shift by some number of digits. For example:

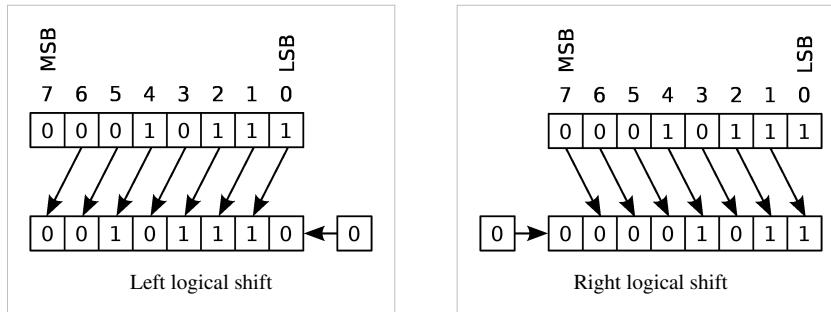
```

00010111 (decimal +23) LEFT-SHIFT-BY-TWO
=
01011100 (decimal +92)

```

A left arithmetic shift by n is equivalent to multiplying by 2^n (provided the value does not overflow), while a right arithmetic shift by n of a two's complement value is equivalent to dividing by 2^n and rounding toward negative infinity. If the binary number is treated as ones' complement, then the same right-shift operation results in division by 2^n and rounding toward zero.

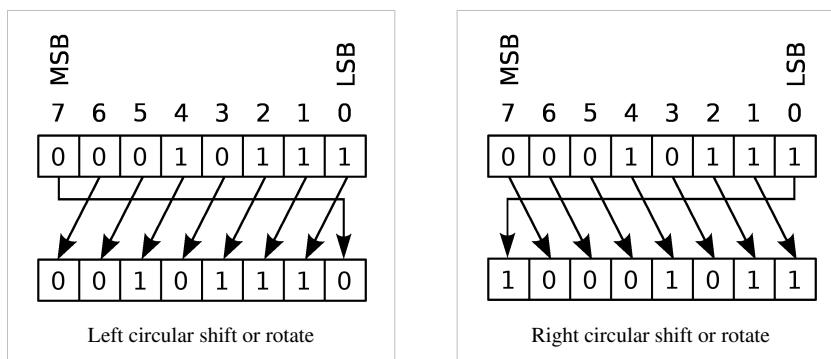
Logical shift



In a *logical shift*, zeros are shifted in to replace the discarded bits. Therefore the logical and arithmetic left-shifts are exactly the same.

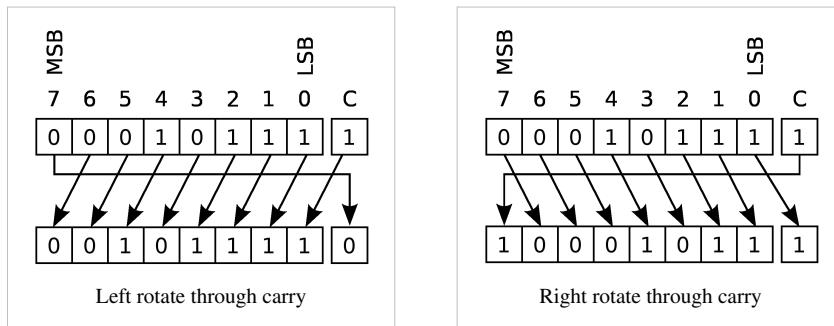
However, as the logical right-shift inserts value 0 bits into the most significant bit, instead of copying the sign bit, it is ideal for unsigned binary numbers, while the arithmetic right-shift is ideal for signed two's complement binary numbers.

Rotate no carry



Another form of shift is the *circular shift* or *bit rotation*. In this operation, the bits are "rotated" as if the left and right ends of the register were joined. The value that is shifted in on the right during a left-shift is whatever value was shifted out on the left, and vice versa. This operation is useful if it is necessary to retain all the existing bits, and is frequently used in digital cryptography.

Rotate through carry



Rotate through carry is similar to the *rotate no carry* operation, but the two ends of the register are separated by the carry flag. The bit that is shifted in (on either end) is the old value of the carry flag, and the bit that is shifted out (on the other end) becomes the new value of the carry flag.

A single *rotate through carry* can simulate a logical or arithmetic shift of one position by setting up the carry flag beforehand. For example, if the carry flag contains 0, then `x RIGHT-ROTATE-THROUGH-CARRY-BY-ONE` is a logical right-shift, and if the carry flag contains a copy of the sign bit, then `x RIGHT-ROTATE-THROUGH-CARRY-BY-ONE` is an arithmetic right-shift. For this reason, some microcontrollers such as PICs just have *rotate* and *rotate through carry*, and don't bother with arithmetic or logical shift instructions.

Rotate through carry is especially useful when performing shifts on numbers larger than the processor's native word size, because if a large number is stored in two registers, the bit that is shifted off the end of the first register must come in at the other end of the second. With rotate-through-carry, that bit is "saved" in the carry flag during the first shift, ready to shift in during the second shift without any extra preparation.

Shifts in C, C++, C#

In C-inspired languages, the left and right shift operators are "`<<`" and "`>>`", respectively. The number of places to shift is given as the second argument to the shift operators. For example,

```
x = y << 2;
```

assigns *x* the result of shifting *y* to the left by two bits.

In C and C++, computations with the left operand as an unsigned integer use logical shifts. In C#, the right-shift is an arithmetic shift when the first operand is an int or long. If the first operand is of type uint or ulong, the right-shift is a logical shift. In C, the results with the left operand as a signed integer are:^[1] In general case:

```
x = a << b then x = a*2^b;
```

```
x = a >> b then x = a/2^b; (rounding towards negative infinity)
```

- for "`<<`": $y \times 2^{\text{right}}$ (undefined if an overflow occurs);
- for "`>>`": implementation-defined (most often the result of the arithmetic shift: $y / 2^{\text{right}}$).

There are also compiler-specific intrinsics implementing circular shifts, like `_rotl8`, `_rotl16`^[2], `_rotr8`, `_rotr16`^[3] in Microsoft Visual C++.

Shifts in Java

In Java, all integer types are signed, and the "<<" and ">>" operators perform arithmetic shifts. Java adds the operator ">>>" to perform logical right shifts, but because the logical and arithmetic left-shift operations are identical, there is no "<<<" operator in Java.

More details of Java shift operators:^[4]

- The operators << (left shift), >> (signed right shift), and >>> (unsigned right shift) are called the *shift operators*.
- The type of the shift expression is the promoted type of the left-hand operand. For example, `aByte >>> 2` is equivalent to `((int) aByte) >>> 2`.
- If the promoted type of the left-hand operand is int, only the five lowest-order bits of the right-hand operand are used as the shift distance. It is as if the right-hand operand were subjected to a bitwise logical AND operator & with the mask value 0x1f (0b11111).^[5] The shift distance actually used is therefore always in the range 0 to 31, inclusive.
- If the promoted type of the left-hand operand is long, then only the six lowest-order bits of the right-hand operand are used as the shift distance. It is as if the right-hand operand were subjected to a bitwise logical AND operator & with the mask value 0x3f (0b111111). The shift distance actually used is therefore always in the range 0 to 63, inclusive.
- The value of $n >>> s$ is n right-shifted s bit positions with zero-extension.

Shifts in Pascal

In Pascal, as well as in all its dialects (such as Object Pascal and Standard Pascal), the left and right shift operators are "shl" and "shr", respectively. The number of places to shift is given as the second argument. For example, the following assigns x the result of shifting y to the left by two bits:

```
x := y shl 2;
```

Applications

Bitwise operations are necessary particularly in lower-level programming such as writing device drivers, low-level graphics, communications protocol packet assembly, and decoding.

Although machines often have efficient built-in instructions for performing arithmetic and logical operations, in fact, all these operations can be performed by combining the bitwise operators and zero-testing in various ways.^[citation needed]

For example, here is a pseudocode example showing how to multiply two arbitrary integers a and b (a greater than b) using only bitshifts and addition:

```
c = 0
while b ≠ 0
    if (b and 1) ≠ 0
        c = c + a
    left shift a by 1
    right shift b by 1

return c
```

NB: In this code = is the assignment operator, not the equality operator.

This implementation of ancient Egyptian multiplication, like most multiplication algorithms, involves bitshifts. In turn, even addition can be written using just bitshifts and zero-testing^[citation needed].

```
while a ≠ 0
    c = b and a
    b = b xor a
    left shift c by 1
    a = c

return b
```

NB: In this code = is the assignment operator, not the equality operator.

References

- [1] JTC1/SC22/WG14 N843 "C programming language" (<http://std.dkuug.dk/JTC1/SC22/WG14/www/docs/n843.htm>), section 6.5.7#5
- [2] [http://msdn.microsoft.com/en-us/library/t5e2f3sc\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/t5e2f3sc(VS.80).aspx)
- [3] [http://msdn.microsoft.com/en-us/library/yy0728bz\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/yy0728bz(VS.80).aspx)
- [4] The Java Language Specification, section 15.19. Shift Operators (<http://docs.oracle.com/javase/specs/jls/se7/html/jls-15.html#jls-15.19>)
- [5] JLS §15.22.1 (<http://docs.oracle.com/javase/specs/jls/se7/html/jls-15.html#jls-15.22.1>)

External links

- Online Bitwise Calculator (<http://www.miniwebtool.com/bitwise-calculator/>) supports Bitwise AND, OR and XOR
- Division using bitshifts (<http://www.cs.uiowa.edu/~jones/bcd/divide.html>)
- " Bitwise Operations Mod N (<http://demonstrations.wolfram.com/BitwiseOperationsModN/>)" by Enrique Zeleny, Wolfram Demonstrations Project.
- " Plots Of Compositions Of Bitwise Operations (<http://demonstrations.wolfram.com/PlotsOfCompositionsOfBitwiseOperations/>)" by Enrique Zeleny, The Wolfram Demonstrations Project.

Boolean data type

In computer science, the **Boolean** or **logical data type** is a data type, having two values (usually denoted **true** and **false**), intended to represent the truth values of logic and Boolean algebra. It is named after George Boole, who first defined an algebraic system of logic in the mid 19th century. The Boolean data type is the primary result of conditional statements, which allow different actions and change control flow depending on whether a programmer-specified boolean *condition* evaluates to true or false.

Generalities

In programming languages that have a built-in Boolean data type, such as Pascal and Java, the comparison operators such as `>` and `≠` are usually defined to return a Boolean value. Conditional and iterative commands may be defined to test Boolean-valued expressions.

Languages without an explicit Boolean data type, like C90 and Lisp, may still represent truth values by some other data type. Lisp uses an empty list for false, and any other value for true. C uses an integer type, where relational expressions like `i > j` and logical expressions connected by `&&` and `||` are defined to have value 1 if true and 0 if false, whereas the test parts of `if`, `while`, `for`, etc., treat any non-zero value as true. Indeed, a Boolean variable may be regarded (and be implemented) as a numerical variable with a single binary digit (bit), which can store only two values. It is worth noting that the implementation of booleans in computers are most likely represented as a full word, rather than a bit; this is usually due to the ways computers transfer blocks of information.

Most programming languages, even those that do not have an explicit Boolean type, have support for Boolean algebraic operations such as conjunction (`AND`, `&`, `*`), disjunction (`OR`, `|`, `+`), equivalence (`EQV`, `=`, `==`), exclusive or/non-equivalence (`XOR`, `NEQV`, `^`, `!=`), and not (`NOT`, `~`, `!`).

In some languages, like Ruby, Smalltalk, and Alice the "true" and "false" values belong to separate classes—e.g. `True` and `False`, resp.—so there is no single Boolean "type."

In SQL, which uses a three-valued logic for explicit comparisons because of its special treatment of Nulls, the Boolean data type (introduced in SQL:1999) is also defined to include more than two truth values, so that SQL "Booleans" can store all logical values resulting from the evaluation of predicates in SQL. A column of Boolean type can also be restricted to just `TRUE` and `FALSE` though.

In the lambda calculus model of computing, Boolean values can be represented as Church booleans.

ALGOL, Java, and C#

One of the earliest languages to provide an explicit **Boolean** data type was ALGOL 60 (1960) with values **true** and **false** and logical operators denoted by symbols '`∧`' (and), '`∨`' (or), '`▷`' (implies), '`≡`' (equivalence), and '`¬`' (not). Due to input device limitations of the time, however, most compilers used alternative representations for the latter, such as `AND` or '`AND`'.

This approach ("Boolean is a separate built-in primitive data type") was adopted by many later languages, such as ALGOL 68 (1970), Java, and C#.

Fortran

The first version of FORTRAN (1957) and its successor FORTRAN II (1958) did not have logical values or operations; even the conditional `IF` statement took an arithmetic expression and branched to one of three locations according to its sign; see arithmetic IF. FORTRAN IV (1962), however, followed the ALGOL 60 example by providing a Boolean data type (`LOGICAL`), truth literals (`.TRUE.` and `.FALSE.`), Boolean-valued numeric comparison operators (`.EQ.`, `.GT.`, etc.), and logical operators (`.NOT.`, `.AND.`, `.OR.`). In `FORMAT` statements, a

specific control character ('L') was provided for the parsing or formatting of logical values.^[1]

Lisp and Scheme

The Lisp language (1958) never had a built-in Boolean data type. Instead, conditional constructs like `cond` assume that the logical value "false" is represented by the empty list `()`, which is defined to be the same as the special atom `nil` or `NIL`; whereas any other s-expression is interpreted as "true". For convenience, most modern dialects of Lisp predefined the atom `t` to have value `t`, so that one can use `t` as a mnemonic notation for "true".

This approach ("any value can be used as a Boolean value") was retained in most Lisp dialects (Common Lisp, Scheme, Emacs Lisp), and similar models were adopted by many scripting languages, even ones that do have a distinct Boolean type or Boolean values; although which values are interpreted as "false" and which are "true" vary from language to language. In Scheme, for example, the "false" value is an atom distinct from the empty list, so the latter is interpreted as "true".

Pascal, Ada, and Haskell

The Pascal language (1978) introduced the concept of programmer-defined enumerated types. A built-in Boolean data type was then provided as a predefined enumerated type with values `FALSE` and `TRUE`. By definition, all comparisons, logical operations, and conditional statements applied to and/or yielded Boolean values. Otherwise, the Boolean type had all the facilities which were available for enumerated types in general — such as ordering and use as indices. On the other hand, the conversion between Booleans and integers (or any other types) still required explicit tests or function calls, as in ALGOL 60. This approach ("Boolean is an enumerated type") was adopted by most later languages which had enumerated types, such as Modula, Ada and Haskell.

C, C++, Objective-C, Awk, Perl

The initial implementations of the C language (1972) provided no Boolean type, and to this day Boolean values are commonly represented by integers (`ints`) in C programs. The comparison operators ('>', '==', etc.) are defined to return a signed integer (`int`) result, either zero (for false) or 1 (for true). Logical operators ('&&', '||', '!', etc.) and condition-testing statements ('`if`', '`while`') assume that zero is false and all other values are true. One problem with this approach is that the tests `if (t==TRUE) { . . . }` and `if (t)` are not equivalent.

After enumerated types (`enums`) were added to the ANSI version of C (1989), many C programmers got used to defining their own Boolean types as such, for readability reasons. However, enumerated types are equivalent to integers according to the language standards; so the effective identity between Booleans and integers is still valid for C programs.

Standard C (since C99) and several dialects of C such as Objective-C provide definitions of a Boolean type as an integer type and macros for "false" and "true" as 0 and 1, respectively. Thus logical values can be stored in integer variables, and used anywhere integers would be valid, including in indexing, arithmetic, parsing, and formatting. This approach ("Boolean values are just integers") has been retained in all later versions of C.

C++ has a separate Boolean data type ('`bool`'), but with automatic conversions from scalar and pointer values that are very similar to those of C. This approach was adopted also by many later languages, especially by some scripting ones such as AWK.

Objective-C also has a separate Boolean data type ('`BOOL`'), with possible values being `YES` or `NO`, equivalents of true and false respectively.^[2]

Perl has many values of `false`: the number zero, the strings "0" and "", the empty list "()", and the special value `undef`. Everything else evaluates to `true`.

Python, Ruby, and JavaScript

In Python from version 2.3 forward, there is a `bool` type which is a subclass of `int`, the standard integer type. It has two possible values: `True` and `False`, which are "special versions" of 1 and 0 respectively and behave as such in arithmetic contexts. In addition, a numeric value of zero (integer or fractional), the null value (`None`), the empty string, and empty containers (i.e. lists, sets, etc.) are considered Boolean false; all other values are considered Boolean true by default. Classes can define how their instances are treated in a Boolean context through the special method `__nonzero__` (Python 2) or `__bool__` (Python 3). For containers, `__len__` (the special method for determining the length of containers) is used if the explicit Boolean conversion method is not defined.

In Ruby, on the other hand, only `nil` (Ruby's null value) and a special `false` object are "false", everything else (including the integer 0 and empty arrays) is "true".

In JavaScript, the empty string (" "), `null`, `undefined`, `NaN`, `+0`, `-0` and `false` are sometimes called "falsy", and their complement, "truthy", to distinguish between strictly type-checked and coerced Booleans. Languages such as PHP also use this approach.

SQL

The SQL:1999 standard introduced a BOOLEAN data type as an optional feature (T031). When restricted by a NOT NULL constraint, a SQL BOOLEAN behaves like Booleans in other languages. In SQL however, the BOOLEAN type is nullable by default like all other SQL data types, meaning it can have the special null value as well. Although the SQL standard defines three literals for the BOOLEAN type—TRUE, FALSE and UNKNOWN—, it also says that the NULL BOOLEAN and UNKNOWN "may be used interchangeably to mean exactly the same thing".^[3] This has caused some controversy because the identification subjects UNKNOWN to the equality comparison rules for NULL. More precisely UNKNOWN = UNKNOWN is not TRUE but UNKNOWN=NULL. As of 2012 few major SQL systems implement the T031 feature.^[4] PostgreSQL is a notable exception, although it does not implement the UNKNOWN literal; NULL can be used instead.^[5] (PostgreSQL does implement the IS UNKNOWN operator, which is part of an orthogonal feature, F571.) In other SQL implementations various ad hoc solutions are used, like bit, byte, and character to simulate Boolean values.

References

- [1] Digital Equipment Corporation, *DECSystem-10 FORTRAN IV Programmers Reference Manual*. Reprinted in *Mathematical Languages Handbook*. Online version (http://www.bitsavers.org/pdf/tymshare/tymcom-x/Tymcom-X_Reference_Series_Fortran_IV_Jan73.pdf) accessed 2011-11-16.
- [2] <http://developer.apple.com/library/ios/#documentation/cocoa/conceptual/ProgrammingWithObjectiveC/FoundationTypesandCollections/FoundationTypesandCollections.html>
- [3] ISO/IEC 9075-2:2011 §4.5
- [4] Troels Arvin, Survey of BOOLEAN data type implementation (http://troels.arvin.dk/db/rdbms/#data_types-boolean)
- [5] <http://www.postgresql.org/docs/current/static/datatype-boolean.html>

Boolean expression

In computer science, a **Boolean expression** is an expression in a programming language that produces a Boolean value when evaluated, i.e. one of **true** or **false**. A Boolean expression may be composed of a combination of the Boolean constants **true** or **false**, Boolean-typed variables, Boolean-valued operators, and Boolean-valued functions.

Boolean operators

Most programming languages have the Boolean operators OR, AND and NOT; in C and some newer languages, these are represented by "||" (double pipe character), "&&" (double ampersand) and "!" (exclamation point) respectively, while the corresponding bitwise operations are represented by "|", "&" and "~" (tilde). In theoretical literature the symbols used are often "+" (plus), "·" (dot) and overbar, or " \vee " (cup), " \wedge " (cap) and " \neg " or " $'$ " (prime).

Examples

- The expression "5 > 3" is evaluated as **true**.
- "5>=3" and "3<=5" are equivalent Boolean expressions, both of which are evaluated as **true**.
- Of course, most Boolean expressions will contain at least one variable ($X > 3$), and often more ($X > Y$).

External links

- The Calculus of Logic^[1], by George Boole, Cambridge and Dublin Mathematical Journal Vol. III (1848), pp. 183–98.

References

[1] <http://www.maths.tcd.ie/pub/HistMath/People/Boole/CalcLogic/CalcLogic.html>

Boolean satisfiability problem

In computer science, **satisfiability** (often written in all capitals or abbreviated **SAT**) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula. In other words, it establishes if the variables of a given Boolean formula can be assigned in such a way as to make the formula evaluate to TRUE. If no such assignments exist, the function expressed by the formula is identically FALSE for all possible variable assignments. In this latter case, it is called **unsatisfiable**, otherwise **satisfiable**. For example, the formula " a AND NOT b " is satisfiable because one can find the values $a = \text{TRUE}$ and $b = \text{FALSE}$, which make $(a \text{ AND NOT } b) = \text{TRUE}$. In contrast, " a AND NOT a " is unsatisfiable. To emphasize the binary nature of this problem, it is frequently referred to as *Boolean* or *propositional satisfiability*.

SAT was the first known example of an NP-complete problem. That briefly means that there is no known algorithm that efficiently solves all instances of SAT, and it is generally believed (but not proven, see P versus NP problem) that no such algorithm can exist. Further, a wide range of other naturally occurring decision and optimization problems can be transformed into instances of SAT. A class of algorithms called SAT solvers can efficiently solve a large enough subset of SAT instances to be useful in various practical areas such as circuit design and automatic theorem proving, by solving SAT instances made by transforming problems that arise in those areas. Extending the capabilities of SAT solving algorithms is an ongoing area of progress. However, no current such methods can efficiently solve *all* SAT instances.

Basic definitions and terminology

A **propositional logic formula**, also called **Boolean expression**, is built from variables, operators AND (conjunction, also denoted by \wedge), OR (disjunction, \vee), NOT (negation, \neg), and parentheses. A formula is said to be **satisfiable** if it can be made true by assigning appropriate logical values (i.e. TRUE, FALSE) to its variables. The **Boolean satisfiability problem (SAT)** is, given a formula, to check whether it is satisfiable. This decision problem is of central importance in various areas of computer science, including theoretical computer science, complexity theory, algorithmics, and artificial intelligence..

There are several special cases of the Boolean satisfiability problem in which the formulas are required to have a particular structure. A **literal** is either a variable, then called **positive literal**, or the negation of a variable, then called **negative literal**. A **clause** is a disjunction of literals (or a single literal). A clause is called **Horn clause** if it contains at most one positive literal. A formula is in **conjunctive normal form (CNF)** if it is a conjunction of clauses (or a single clause). For example, " x_1 " is a positive literal, " $\neg x_2$ " is a negative literal, " $x_1 \vee \neg x_2$ " is a clause, and " $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge \neg x_1$ " is a formula in conjunctive normal form, its 1st and 3rd clause is a Horn clause, but its 2nd clause is not. The formula is satisfiable, choosing $x_1 = \text{FALSE}$, $x_2 = \text{FALSE}$, and x_3 arbitrarily.

As another example, $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$ is a conjunctive normal form having two clauses (denoted by parentheses), four variables (x_1, x_2, x_3, x_4), and three literals per clause. To solve this instance of the decision problem we must determine whether there is a truth value we can assign to each of the variables (x_1 through x_4) such that the entire expression is TRUE. In this instance, there is such an assignment ($x_1 = \text{TRUE}, x_2 = \text{TRUE}, x_3 = \text{TRUE}, x_4 = \text{TRUE}$), so the answer to this instance is YES. This is one of many possible assignments, with for instance, any set of assignments including $x_1 = \text{TRUE}$ being sufficient. If there were no such assignment(s), the answer would be NO.

Using the laws of Boolean algebra, every propositional logic formula can be transformed into an equivalent conjunctive normal form, which may, however, be exponentially longer.

Complexity and restricted versions

Unrestricted satisfiability (SAT)

SAT was the first known NP-complete problem, as proved by Stephen Cook in 1971 and independently by Leonid Levin in 1973.^[1] Until that time, the concept of an NP-complete problem did not even exist. The proof shows how every decision problem in the complexity class NP can be reduced to the SAT problem for CNF^[2] formulas. A useful property of Cook's reduction is that it preserves the number of accepting answers. For example, deciding whether a given graph has a 3-coloring is another problem in NP; if a graph has 17 valid 3-colorings, the SAT formula produced by the Cook–Levin reduction will have 17 satisfying assignments.

NP-completeness only refers to the run-time of the worst case instances. Many of the instances that occur in practical applications can be solved much more quickly. See Algorithms for solving SAT below.

SAT is trivial if the formulas are restricted to those in **disjunctive normal form**, that is, they are disjunction of conjunctions of literals. Such a formula is indeed satisfiable if and only if at least one of its conjunctions is satisfiable, and a conjunctions is satisfiable if and only if it does not contain both x and NOT x for some variable x . This can be checked in linear time. Furthermore, if they are restricted to being in **full disjunctive normal form**, in which every variable appears exactly once in every conjunction, they can be checked in constant time (each conjunction represents one satisfying assignment). But it can take exponential time and space to convert a general SAT problem to disjunctive normal form.

3-satisfiability

Like the satisfiability problem for arbitrary formulas, determining the satisfiability of a formula in conjunctive normal form where each clause is limited to at most three literals is NP-complete also; this problem is called **3-SAT**, **3CNFSAT**, or **3-satisfiability**. To reduce the unrestricted SAT problem to 3-SAT, transform each clause " $l_1 \vee \dots \vee l_n$ " to a conjunction of $n-2$ clauses " $(l_1 \vee l_2 \vee x_2) \wedge (\neg l_2 \vee l_3 \vee x_3) \wedge (\neg l_3 \vee l_4 \vee x_4) \wedge \dots \wedge (\neg l_{n-3} \vee l_{n-2} \vee x_{n-2}) \wedge (\neg l_{n-2} \vee l_{n-1} \vee l_n)$ ", where x_2, \dots, x_{n-2} are new variables not occurring elsewhere. Although both formulas are not logically equivalent, they are equisatisfiable. The formula resulting from transforming all clauses is at most 3 times as long as its original, i.e. the length growth is polynomial.^[3]

3-SAT is one of Karp's 21 NP-complete problems, and it is used as a starting point for proving that other problems are also NP-hard. This is done by polynomial-time reduction from 3-SAT to the other problem.

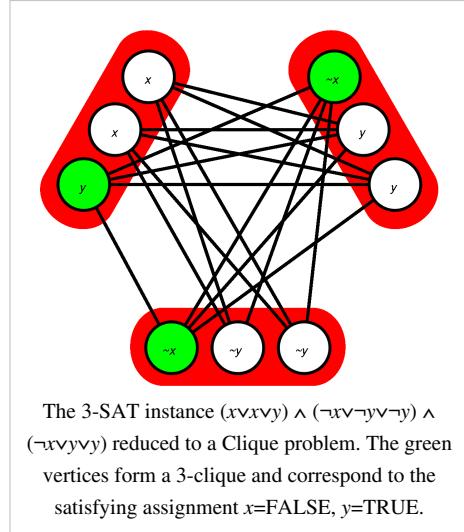
An example of a problem where this method has been used is the Clique problem: given a CNF formula consisting of c clauses, the corresponding graph consists of an edge for each literal, and a vertex between each two non-contradicting^[4] literals from different clauses, cf. picture. The graph has a c -clique if and only if the formula is satisfiable.^[5]

There is a simple randomized algorithm due to Schöning (1999) that runs in time $(4/3)^n$ where n is the number of clauses and succeeds with high probability to correctly decide 3-SAT.

The exponential time hypothesis is that no algorithm can solve 3-SAT faster than $\exp(o(n))$.

Selman, Mitchell, and Levesque (1996) give empirical data on the difficulty of randomly generated 3-SAT formulas, depending on their size parameters. Difficulty is measured in number recursive calls made by a DPLL algorithm.

3-satisfiability can be generalized to **k-satisfiability (k-SAT)**, when formulas in CNF are considered with each clause containing up to k literals. However, since for any $k \geq 3$, this problem can neither be easier than 3-SAT nor



harder than SAT, and the latter two are NP-complete, so must be k-SAT.

Some authors restrict k-SAT to CNF formulas with **exactly k literals**. This doesn't lead to a different complexity class either, as each clause " $l_1 \vee \dots \vee l_j$ " with $j < k$ literals can be padded with fixed dummy variables to " $l_1 \vee \dots \vee l_j \vee d_{j+1} \vee \dots \vee d_k$ ". After padding all clauses, $2^k - 1$ extra clauses^[6] have to be appended to ensure that only $d_1 = \dots = d_k = \text{FALSE}$ can lead to a satisfying assignment. Since k doesn't depend on the formula length, the extra clauses lead to a constant increase in length. For the same reason, it does not matter whether **duplicate literals** are allowed in clauses (like e.g. " $\neg x \vee \neg y \vee \neg y$ "), or not.

Exactly-1 3-satisfiability

A variant of the 3-satisfiability problem is the **one-in-three 3-SAT** (also known variously as **1-in-3-SAT** and **exactly-1 3-SAT**). Given a conjunctive normal form, the problem is to determine whether there exists a truth assignment to the variables so that each clause has *exactly* one true literal (and thus exactly two false literals). In contrast, ordinary 3-SAT requires that every clause has at least one true literal.

$R(x, a, d) \wedge R(y, b, d) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(z, c, 0)$	$R(\neg x, a, b) \wedge R(b, y, c) \wedge R(c, d, \neg z)$
$R(0, a, 0) \wedge R(0, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(0, c, 0)$	$R(1, a, 1) \wedge R(b, 0, c) \wedge R(c, d, 1)$
$R(0, a, 0) \wedge R(0, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(1, c, 0)$	$R(1, a, 0) \wedge R(b, 0, c) \wedge R(c, d, 0)$
$R(0, a, 0) \wedge R(1, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(0, c, 0)$	$R(1, a, 1) \wedge R(b, 1, c) \wedge R(c, d, 1)$
$R(0, a, 0) \wedge R(1, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(1, c, 0)$	$R(1, a, 0) \wedge R(b, 1, c) \wedge R(c, d, 0)$
$R(1, a, 0) \wedge R(0, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(0, c, 0)$	$R(0, a, 1) \wedge R(b, 0, c) \wedge R(c, d, 1)$
$R(1, a, 0) \wedge R(0, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(1, c, 0)$	$R(0, a, 0) \wedge R(b, 0, c) \wedge R(c, d, 0)$
$R(1, a, 0) \wedge R(1, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(0, c, 0)$	$R(0, a, 1) \wedge R(b, 1, c) \wedge R(c, d, 1)$
$R(1, a, 0) \wedge R(1, b, 0) \wedge R(a, b, e) \wedge R(c, d, f) \wedge R(1, c, 0)$	$R(0, a, 0) \wedge R(b, 1, c) \wedge R(c, d, 0)$

Left: Schaefer's reduction of a 3-SAT clause $x \vee y \vee z$. The result of R is **true (1)** if exactly one of its arguments is, and **false (0)** else. All 8 combinations of values for x, y, z are examined, one per line. The fresh variables a, \dots, f can be chosen to satisfy all clauses (exactly one green argument for each R) in all lines except the first, where $x \vee y \vee z$ is false.

Right: A simpler reduction with the same properties.

One-in-three 3-SAT is listed as NP-complete problem "LO4" in the standard reference, *Computers and Intractability: A Guide to the Theory of NP-Completeness* by Michael R. Garey and David S. Johnson. It was proved to be NP-complete by Thomas J. Schaefer as a special case of Schaefer's dichotomy theorem, which asserts that any problem generalizing Boolean satisfiability in a certain way is either in the class P or is NP-complete.

Schaefer gives a construction allowing an easy polynomial-time reduction from 3-SAT to one-in-three 3-SAT. Let " $(x \text{ or } y \text{ or } z)$ " be a clause in a 3CNF formula. Add six new boolean variables a, b, c, d, e , and f , to be used to simulate this clause and no other. Let $R(u, v, w)$ be a predicate that is true if and only if exactly one of the booleans u, v , and w is true. Then the formula " $R(x, a, d) \text{ and } R(y, b, d) \text{ and } R(a, b, e) \text{ and } R(c, d, f) \text{ and } R(z, c, \text{false})$ " is satisfiable by some setting of the new variables if and only if at least one of x, y , or z is true, see picture (left). Thus any 3-SAT instance with m clauses and n variables may be converted into a one-in-three 3-SAT instance with $5m$ clauses and $n+6m$ variables.^[7] Another reduction involves only four new variables and three clauses: $R(\neg x, a, b) \wedge R(b, y, c) \wedge R(c, d, \neg z)$, see picture (right).

The one-in-three 3-SAT problem is often used in the literature as a known NP-complete problem in a reduction to show that other problems are NP-complete.^[citation needed]

2-satisfiability

SAT is easier if the number of literals in a clause is limited to at most 2, in which case the problem is called **2-SAT**. This problem can be solved in polynomial time, and in fact is complete for the complexity class NL. If additionally all OR operations in literals are changed to XOR operations, the result is called **exclusive-or 2-satisfiability**, which is a problem complete for the complexity class SL = L.

Horn-satisfiability

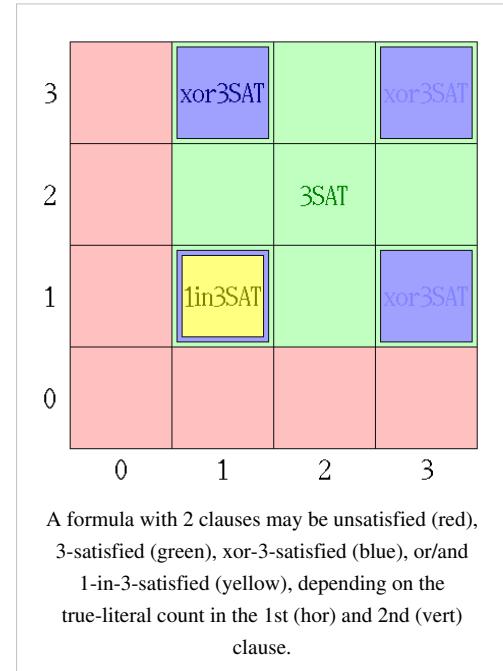
The problem of deciding the satisfiability of a given conjunction of Horn clauses is called **Horn-satisfiability**, or **HORN-SAT**. It can be solved in polynomial time by a single step of the Unit propagation algorithm, which produces the single minimal model of the set of Horn clauses (w.r.t. the set of literals assigned to true). Horn-satisfiability is P-complete. It can be seen as P's version of the Boolean satisfiability problem.

Horn clauses are of interest because they are able to express implication of one variable from a set of other variables. Indeed, one such clause $\neg x_1 \vee \dots \vee \neg x_n \vee y$ can be rewritten as $x_1 \wedge \dots \wedge x_n \rightarrow y$, that is, if x_1, \dots, x_n are all true, then y needs to be true as well.

A generalization of the class of Horn formulae is that of renamable-Horn formulae, which is the set of formulae that can be placed in Horn form by replacing some variables with their respective negation. Checking the existence of such a replacement can be done in linear time; therefore, the satisfiability of such formulae is in P as it can be solved by first performing this replacement and then checking the satisfiability of the resulting Horn formula.

XOR-satisfiability

Solving an XOR-SAT example
by Gaussian elimination



Given formula

("⊕" means XOR, the red clause is optional)

$$(a \oplus c \oplus d) \wedge (b \oplus \neg c \oplus d) \wedge (a \oplus b \oplus \neg d) \wedge (a \oplus \neg b \oplus \neg c) \wedge (\neg a \oplus b \oplus c)$$

Equation system

("1" means TRUE, "0" means FALSE)

Each clause leads to one equation.

$$\begin{array}{cccccc} a & \oplus & c & \oplus & d & = 1 \\ b & \oplus & \neg c & \oplus & d & = 1 \\ a & \oplus & b & \oplus & \neg d & = 1 \\ a & \oplus & \neg b & \oplus & \neg c & = 1 \\ \neg a & \oplus & b & \oplus & c & \simeq 1 \end{array}$$

Normalized equation systemusing properties of Boolean rings ($\neg x = 1 \oplus x$, $x \oplus x = 0$)

$$\begin{array}{cccccc} a & \oplus & c & \oplus & d & = 1 \\ b & \oplus & c & \oplus & d & = 0 \\ a & \oplus & b & \oplus & d & = 0 \\ a & \oplus & b & \oplus & c & = 1 \\ \textcolor{red}{a} & \oplus & \textcolor{red}{b} & \oplus & \textcolor{red}{c} & \simeq 0 \end{array}$$

(If the red equation is present, it contradicts
the last black one, so the system is unsolvable.

Therefore, Gauss' algorithm is
used only for the black equations.)

Associated coefficient matrix

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>	line
1	0	1	1	1 A
0	1	1	1	0 B
1	1	0	1	0 C
1	1	1	0	1 D

Transforming to echelon form

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>		operation
1	0	1	1	1	A
1	1	0	1	0	C
1	1	1	0	1	D
0	1	1	1	0	B (swapped)
1	0	1	1	1	A
0	1	1	0	1	E = C \oplus A
0	1	0	1	0	F = D \oplus A
0	1	1	1	0	B
1	0	1	1	1	A
0	1	1	0	1	E
0	0	1	1	1	G = F \oplus E
0	0	0	1	1	H = B \oplus E

Transforming to diagonal form

<i>a</i>	<i>b</i>	<i>c</i>	<i>d</i>		operation
1	0	1	0	0	I = A \oplus H
0	1	1	0	1	E
0	0	1	0	0	J = G \oplus H
0	0	0	1	1	H
1	0	0	0	0	K = I \oplus J
0	1	0	0	1	L = E \oplus J
0	0	1	0	0	J
0	0	0	1	1	H

Solution:

If the red clause is present: Unsolvable

Else: $a = 0 = \text{FALSE}$

$b = 1 = \text{TRUE}$

$c = 0 = \text{FALSE}$

$d = 1 = \text{TRUE}$

As a consequence:

$$R(a,c,d) \wedge R(b,\neg c,d) \wedge R(a,b,\neg d) \wedge R(a,\neg b,\neg c) \wedge R(\neg a,b,c)$$

is not 1-in-3-satisfiable,

$$\text{while } (a \vee c \vee d) \wedge (b \vee \neg c \vee d) \wedge (a \vee b \vee \neg d) \wedge (a \vee \neg b \vee \neg c)$$

is 3-satisfiable with $a=c=\text{FALSE}$ and $b=d=\text{TRUE}$.

Another special case is the class of problems where each clause contains XOR (i.e. exclusive or) rather than (plain) OR operators. This is in P, since an XOR-SAT formula can also be viewed as a system of linear equations mod 2, and can be solved in cubic time by Gaussian elimination, see box for an example.^[citation needed] This recast is based on the kinship between Boolean algebras and Boolean rings. Since $a \text{ XOR } b \text{ XOR } c$ evaluates to TRUE if and only if exactly 1 or 3 members of $\{a,b,c\}$ are TRUE, each solution of the 1-in-3-SAT problem for a given CNF formula is also a solution of the XOR-SAT problem, and in turn each solution of XOR-SAT is a solution of 3-SAT, cf. picture. As a consequence, for each CNF formula, either the 3-SAT problem or the 1-in-3-SAT problem can be decided in cubic time.Wikipedia:Please clarify

Provided that the complexity classes P and NP are not equal, neither 2-, nor Horn-, nor XOR-satisfiability is NP-complete, unlike SAT. The assumption that P and NP are not equal is currently not proven.

Schaefer's dichotomy theorem

The restrictions above (CNF, 2CNF, 3CNF, Horn, XOR-SAT) bound the considered formulae to be conjunction of subformulae; each restriction states a specific form for all subformulae: for example, only binary clauses can be subformulae in 2CNF.

Schaefer's dichotomy theorem states that, for any restriction to Boolean operators that can be used to form these subformulae, the corresponding satisfiability problem is in P or NP-complete. The membership in P of the satisfiability of 2CNF, Horn, and XOR-SAT formulae are special cases of this theorem.

Extensions of SAT

An extension that has gained significant popularity since 2003 is **Satisfiability modulo theories (SMT)** that can enrich CNF formulas with linear constraints, arrays, all-different constraints, uninterpreted functions,^[8] etc. Such extensions typically remain NP-complete, but very efficient solvers are now available that can handle many such kinds of constraints.

The satisfiability problem becomes more difficult if both "for all" (\forall) and "there exists" (\exists) quantifiers are allowed to bind the Boolean variables. An example of such an expression would be " $\forall x \forall y \exists z (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z)$ "; it is valid, since for all values of x and y , an appropriate value of z can be found, viz. $z=\text{TRUE}$ if both x and y are FALSE, and $z=\text{FALSE}$ else. SAT itself (tacitly) uses only \exists quantifiers. If only \forall quantifiers are allowed instead, the so-called **tautology problem** is obtained, which is Co-NP-complete. If both quantifiers are allowed, the problem is called the **quantified Boolean formula problem (QBF)**, which can be shown to be PSPACE-complete. It is widely believed that PSPACE-complete problems are strictly harder than any problem in NP, although this has not yet been proved.

A number of variants deal with the number of variable assignments making the formula true. Ordinary SAT asks if there is at least one such assignment. **MAJ-SAT**, which asks if the majority of all assignments make the formula true, is complete for PP, a probabilistic class. The problem of **how many variable assignments** satisfy a formula, not a decision problem, is in #P. **UNIQUE-SAT** is the problem of determining whether a formula has exactly one assignment, it is complete for US.Wikipedia:Please clarify When the input is restricted to formulas having at most one satisfying assignment (or none), the problem is called **UNAMBIGUOUS-SAT**. A solving algorithm for UNAMBIGUOUS-SAT is allowed to exhibit any behavior, including endless looping, on a formula having several satisfying assignments. Although this problem seems easier, Valiant and Vazirani have shown that if there is a practical (i.e. randomized polynomial-time) algorithm to solve it, then all problems in NP can be solved just as easily.

The **maximum satisfiability problem**, an FNP generalization of SAT, asks for the maximum number of clauses which can be satisfied by any assignment. It has efficient approximation algorithms, but is NP-hard to solve exactly. Worse still, it is APX-complete, meaning there is no polynomial-time approximation scheme (PTAS) for this

problem unless P=NP.

Other generalisations include satisfiability for first- and second-order logic, constraint satisfaction problems, 0-1 integer programming.

Self-reducibility

The SAT problem is **self-reducible**, that is, each algorithm which correctly answers if an instance of SAT is solvable can be used to find a satisfying assignment. First, the question is asked on the given formula Φ . If the answer is "no", the formula is unsatisfiable. Otherwise, the question is asked on the partly instantiated formula $\Phi\{x_1=\text{TRUE}\}$, i.e. Φ with the first variable x_1 replaced by TRUE, and simplified accordingly. If the answer is "yes", then $x_1=\text{TRUE}$, otherwise $x_1=\text{FALSE}$. Values of other variables can be found subsequently in the same way. In total, $n+1$ runs of the algorithm are required, where n is the number of distinct variables in Φ .

This property of self-reducibility is used in several theorems in complexity theory:

$$\text{NP} \subseteq \text{P/poly} \Rightarrow \text{PH} = \Sigma_2 \quad (\text{Karp-Lipton theorem})$$

(Wikipedia:Please clarify)

$$\text{NP} \subseteq \text{BPP} \Rightarrow \text{NP} = \text{RP}$$

$$\text{P} = \text{NP} \Rightarrow \text{FP} = \text{FNP}$$

Algorithms for solving SAT

Since the SAT problem is NP-complete, only algorithms with exponential worst-case complexity are known for it. In spite of this state of the art in complexity theory, efficient and scalable algorithms for SAT were developed over the last decade and have contributed to dramatic advances in our ability to automatically solve problem instances involving tens of thousands of variables and millions of constraints (i.e. clauses). [citation needed] Examples of such problems in electronic design automation (EDA) include formal equivalence checking, model checking, formal verification of pipelined microprocessors, automatic test pattern generation, routing of FPGAs, planning, and scheduling problems, and so on. A SAT-solving engine is now considered to be an essential component in the EDA toolbox.

There are two classes of high-performance algorithms for solving instances of SAT in practice: the conflict-driven clause learning algorithm, which can be viewed as a modern variant of the DPLL algorithm (well known implementation include Chaff, GRASP) and stochastic local search algorithms, such as WalkSAT.

A DPLL SAT solver employs a systematic backtracking search procedure to explore the (exponentially sized) space of variable assignments looking for satisfying assignments. The basic search procedure was proposed in two seminal papers in the early 60s (see references below) and is now commonly referred to as the Davis–Putnam–Logemann–Loveland algorithm ("DPLL" or "DLL"). Theoretically, exponential lower bounds have been proved for the DPLL family of algorithms.

In contrast, randomized algorithms like the PPSZ algorithm by Paturi, Pudlak, Saks, and Zane set variables in a random order according to some heuristics, for example bounded-width resolution. If the heuristic can't find the correct setting, the variable is assigned randomly. The PPSZ algorithm has a runtime of $O(2^{0.386n})$ for 3-SAT with a single satisfying assignment. Currently this is the best known runtime for this problem. In the setting with many satisfying assignments the randomized algorithm by Schöning has a better bound.^[9]

Modern SAT solvers (developed in the last ten years) come in two flavors: "conflict-driven" and "look-ahead".(Wikipedia:Please clarify) Conflict-driven solvers augment the basic DPLL search algorithm with efficient conflict analysis, clause learning, non-chronological backtracking (aka backjumping), as well as "two-watched-literals" unit propagation, adaptive branching, and random restarts. These "extras" to the basic systematic search have been empirically shown to be essential for handling the large SAT instances that arise in electronic design automation (EDA). Look-ahead solvers have especially strengthened reductions (going beyond

unit-clause propagation) and the heuristics, and they are generally stronger than conflict-driven solvers on hard instances (while conflict-driven solvers can be much better on large instances which actually have an easy instance inside).

Modern SAT solvers are also having significant impact on the fields of software verification, constraint solving in artificial intelligence, and operations research, among others. Powerful solvers are readily available as free and open source software. In particular, the conflict-driven MiniSAT^[10], which was relatively successful at the 2005 SAT competition^[11], only has about 600 lines of code. An example for look-ahead solvers is march_dl^[12], which won a prize at the 2007 SAT competition^[11].

Certain types of large random satisfiable instances of SAT can be solved by survey propagation (SP). Particularly in hardware design and verification applications, satisfiability and other logical properties of a given propositional formula are sometimes decided based on a representation of the formula as a binary decision diagram (BDD).

Almost all SAT solvers include time-outs, so they will terminate in reasonable time even if they cannot find a solution. Different SAT solvers will find different instances easy or hard, and some excel at proving unsatisfiability, and others at finding solutions. All of these behaviors can be seen in the SAT solving contests.

Notes

- [1] (pdf) (http://www.mathnet.ru/php/getFT.phtml?jrnid=ppi&paperid=914&volume=9&year=1973&issue=3&fpage=115&what=fullt&option_lang=eng), translated into English by
- [2] The SAT problem for *arbitrary* formulas is NP-complete, too, since it is easily shown to be in NP, and it cannot be easier than SAT for CNF formulas.
- [3] ; here: Thm.10.4
- [4] i.e. such that one literal isn't the negation of the other
- [5] Aho, Hopcroft, Ullman UNIQ-ref-0-a68bc4e06c66b481-QINU (1974); Thm.10.5
- [6] viz. all maxterms that can be built with d_1, \dots, d_k , except " $d_1 \vee \dots \vee d_k$ "
- [7] Schaefer, 1978), p.222, Lemma 3.5
- [8] R. E. Bryant, S. M. German, and M. N. Velev, Microprocessor Verification Using Efficient Decision Procedures for a Logic of Equality with Uninterpreted Functions (<http://portal.acm.org/citation.cfm?id=709275>), in Analytic Tableaux and Related Methods, pp. 1–13, 1999.
- [9] "An improved exponential-time algorithm for k-SAT" (<http://dl.acm.org/citation.cfm?id=1066101>), Paturi, Pudlak, Saks, Zani
- [10] <http://minisat.se/>
- [11] <http://www.satcompetition.org/>
- [12] http://www.st.ewi.tudelft.nl/sat/march_dl.php

References

References are ordered by date of publication:

- Michael R. Garey and David S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. ISBN 0-7167-1045-5. A9.1: LO1 – LO7, pp. 259 – 260.
- Marques-Silva, J.; Glass, T. (1999). "Combinational equivalence checking using satisfiability and recursive learning". *Design, Automation and Test in Europe Conference and Exhibition, 1999. Proceedings (Cat. No. PR00078)*. p. 145. doi: 10.1109/DATE.1999.761110 (<http://dx.doi.org/10.1109/DATE.1999.761110>). ISBN 0-7695-0078-1.
- Clarke, E.; Biere, A.; Raimi, R.; Zhu, Y. (2001). *Formal Methods in System Design* **19**: 7. doi: 10.1023/A:1011276507260 (<http://dx.doi.org/10.1023/A:1011276507260>).
- Giunchiglia, E.; Tacchella, A. (2004). Giunchiglia, Enrico; Tacchella, Armando, eds. *Theory and Applications of Satisfiability Testing*. Lecture Notes in Computer Science **2919**. doi: 10.1007/b95238 (<http://dx.doi.org/10.1007/b95238>). ISBN 978-3-540-20851-8.
- Babic, D.; Bingham, J.; Hu, A. J. (2006). "B-Cubing: New Possibilities for Efficient SAT-Solving". *IEEE Transactions on Computers* **55** (11): 1315. doi: 10.1109/TC.2006.175 (<http://dx.doi.org/10.1109/TC.2006.175>).

- Rodriguez, C.; Villagra, M.; Baran, B. (2007). "Asynchronous team algorithms for Boolean Satisfiability". *2007 2nd Bio-Inspired Models of Network, Information and Computing Systems*. p. 66. doi: 10.1109/BIMNICS.2007.4610083 (<http://dx.doi.org/10.1109/BIMNICS.2007.4610083>).
- Carla P. Gomes, Henry Kautz, Ashish Sabharwal, Bart Selman (2008). "Satisfiability Solvers". In Frank Van Harmelen, Vladimir Lifschitz, Bruce Porter. *Handbook of knowledge representation*. Foundations of Artificial Intelligence **3**. Elsevier. pp. 89–134. doi: 10.1016/S1574-6526(07)03002-7 ([http://dx.doi.org/10.1016/S1574-6526\(07\)03002-7](http://dx.doi.org/10.1016/S1574-6526(07)03002-7)). ISBN 978-0-444-52211-5.

External links

More information on SAT:

- SAT and MAX-SAT for the Lay-researcher (<http://www.mqasem.net/sat/sat>)

SAT Applications:

- WinSAT v2.04 (<http://www.mqasem.net/sat/winsat/>): A Windows-based SAT application made particularly for researchers.

SAT Solvers:

- Chaff (<http://www.princeton.edu/~chaff/>)
- MiniSat (<http://minisat.se/>)
- lingeling/plingeling (<http://fmv.jku.at/lingeling/>)
- Sat4j (<http://www.sat4j.org/>)
- RSat (<http://reasoning.cs.ucla.edu/rsat/home.html>)
- UBCSAT (<http://www.satlib.org/ubcsat/>)
- Fast SAT Solver (<http://dudka.cz/fss>) - simple but fast implementation of SAT solver based on genetic algorithms
- PicoSAT (<http://fmv.jku.at/picosat/>)
- CryptoMiniSat (<http://www.msoos.org/cryptominisat2>)
- HyperSAT (<http://www.domagoj-babic.com/index.php/ResearchProjects/HyperSAT>)
- Spear (<http://www.domagoj-babic.com/index.php/ResearchProjects/Spear>)

International Conference on Theory and Applications of Satisfiability Testing:

- SAT 2013 (<http://sat2013.cs.helsinki.fi/>)
- SAT 2012 (<http://sat2012.fbk.eu>)
- SAT 2011 (<http://www.lri.fr/SAT2011/>)
- SAT 2010 (<http://ie.technion.ac.il/SAT10/>)
- SAT 2009 (<http://www.cs.swansea.ac.uk/~csoliver/SAT2009/>)
- SAT 2008 (<http://www.wcs.uni-paderborn.de/cs/ag-klbue/en/workshops/sat-08/sat08-main.php>)
- SAT 2007 (<http://sat07.ecs.soton.ac.uk/>)

Publications:

- Journal on Satisfiability, Boolean Modeling and Computation (<http://jsat.ewi.tudelft.nl>)
- Survey Propagation (<http://www.ictp.trieste.it/~zecchina/SP/>)

Benchmarks:

- Forced Satisfiable SAT Benchmarks (<http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/benchmarks.htm>)
- SATLIB (<http://www.satlib.org>)
- Software Verification Benchmarks (http://www.cs.ubc.ca/~babic/index_benchmarks.htm)
- Fadi Aloul SAT Benchmarks (<http://www.aloul.net/benchmarks.html>)

SAT solving in general:

- <http://www.satlive.org>
- <http://www.satisfiability.org>

Evaluation of SAT solvers:

- Yearly evaluation of SAT solvers (<http://www.maxsat.udl.cat/>)
- SAT solvers evaluation results for 2008 (<http://www.maxsat.udl.cat/08/ms08.pdf>)
- International SAT Competitions (<http://www.satcompetition.org>) History (<http://www.satcompetition.org/2002/onlinereport/node2.html>)

This article includes material from a column in the ACM SIGDA (<http://www.sigda.org>) e-newsletter (<http://www.sigda.org/newsletter/index.html>) by Prof. Karem Sakallah (<http://www.eecs.umich.edu/~karem>)

Original text is available here (http://www.sigda.org/newsletter/2006/eNews_061201.html)

Boolean-valued model

In mathematical logic, a **Boolean-valued model** is a generalization of the ordinary Tarskian notion of structure from model theory. In a Boolean-valued model, the truth values of propositions are not limited to "true" and "false", but instead take values in some fixed complete Boolean algebra.

Boolean-valued models were introduced by Dana Scott, Robert M. Solovay, and Petr Vopěnka in the 1960s in order to help understand Paul Cohen's method of forcing. They are also related to Heyting algebra semantics in intuitionistic logic.

Definition

Fix a complete Boolean algebra $B^{[1]}$ and a first-order language L ; the signature of L will consist of a collection of constant symbols, function symbols, and relation symbols.

A Boolean-valued model for the language L consists of a universe M , which is a set of elements (or **names**), together with interpretations for the symbols. Specifically, the model must assign to each constant symbol of L an element of M , and to each n -ary function symbol f of L and each n -tuple $\langle a_0, \dots, a_{n-1} \rangle$ of elements of M , the model must assign an element of M to the term $f(a_0, \dots, a_{n-1})$.

Interpretation of the atomic formulas of L is more complicated. To each pair a and b of elements of M , the model must assign a truth value $\|a=b\|$ to the expression $a=b$; this truth value is taken from the Boolean algebra B . Similarly, for each n -ary relation symbol R of L and each n -tuple $\langle a_0, \dots, a_{n-1} \rangle$ of elements of M , the model must assign an element of B to be the truth value $\|R(a_0, \dots, a_{n-1})\|$.

Interpretation of other formulas and sentences

The truth values of the atomic formulas can be used to reconstruct the truth values of more complicated formulas, using the structure of the Boolean algebra. For propositional connectives, this is easy; one simply applies the corresponding Boolean operators to the truth values of the subformulae. For example, if $\varphi(x)$ and $\psi(y, z)$ are formulas with one and two free variables, respectively, and if a, b, c are elements of the model's universe to be substituted for x, y , and z , then the truth value of

$$\phi(a) \wedge \psi(b, c)$$

is simply

$$\|\phi(a) \wedge \psi(b, c)\| = \|\phi(a)\| \wedge \|\psi(b, c)\|$$

The completeness of the Boolean algebra is required to define truth values for quantified formulas. If $\varphi(x)$ is a formula with free variable x (and possibly other free variables that are suppressed), then

$$\|\exists x\phi(x)\| = \bigvee_{a \in M} \|\phi(a)\|,$$

where the right-hand side is to be understood as the supremum in B of the set of all truth values $\|\varphi(a)\|$ as a ranges over M .

The truth value of a formula is sometimes referred to as its probability. However, these are not probabilities in the ordinary sense, because they are not real numbers, but rather elements of the complete Boolean algebra B .

Boolean-valued models of set theory

Given a complete Boolean algebra B there is a Boolean-valued model denoted by V^B , which is the Boolean-valued analogue of the von Neumann universe V . (Strictly speaking, V^B is a proper class, so we need to reinterpret what it means to be a model appropriately.) Informally, the elements of V^B are "Boolean-valued sets". Given an ordinary set A , every set either is or is not a member; but given a Boolean-valued set, every set has a certain, fixed "probability" of being a member of A . Again, the "probability" is an element of B , not a real number. The concept of Boolean-valued sets resembles, but is not the same as, the notion of a fuzzy set.

The ("probabilistic") elements of the Boolean-valued set, in turn, are also Boolean-valued sets, whose elements are also Boolean-valued sets, and so on. In order to obtain a non-circular definition of Boolean-valued set, they are defined inductively in a hierarchy similar to the cumulative hierarchy. For each ordinal α of V , the set V_α^B is defined as follows.

- V_0^B is the empty set.
- $V_{\alpha+1}^B$ is the set of all functions from V_α^B to B . (Such a function represents a "probabilistic" subset of V_α^B ; if f is such a function, then for any $x \in V_\alpha^B$, $f(x)$ is the probability that x is in the set.)
- If α is a limit ordinal, V_α^B is the union of V_β^B for $\beta < \alpha$

The class V^B is defined to be the union of all sets V_α^B .

It is also possible to relativize this entire construction to some transitive model M of ZF (or sometimes a fragment thereof). The Boolean-valued model M^B is obtained by applying the above construction *inside* M . The restriction to transitive models is not serious, as the Mostowski collapsing theorem implies that every "reasonable" (well-founded, extensional) model is isomorphic to a transitive one. (If the model M is not transitive things get messier, as M 's interpretation of what it means to be a "function" or an "ordinal" may differ from the "external" interpretation.)

Once the elements of V^B have been defined as above, it is necessary to define B -valued relations of equality and membership on V^B . Here a B -valued relation on V^B is a function from $V^B \times V^B$ to B . To avoid confusion with the usual equality and membership, these are denoted by $\|x=y\|$ and $\|x \in y\|$ for x and y in V^B . They are defined as follows:

$\|x \in y\|$ is defined to be $\sum_{t \in \text{Dom}(y)} \|x=t\| \wedge y(t)$ (" x is in y if it is equal to something in y ").

$\|x=y\|$ is defined to be $\|x \subseteq y\| \wedge \|y \subseteq x\|$ (" x equals y if x and y are both subsets of each other"), where

$\|x \subseteq y\|$ is defined to be $\prod_{t \in \text{Dom}(x)} x(t) \Rightarrow \|t \in y\|$ (" x is a subset of y if all elements of x are in y ")

The symbols \sum and \prod denote the least upper bound and greatest lower bound operations, respectively, in the complete Boolean algebra B . At first sight the definitions above appear to be circular: $\| \in \|$ depends on $\|= \|$, which depends on $\| \subseteq \|$, which depends on $\| \in \|$. However, a close examination shows that the definition of $\| \in \|$ only depends on $\| \in \|$ for elements of smaller rank, so $\| \in \|$ and $\|= \|$ are well defined functions from $V^B \times V^B$ to B .

It can be shown that the B -valued relations $\| \in \|$ and $\|= \|$ on V^B make V^B into a Boolean-valued model of set theory. Each sentence of first order set theory with no free variables has a truth value in B ; it must be shown that the axioms for equality and all the axioms of ZF set theory (written without free variables) have truth value 1 (the largest element of B). This proof is straightforward, but it is long because there are many different axioms that need to be checked.

Relationship to forcing

Set theorists use a technique called forcing to obtain independence results and to construct models of set theory for other purposes. The method was originally developed by Paul Cohen but has been greatly extended since then. In one form, forcing "adds to the universe" a generic subset of a poset, the poset being designed to impose interesting properties on the newly-added object. The wrinkle is that (for interesting posets) it can be proved that there simply *is* no such generic subset of the poset. There are three usual ways of dealing with this:

- **syntactic forcing** A *forcing relation* $p \Vdash \phi$ is defined between elements p of the poset and formulas ϕ of the *forcing language*. This relation is defined syntactically and has no semantics; that is, no model is ever produced. Rather, starting with the assumption that ZFC (or some other axiomatization of set theory) proves the independent statement, one shows that ZFC must also be able to prove a contradiction. However, the forcing is "over V "; that is, it is not necessary to start with a countable transitive model. See Kunen (1980) for an exposition of this method.
- **countable transitive models** One starts with a countable transitive model M of as much of set theory as is needed for the desired purpose, and that contains the poset. Then there *do* exist filters on the poset that are generic *over* M ; that is, that meet all dense open subsets of the poset that happen also to be elements of M .
- **fictional generic objects** Commonly, set theorists will simply *pretend* that the poset has a subset that is generic over all of V . This generic object, in nontrivial cases, cannot be an element of V , and therefore "does not really exist". (Of course, it is a point of philosophical contention whether *any* sets "really exist", but that is outside the scope of the current discussion.) Perhaps surprisingly, with a little practice this method is useful and reliable, but it can be philosophically unsatisfying.

Boolean-valued models and syntactic forcing

Boolean-valued models can be used to give semantics to syntactic forcing; the price paid is that the semantics is not 2-valued ("true or false"), but assigns truth values from some complete Boolean algebra. Given a forcing poset P , there is a corresponding complete Boolean algebra B , often obtained as the collection of regular open subsets of P , where the topology on P is generated by *cones* (sets of the form $\{q | q \leq p\}$, for fixed p). (Other approaches to constructing B are discussed below.)

Now the order on B (after removing the zero element) can replace P for forcing purposes, and the forcing relation can be interpreted semantically by saying that, for p an element of B and ϕ a formula of the forcing language,

$$p \Vdash \phi \iff p \leq \|\phi\|$$

where $\|\phi\|$ is the truth value of ϕ in V^B .

This approach succeeds in assigning a semantics to forcing over V without resorting to fictional generic objects. The disadvantages are that the semantics is not 2-valued, and that the combinatorics of B are often more complicated than those of the underlying poset P .

Boolean-valued models and generic objects over countable transitive models

One interpretation of forcing starts with a countable transitive model M of ZF set theory, a partially ordered set P , and a "generic" subset G of P , and constructs a new model of ZF set theory from these objects. (The conditions that the model be countable and transitive simplify some technical problems, but are not essential.) Cohen's construction can be carried out using Boolean-valued models as follows.

- Construct a complete Boolean algebra B as the complete Boolean algebra "generated by" the poset P .
- Construct an ultrafilter U on B (or equivalently a homomorphism from B to the Boolean algebra {true, false}) from the generic subset G of P .
- Use the homomorphism from B to {true, false} to turn the Boolean-valued model M^B of the section above into an ordinary model of ZF.

We now explain these steps in more detail.

For any poset P there is a complete Boolean algebra B and a map e from P to B^+ (the non-zero elements of B) such that the image is dense, $e(p) \leq e(q)$ whenever $p \leq q$, and $e(p)e(q)=0$ whenever p and q are incompatible. This Boolean algebra is unique up to isomorphism. It can be constructed as the algebra of regular open sets in the topological space of P (with underlying set P , and a base given by the sets U_p of elements q with $q \leq p$).

The map from the poset P to the complete Boolean algebra B is not injective in general. The map is injective if and only if P has the following property: if every $r \leq p$ is compatible with q , then $p \leq q$.

The ultrafilter U on B is defined to be the set of elements b of B that are greater than some element of (the image of) G . Given an ultrafilter U on a Boolean algebra, we get a homomorphism to $\{\text{true}, \text{false}\}$ by mapping U to true and its complement to false. Conversely, given such a homomorphism, the inverse image of true is an ultrafilter, so ultrafilters are essentially the same as homomorphisms to $\{\text{true}, \text{false}\}$. (Algebraists might prefer to use maximal ideals instead of ultrafilters: the complement of an ultrafilter is a maximal ideal, and conversely the complement of a maximal ideal is an ultrafilter.)

If g is a homomorphism from a Boolean algebra B to a Boolean algebra C and M^B is any B -valued model of ZF (or of any other theory for that matter) we can turn M^B into a C -valued model by applying the homomorphism g to the value of all formulas. In particular if C is $\{\text{true}, \text{false}\}$ we get a $\{\text{true}, \text{false}\}$ -valued model. This is almost the same as an ordinary model: in fact we get an ordinary model on the set of equivalence classes under $\| = \|$ of a $\{\text{true}, \text{false}\}$ -valued model. So we get an ordinary model of ZF set theory by starting from M , a Boolean algebra B , and an ultrafilter U on B . (The model of ZF constructed like this is not transitive. In practice one applies the Mostowski collapsing theorem to turn this into a transitive model.)

We have seen that forcing can be done using Boolean-valued models, by constructing a Boolean algebra with ultrafilter from a poset with a generic subset. It is also possible to go back the other way: given a Boolean algebra B , we can form a poset P of all the nonzero elements of B , and a generic ultrafilter on B restricts to a generic set on P . So the techniques of forcing and Boolean-valued models are essentially equivalent.

Notes

- [1] B here is assumed to be *nondegenerate*; that is, 0 and 1 must be distinct elements of B . Authors writing on Boolean-valued models typically take this requirement to be part of the definition of "Boolean algebra", but authors writing on Boolean algebras in general often do not.

References

- Bell, J. L. (1985) *Boolean-Valued Models and Independence Proofs in Set Theory*, Oxford. ISBN 0-19-853241-5
- Grishin, V.N. (2001), "b/b016990" (<http://www.encyclopediaofmath.org/index.php?title=b/b016990>), in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Jech, Thomas (2002). *Set theory, third millennium edition (revised and expanded)*. Springer. ISBN 3-540-44085-2. OCLC 174929965 (<http://www.worldcat.org/oclc/174929965>).
- Kunen, Kenneth (1980). *Set Theory: An Introduction to Independence Proofs*. North-Holland. ISBN 0-444-85401-0. OCLC 12808956 (<http://www.worldcat.org/oclc/12808956>).
- Kusraev, A. G. and S. S. Kutateladze (1999). *Boolean Valued Analysis*. Kluwer Academic Publishers. ISBN 0-7923-5921-6. OCLC 41967176 (<http://www.worldcat.org/oclc/41967176>). Contains an account of Boolean-valued models and applications to Riesz spaces, Banach spaces and algebras.
- Manin, Yu. I. (1977). *A Course in Mathematical Logic*. Springer. ISBN 0-387-90243-0. OCLC 2797938 (<http://www.worldcat.org/oclc/2797938>). Contains an account of forcing and Boolean-valued models written for mathematicians who are not set theorists.

Booleo

Booleo is a strategy card game using boolean logic gates. It was developed by Jonathan Brandt and Chris Kampf with Sean P. Dennis in 2008, and it was first published by **Tessera Games LLC** in 2009.^[1]

The game

The deck consists of 64 cards:

- 48 “Gate” cards using three Boolean operators AND, OR, and XOR
 - 8 **OR** cards resolving to 1
 - 8 **OR** cards resolving to 0
 - 8 **AND** cards resolving to 1
 - 8 **AND** cards resolving to 0
 - 8 **XOR** cards resolving to 1
 - 8 **XOR** cards resolving to 0
- 8 **NOT** cards
- 6 **Initial Binary** cards, each displaying a “0” and a “1” aligned to the two short ends of the card
- 2 **Truth Tables** (used for reference, not in play)

Play

Starting with up a line of Initial Binary cards laid perpendicular to two facing players, the object of the game is to be the first to complete a logical pyramid whose final output equals that of the righmost **Initial Binary** card facing that player.

The game is played in “draw one play one” format. The pyramid consists of decreasing rows of gate cards, where the outputs of any contiguous pair of cards comprise the input values to a single card in the following row. The pyramid, therefore, has **Initial Binary** values as its base and tapers to a single card closest to the player. By tracing the “flow” of values through any series of gate, every card placed in the pyramid must make “logical sense”, i.e. the inputs and output value of every gate card must conform to the rule of that gate card.

The **NOT** cards are played against any of the Initial Binary cards in play, causing that card to be rotated 180 degrees, literally “flipping” the value of that card from 0 to 1 or vice versa.

By changing the value of any **Initial Binary**, any and all gate cards which “flow” from it must be re-evaluated to ensure its placement makes “logical sense”. If it does not, that gate card is removed from the player’s pyramid.

Since both player’s pyramids share the Initial Binary cards as a base, “flipping” an **Initial Binary** has an effect on both player’s pyramids. *A principal strategy during game play is to invalidate gate cards in the opponent’s logic pyramid while rendering as little damage to one’s own pyramid in the process.*

Some logic gates are more robust than others to a change to their inputs. Therefore, *not all logic gate cards have the same strategic value.*

Variations

The number of cards in **bOOleO** will comfortably support a match between two players whose logic pyramids are 6 cards wide at their base. By combining decks, it is possible to construct larger pyramids or to have matches among more than 2 players. For example:

- Four players may play individually or as facing teams by arranging a cross of Initial Binary Cards, where four logic pyramids extend like compass points in four directions
- Four or more players may build partially overlapping pyramids from a long base of Initial Binary Cards

Tessera Games also published **bOOleO-N Edition**. This is identical to **bOOleO** with the exception that it uses the inverse set of logic gates: NAND, NOR, and XNOR. **bOOleO-N Edition** may be played on its own, or it may be combined with **bOOleO**.

References

[1] <http://boardgamegeek.com/boardgame/40943/booleo>

Chaff algorithm

Chaff is an algorithm for solving instances of the Boolean satisfiability problem in programming. It was designed by researchers at Princeton University, USA. The algorithm is an instance of the DPLL algorithm with a number of enhancements for efficient implementation.

Implementations

Some available implementations of the algorithm in software are mChaff and **zChaff**, the latter one being the most widely known and used. zChaff was originally written by Dr. Lintao Zhang, now at Microsoft Research, hence the “z”. It is now maintained by researchers at Princeton University and available for download as both source code and binaries on Linux. zChaff is free for non-commercial use.

References

- M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, S. Malik. *Chaff: Engineering an Efficient SAT Solver*^[1], 39th Design Automation Conference (DAC 2001), Las Vegas, ACM 2001.

External links

- Web page about zChaff^[2]

References

[1] <http://www.princeton.edu/~chaff/publication/DAC2001v56.pdf>

[2] <http://www.princeton.edu/~chaff/zchaff.html>

Correlation immunity

In mathematics, the **correlation immunity** of a Boolean function is a measure of the degree to which its outputs are uncorrelated with some subset of its inputs. Specifically, a Boolean function is said to be correlation-immune of *order m* if every subset of *m* or fewer variables in x_1, x_2, \dots, x_n is statistically independent of the value of $f(x_1, x_2, \dots, x_n)$.

Definition

A function $f : \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ is *k*-th order correlation immune if for any independent *n* binary random variables $X_0 \dots X_{n-1}$, the random variable $Z = f(X_0, \dots, X_{n-1})$ is independent from any random vector $(X_{i_1} \dots X_{i_k})$ with $0 \leq i_1 < \dots < i_k < n$.

Results in cryptography

When used in a stream cipher as a combining function for linear feedback shift registers, a Boolean function with **low-order** correlation-immunity is **more susceptible** to a correlation attack than a function with correlation immunity of **high order**.

Siegenthaler showed that the correlation immunity *m* of a Boolean function of algebraic degree *d* of *n* variables satisfies $m + d \leq n$; for a given set of input variables, this means that a high algebraic degree will restrict the maximum possible correlation immunity. Furthermore, if the function is balanced then $m + d \leq n - 1$.

References

Davis–Putnam algorithm

The **Davis–Putnam algorithm** was developed by Martin Davis and Hilary Putnam for checking the validity of a first-order logic formula using a resolution-based decision procedure for propositional logic. Since the set of valid first-order formulas is recursively enumerable but not recursive, there exists no general algorithm to solve this problem. Therefore, the Davis–Putnam algorithm only terminates on valid formulas. Today, the term "Davis–Putnam algorithm" is often used synonymously with the resolution-based propositional decision procedure that is actually only one of the steps of the original algorithm.

The procedure is based on Herbrand's theorem, which implies that an unsatisfiable formula has an unsatisfiable ground instance, and on the fact that a formula is valid if and only if its negation is unsatisfiable. Taken together, these facts imply that to prove the validity of φ it is enough to prove that a ground instance of $\neg\varphi$ is unsatisfiable. If φ is not valid, then the search for an unsatisfiable ground instance will not terminate.

The procedure roughly consists of these three parts:

- put the formula in prenex form and eliminate quantifiers
- generate one by one all propositional ground instances
- and check for each if it is satisfiable

The last part is probably the most innovative one, and works as follows:

- for every variable in the formula
 - for every clause *c* containing the variable and every clause *n* containing the negation of the variable
 - resolve *c* and *n* and add the resolvent to the formula
 - remove all original clauses containing the variable or its negation

At each step, the intermediate formula generated is equisatisfiable to the original formula, but it does not retain equivalence. The resolution step leads to a worst-case exponential blow-up in the size of the formula. The DPLL algorithm is a refinement of the propositional satisfiability step of the Davis–Putnam procedure, that requires only a linear amount of memory in the worst case.

References

- Davis, Martin; Putnam, Hilary (1960). "A Computing Procedure for Quantification Theory" ^[1]. *Journal of the ACM* **7** (3): 201–215. doi:10.1145/321033.321034 ^[2].
- Davis, Martin; Logemann, George, and Loveland, Donald (1962). "A Machine Program for Theorem Proving" ^[3]. *Communications of the ACM* **5** (7): 394–397. doi:10.1145/368273.368557 ^[4].
- R. Dechter; I. Rish. "Directional Resolution: The Davis–Putnam Procedure, Revisited". In J. Doyle and E. Sandewall and P. Torasso. *Principles of Knowledge Representation and Reasoning: Proc. of the Fourth International Conference (KR'94)*. Kaufmann. pp. 134–145.
- John Harrison (2009). *Handbook of practical logic and automated reasoning*. Cambridge University Press. pp. 79–90. ISBN 978-0-521-89957-4.

References

- [1] <http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=321034>
[2] <http://dx.doi.org/10.1145%2F321033.321034>
[3] <http://portal.acm.org/citation.cfm?doid=368273.368557>
[4] <http://dx.doi.org/10.1145%2F368273.368557>

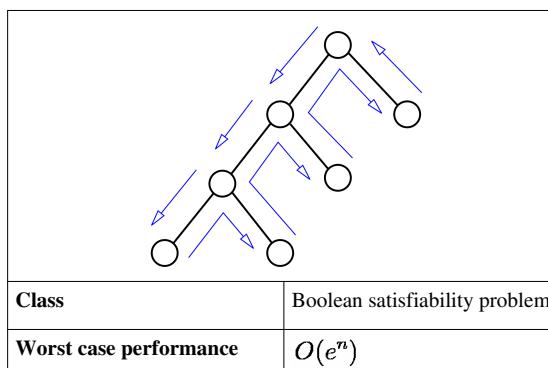
DPLL algorithm

The **Davis–Putnam–Logemann–Loveland (DPLL) algorithm** is a complete, backtracking-based search algorithm for deciding the satisfiability of propositional logic formulae in conjunctive normal form, i.e. for solving the CNF-SAT problem.

It was introduced in 1962 by Martin Davis, Hilary Putnam, George Logemann and Donald W. Loveland and is a refinement of the earlier Davis–Putnam algorithm, which is a resolution-based procedure developed by Davis and Putnam in 1960. Especially in older publications, the Davis–Logemann–Loveland algorithm is often referred to as the “Davis–Putnam method” or the “DP algorithm”. Other common names that maintain the distinction are DLL and DPLL.

DPLL is a highly efficient procedure and after almost 50 years still forms the basis for most efficient complete SAT solvers, as well as for many theorem provers for fragments of first-order logic.

DPLL



Worst case space complexity	$O(n)$ (basic algorithm)
-----------------------------	--------------------------

Implementations and applications

The SAT problem is important both from theoretical and practical points of view. In complexity theory it was the first problem proved to be NP-complete, and can appear in a broad variety of applications such as *model checking*, automated planning and scheduling, diagnosis in artificial intelligence.

As such, it was and still is a hot topic in research for many years, and competitions between SAT solvers regularly take place. DPLL's modern implementations like Chaff and zChaff, GRASP or Minisat are in the first places of the competitions these last years.

Another application which often involves DPLL is *automated theorem proving* or *Satisfiability Modulo Theories* (SMT) which is a SAT problem in which propositional variables are replaced with formulas of another mathematical theory.

The algorithm

The basic backtracking algorithm runs by choosing a literal, assigning a truth value to it, simplifying the formula and then recursively checking if the simplified formula is satisfiable; if this is the case, the original formula is satisfiable; otherwise, the same recursive check is done assuming the opposite truth value. This is known as the *splitting rule*, as it splits the problem into two simpler sub-problems. The simplification step essentially removes all clauses which become true under the assignment from the formula, and all literals that become false from the remaining clauses.

The DPLL algorithm enhances over the backtracking algorithm by the eager use of the following rules at each step:

Unit propagation

If a clause is a *unit clause*, i.e. it contains only a single unassigned literal, this clause can only be satisfied by assigning the necessary value to make this literal true. Thus, no choice is necessary. In practice, this often leads to deterministic cascades of units, thus avoiding a large part of the naive search space.

Pure literal elimination

If a propositional variable occurs with only one polarity in the formula, it is called *pure*. Pure literals can always be assigned in a way that makes all clauses containing them true. Thus, these clauses do not constrain the search anymore and can be deleted.

Unsatisfiability of a given partial assignment is detected if one clause becomes empty, i.e. if all its variables have been assigned in a way that makes the corresponding literals false. Satisfiability of the formula is detected either when all variables are assigned without generating the empty clause, or, in modern implementations, if all clauses are satisfied. Unsatisfiability of the complete formula can only be detected after exhaustive search.

The DPLL algorithm can be summarized in the following pseudocode, where Φ is the CNF formula:

Algorithm DPLL

Input: A set of clauses Φ .

Output: A Truth Value.

function DPLL(Φ)

```

if  $\Phi$  is a consistent set of literals
    then return true;
if  $\Phi$  contains an empty clause
    then return false;
for every unit clause  $l$  in  $\Phi$ 
     $\Phi \leftarrow \text{unit-propagate}(l, \Phi);$ 

```

```

for every literal  $l$  that occurs pure in  $\Phi$ 
     $\Phi \leftarrow \text{pure-literal-assign}(l, \Phi)$ ;
     $l \leftarrow \text{choose-literal}(\Phi)$ ;
return  $\text{DPLL}(\Phi \sqcup l)$  or  $\text{DPLL}(\Phi \sqcup \text{not}(l))$ ;

```

- " \leftarrow " is a shorthand for "changes to". For instance, " $largest \leftarrow item$ " means that the value of $largest$ changes to the value of $item$.
- "**return**" terminates the algorithm and outputs the value that follows.

In this pseudocode, $\text{unit-propagate}(l, \Phi)$ and $\text{pure-literal-assign}(l, \Phi)$ are functions that return the result of applying unit propagation and the pure literal rule, respectively, to the literal l and the formula Φ . In other words, they replace every occurrence of l with "true" and every occurrence of $\text{not } l$ with "false" in the formula Φ , and simplify the resulting formula. The pseudocode DPLL function only returns whether the final assignment satisfies the formula or not. In a real implementation, the partial satisfying assignment typically is also returned on success; this can be derived from the consistent set of literals of the first **if** statement of the function.

The Davis–Logemann–Loveland algorithm depends on the choice of *branching literal*, which is the literal considered in the backtracking step. As a result, this is not exactly an algorithm, but rather a family of algorithms, one for each possible way of choosing the branching literal. Efficiency is strongly affected by the choice of the branching literal: there exist instances for which the running time is constant or exponential depending on the choice of the branching literals. Such choice functions are also called heuristic functions or branching heuristics.

Current work

Current work on improving the algorithm has been done on three directions:

1. Defining different policies for choosing the branching literals.
2. Defining new data structures to make the algorithm faster, especially the part on **unit propagation**
3. Defining variants of the basic backtracking algorithm. The latter direction include *non-chronological backtracking* (aka. *backjumping*) and *clause learning*. These refinements describe a method of backtracking after reaching a conflict clause which "learns" the root causes (assignments to variables) of the conflict in order to avoid reaching the same conflict again.

A newer algorithm from 1990 is Stålmarck's method. Also since 1986 (reduced ordered) binary decision diagrams have also been used for SAT solving.

Relation to other notions

Runs of DPLL-based algorithms on unsatisfiable instances correspond to tree resolution refutation proofs.

References

General

- Davis, Martin; Putnam, Hilary (1960). "A Computing Procedure for Quantification Theory" ^[1]. *Journal of the ACM* **7** (3): 201–215. doi:10.1145/321033.321034 ^[2].
- Davis, Martin; Logemann, George, and Loveland, Donald (1962). "A Machine Program for Theorem Proving" ^[3]. *Communications of the ACM* **5** (7): 394–397. doi:10.1145/368273.368557 ^[4].
- Ouyang, Ming (1998). "How Good Are Branching Rules in DPLL?". *Discrete Applied Mathematics* **89** (1–3): 281–286. doi:10.1016/S0166-218X(98)00045-6 ^[1].
- John Harrison (2009). *Handbook of practical logic and automated reasoning*. Cambridge University Press. pp. 79–90. ISBN 978-0-521-89957-4.

Specific

[1] [http://dx.doi.org/10.1016/S0166-218X\(98\)00045-6](http://dx.doi.org/10.1016/S0166-218X(98)00045-6)

Further reading

- Malay Ganai; Aarti Gupta; Dr. Aarti Gupta (2007). *SAT-based scalable formal verification solutions*. Springer. pp. 23–32. ISBN 978-0-387-69166-4.
- Carla P. Gomes, Henry Kautz, Ashish Sabharwal, Bart Selman (2008). "Satisfiability Solvers". In Frank Van Harmelen, Vladimir Lifschitz, Bruce Porter. *Handbook of knowledge representation*. Foundations of Artificial Intelligence 3. Elsevier. pp. 89–134. doi: 10.1016/S1574-6526(07)03002-7 ([http://dx.doi.org/10.1016/S1574-6526\(07\)03002-7](http://dx.doi.org/10.1016/S1574-6526(07)03002-7)). ISBN 978-0-444-52211-5.

Formula game

A **formula game** is an artificial game represented by a fully quantified Boolean formula. Players' turns alternate and the space of possible moves is denoted by bound variables. If a variable is universally quantified, the formula following it has the same truth value as the formula beginning with the universal quantifier regardless of the move taken. If a variable is existentially quantified, the formula following it has the same truth value as the formula beginning with the existential quantifier for at least one move available at the turn. Turns alternate, and a player loses if he cannot move at his turn. In computational complexity theory, the language FORMULA-GAME is defined as all formulas Φ such that Player 1 has a winning strategy in the game represented by Φ . FORMULA-GAME is PSPACE-complete.

References

- Sipser, Michael. (2006). *Introduction to the Theory of Computation*. Boston: Thomson Course Technology.

Join (sigma algebra)

In mathematics, a **join** or **refinement** of two sigma algebras is the coarsest sigma algebra containing both.^[1]

Given two sigma sub-algebras $\mathcal{A} \subset \mathcal{C}$ and $\mathcal{B} \subset \mathcal{C}$, their join $\mathcal{A} \vee \mathcal{B} \subset \mathcal{C}$ is given by

$$\mathcal{A} \vee \mathcal{B} = \{A_i \cap B_j \mid A_i \in \mathcal{A}, B_j \in \mathcal{B}\}$$

The above defines the join on sub-sigma algebras of a single common sigma algebra \mathcal{C} , as it is not generally possible to define the join of two unrelated sigma algebras.

Likewise, given two partitions $Q = \{Q_1, \dots, Q_k\}$ and $R = \{R_1, \dots, R_m\}$ of a common set S , their join or refinement $Q \vee R$ is given as

$$Q \vee R = \{Q_i \cap R_j \mid i = 1, \dots, k, j = 1, \dots, m\}.$$

The two definitions are equivalent: The unions of elements from a partition of a set generate a sigma algebra. Likewise, intersections of elements from a sigma algebra result in a partition of a set.

References

[1] Peter Walters, *An Introduction to Ergodic Theory* (1982) Springer Verlag, ISBN 0-387-90599-5

Logic alphabet

The **logic alphabet** constitutes an iconic set of symbols that systematically represents the sixteen possible binary truth functions of logic. The logic alphabet was developed by Shea Zellweger. The major emphasis of his iconic "logic alphabet" is to provide a more cognitively ergonomic notation for logic. Zellweger's visually iconic system more readily reveals, to the novice and expert alike, the underlying symmetry relationships and geometric properties of the sixteen binary connectives within Boolean algebra.

Truth functions

Truth functions are functions from sequences of truth values to truth values. A unary truth function, for example, takes a single truth value and maps it onto another truth value. Similarly, a binary truth function maps ordered pairs of truth values onto truth values, while a ternary truth function maps ordered triples of truth values onto truth values, and so on.

In the unary case, there are two possible inputs, viz. **T** and **F**, and thus four possible unary truth functions: one mapping **T** to **T** and **F** to **F**, one mapping **T** to **F** and **F** to **F**, one mapping **T** to **T** and **F** to **T**, and finally one mapping **T** to **F** and **F** to **T**, this last one corresponding to the familiar operation of logical negation. In the form of a table, the four unary truth functions may be represented as follows.

Unary truth functions

p	p	F	T	$\neg p$
T	T	F	T	F
F	F	F	T	T

In the binary case, there are four possible inputs, viz. **(T,T)**, **(T,F)**, **(F,T)**, and **(F,F)**, thus yielding sixteen possible binary truth functions. Quite generally, for any number n , there are 2^{2^n} possible n -ary truth functions. The sixteen possible binary truth functions are listed in the table below.

Binary truth functions

p	q	T	NAND	\rightarrow	NOT p	\leftarrow	NOT q	\leftrightarrow	NOR	OR	XOR	q	NOT \leftarrow	p	NOT \rightarrow	AND	F
T	T	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
T	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
F	T	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F

The logic alphabet

Dr. Zellweger's logic alphabet offers a visually systematic way of representing each of the sixteen binary truth functions. The idea behind the logic alphabet is to first represent the sixteen binary truth functions in the form of a square matrix rather than the more familiar tabular format seen in the table above, and then to assign a letter shape to each of these matrices. Letter shapes are derived from the distribution of **T**s in the matrix. When drawing a logic symbol, one passes through each square with assigned **F** values while stopping in a square with assigned **T** values. In the extreme examples, the symbol for tautology is a **X** (stops in all four squares), while the symbol for contradiction is an **O** (passing through all squares without stopping). The square matrix corresponding to each binary truth function, as well as its corresponding letter shape, are displayed in the table below.

The logic alphabet

Significance

The interest of the logic alphabet lies in its aesthetic, symmetric, and geometric qualities that allow an individual to more easily, rapidly and visually manipulate the relationships between entire truth tables. For example, by reflecting the symbol for NAND (viz. 'h') across the vertical axis we produce the symbol for \leftarrow , whereas by reflecting it across the horizontal axis we produce the symbol for \rightarrow , and by reflecting it across both the horizontal and vertical axes we produce the symbol for \vee . Similar geometrical transformation can be obtained by operating upon the other symbols. Indeed, Zellweger has constructed intriguing structures involving the symbols of the logic alphabet on the basis of these symmetries ([1] [2]). The considerable aesthetic appeal of the logic alphabet has led to exhibitions of Zellweger's work at the Museum of Jurassic Technology in Los Angeles, among other places.

The value of the logic alphabet lies in its use as a visually simpler pedagogical tool than the traditional system for logic notation. The logic alphabet eases the introduction to the fundamentals of logic, especially for children, at much earlier stages of cognitive development. Because the logic notation system, in current use today, is so deeply embedded in our computer culture, the "logic alphabets" adoption and value by the field of logic itself, at this juncture, is questionable. Additionally, systems of natural deduction, for example, generally require introduction and elimination rules for each connective, meaning that the use of all sixteen binary connectives would result in a highly complex proof system. Various subsets of the sixteen binary connectives (e.g., $\{\vee, \&, \rightarrow, \sim\}$, $\{\vee, \sim\}$, $\{\&, \sim\}$, $\{\rightarrow, \sim\}$) are themselves functionally complete in that they suffice to define the remaining connectives. In fact, both NAND and NOR are sole sufficient operators, meaning that the remaining connectives can all be defined solely in terms of either of them.

External links

- Page dedicated to Zellweger's logic alphabet ^[3]
- Exhibition in a small museum: Flickr photopage ^[4], including a discussion between Tilman Piesk and probably Shea Zellweger

References

- [1] http://www.logic-alphabet.net/images/logicbug_2345_2.jpg
- [2] http://www.logic-alphabet.net/images/clockcompass_2353_2.jpg
- [3] <http://www.logic-alphabet.net/>
- [4] <http://www.flickr.com/photos/43992178@N00/387339135/>

Logic redundancy

Logic redundancy occurs in a digital gate network containing circuitry that does not affect the static logic function. There are several reasons why logic redundancy may exist. One reason is that it may have been added deliberately to suppress transient glitches (thus causing a race condition) in the output signals by having two or more product terms overlap with a third one.

Consider the following equation:

$$Y = AB + \overline{A}C + BC.$$

The third product term BC is a redundant consensus term. If A switches from 1 to 0 while $B = 1$ and $C = 1$, Y remains 1. During the transition of signal A in logic gates, both the first and second term may be 0 momentarily. The third term prevents a glitch since its value of 1 in this case is not affected by the transition of signal A .

Another reason for logic redundancy is poor design practices which unintentionally result in logically redundant terms. This causes an unnecessary increase in network complexity, and possibly hampering the ability to test manufactured designs using traditional test methods (single stuck-at fault models). (Note: testing might be possible using IDDQ models.)

Removing logic redundancy

Logic redundancy is, in general, not desired. Redundancy, by definition, requires extra parts (in this case: logical terms) which raises the cost of implementation (either actual cost of physical parts or CPU time to process). Logic redundancy can be removed by several well-known techniques, such as Karnaugh maps, the Quine–McCluskey algorithm, and the heuristic computer method.

Adding logic redundancy

In some cases it may be desirable to *add* logic redundancy. One of those cases is to avoid race conditions whereby an output can fluctuate because different terms are "racing" to turn off and on. To explain this in more concrete terms the Karnaugh map to the right shows the minterms and maxterms for the following function:

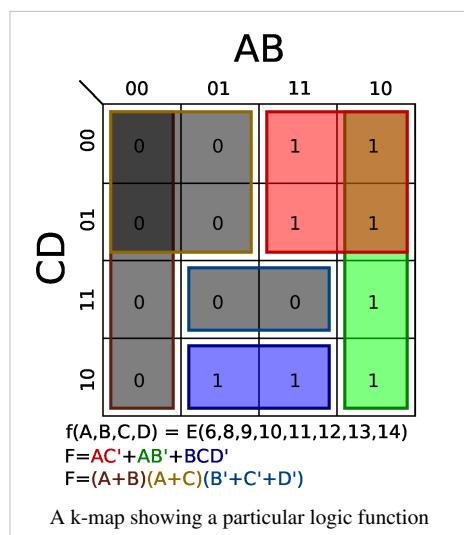
$$f(A, B, C, D) = E(6, 8, 9, 10, 11, 12, 13, 14).$$

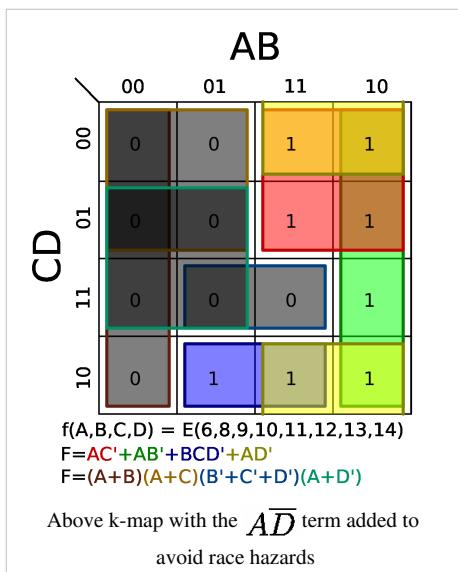
The boxes represent the minimal AND/OR terms needed to implement this function:

$$F = A\overline{C} + A\overline{B} + BCD.$$

The k-map visually shows where race conditions occur in the minimal expression by having gaps between minterms or gaps between maxterms. For example, the gap between the blue and green rectangles.

If the input $ABCD = 1110$ were to change to $ABCD = 1010$ then a race will occur between BCD turning off and $A\overline{B}$ turning off. If the blue term switches off before the green turns on then the output will fluctuate and may register as 0. Another race condition is between the blue and the red for transition of $ABCD = 1110$ to $ABCD = 1100$.





The race condition is removed by adding in logic redundancy, which is contrary to the aims of using a k-map in the first place. Both minterm race conditions are covered by addition of the yellow term $A\bar{D}$. (The maxterm race condition is covered by addition of the green-bordered grey term $A + \bar{D}$.)

In this case, the addition of logic redundancy has stabilized the output to avoid output fluctuations because terms are racing each other to change state.

Logical matrix

A **logical matrix**, **binary matrix**, **relation matrix**, **Boolean matrix**, or **(0,1) matrix** is a matrix with entries from the Boolean domain $\mathbf{B} = \{0, 1\}$. Such a matrix can be used to represent a binary relation between a pair of finite sets.

Matrix representation of a relation

If R is a binary relation between the finite indexed sets X and Y (so $R \subseteq X \times Y$), then R can be represented by the adjacency matrix M whose row and column indices index the elements of X and Y , respectively, such that the entries of M are defined by:

$$M_{i,j} = \begin{cases} 1 & (x_i, y_j) \in R \\ 0 & (x_i, y_j) \notin R \end{cases}$$

In order to designate the row and column numbers of the matrix, the sets X and Y are indexed with positive integers: i ranges from 1 to the cardinality of X and j ranges from 1 to the cardinality of Y . (See the entry on indexed sets for more detail.)

Example

The binary relation R on the set $\{1, 2, 3, 4\}$ is defined so that aRb holds if and only if a divides b evenly, with no remainder. For example, $2R4$ holds because 2 divides 4 without leaving a remainder, but $3R4$ does not hold because when 3 divides 4 there is a remainder of 1. The following set is the set of pairs for which the relation R holds.

$$\{(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 4), (3, 3), (4, 4)\}.$$

The corresponding representation as a Boolean matrix is:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Other examples

- A permutation matrix is a (0,1)-matrix, all of whose columns and rows each have exactly one nonzero element.
 - A Costas array is a special case of a permutation matrix
- An incidence matrix in combinatorics and finite geometry has ones to indicate incidence between points (or vertices) and lines of a geometry, blocks of a block design, or edges of a graph (mathematics)
- A design matrix in analysis of variance is a (0,1)-matrix with constant row sums.
- An adjacency matrix in graph theory is a matrix whose rows and columns represent the vertices and whose entries represent the edges of the graph. The adjacency matrix of a simple, undirected graph is a binary symmetric matrix with zero diagonal.
- The biadjacency matrix of a simple, undirected bipartite graph is a (0,1)-matrix, and any (0,1)-matrix arises in this way.
- The prime factors of a list of m square-free, n -smooth numbers can be described as a $m \times \pi(n)$ (0,1)-matrix, where π is the prime-counting function and a_{ij} is 1 if and only if the j th prime divides the i th number. This representation is useful in the quadratic sieve factoring algorithm.
- A bitmap image containing pixels in only two colors can be represented as a (0,1)-matrix in which the 0's represent pixels of one color and the 1's represent pixels of the other color.
- Using binary matrix to check the game rules in the game of Go [1]

Some properties

The matrix representation of the equality relation on a finite set is an identity matrix, that is, one whose entries on the diagonal are all 1, while the others are all 0.

If the Boolean domain is viewed as a semiring, where addition corresponds to logical OR and multiplication to logical AND, the matrix representation of the composition of two relations is equal to the matrix product of the matrix representations of these relation.

Frequently operations on binary matrices are defined in terms of modular arithmetic mod 2—that is, the elements are treated as elements of the Galois field $\text{GF}(2) = \mathbb{F}_2$. They arise in a variety of representations and have a number of more restricted special forms.

The number of distinct m -by- n binary matrices is equal to 2^{mn} , and is thus finite.

References

- Hogben, Leslie (2006), *Handbook of Linear Algebra (Discrete Mathematics and Its Applications)*, Boca Raton: Chapman & Hall/CRC, ISBN 978-1-58488-510-8, section 31.3, Binary Matrices
- Kim, Ki Hang, *Boolean Matrix Theory and Applications*, ISBN 0-8247-1788-0

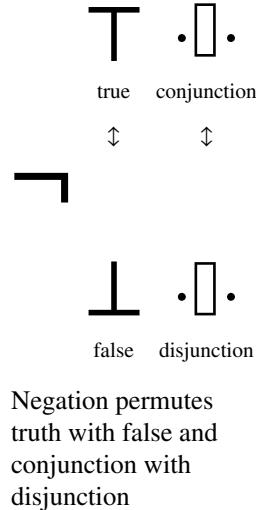
References

[1] <http://senseis.xmp.net/?BinMatrix>

Logical value

In logic and mathematics, a **truth value**, sometimes called a **logical value**, is a value indicating the relation of a proposition to truth.

Classical logic



Negation permutes

truth with false and
conjunction with
disjunction

In classical logic, with its intended semantics, the truth values are **true** (1 or T) and **false** (0 or L); that is, classical logic is a two-valued logic. This set of two values is also called the Boolean domain. Corresponding semantics of logical connectives are truth functions, whose values are expressed in the form of truth tables. Logical biconditional becomes the equality binary relation, and negation becomes a bijection which permutes true and false. Conjunction and disjunction are dual with respect to negation, which is expressed by De Morgan's laws:

$$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$$

$$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$$

Propositional variables become variables in the Boolean domain. Assigning values for propositional variables is referred to as valuation.

Multi-valued logic

Multi-valued logics (such as fuzzy logic and relevance logic) allow for more than two truth values, possibly containing some internal structure. For example, on the unit interval [0,1] such structure is a total order; this may be expressed as existence of various degrees of truth.

Algebraic semantics

Not all logical systems are truth-valuational in the sense that logical connectives may be interpreted as truth functions. For example, intuitionistic logic lacks a complete set of truth values because its semantics, the Brouwer–Heyting–Kolmogorov interpretation, is specified in terms of provability conditions, and not directly in terms of the necessary truth of formulae.

But even non-truth-valuational logics can associate values with logical formulae, as is done in algebraic semantics. The algebraic semantics of intuitionistic logic is given in terms of Heyting algebras, compared to Boolean algebra semantics of classical propositional calculus.

In other theories

Intuitionistic type theory uses types in the place of truth values.

Topos theory uses truth values in a special sense: the truth values of a topos are the global elements of the subobject classifier. Having truth values in this sense does not make a logic truth valutional.

External links

- Truth Values^[1] entry by Yaroslav Shramko, Heinrich Wansing in the *Stanford Encyclopedia of Philosophy*

References

[1] <http://plato.stanford.edu/entries/truth-values>

Modal algebra

In algebra and logic, a **modal algebra** is a structure $\langle A, \wedge, \vee, -, 0, 1, \square \rangle$ such that

- $\langle A, \wedge, \vee, -, 0, 1 \rangle$ is a Boolean algebra,
- \square is a unary operation on A satisfying $\square 1 = 1$ and $\square(x \wedge y) = \square x \wedge \square y$ for all x, y in A .

Modal algebras provide models of propositional modal logics in the same way as Boolean algebras are models of classical logic. In particular, the variety of all modal algebras is the equivalent algebraic semantics of the modal logic K in the sense of abstract algebraic logic, and the lattice of its subvarieties is dually isomorphic to the lattice of normal modal logics.

Stone's representation theorem can be generalized to the Jónsson–Tarski duality, which ensures that each modal algebra can be represented as the algebra of admissible sets in a modal general frame.

References

A. Chagrov and M. Zakharyashev, *Modal Logic*, Oxford Logic Guides vol. 35, Oxford University Press, 1997.
ISBN 0-19-853779-4

Petrick's method

In Boolean algebra, **Petrick's method** (also known as the *branch-and-bound* method) is a technique for determining all minimum sum-of-products solutions from a prime implicant chart. Petrick's method is very tedious for large charts, but it is easy to implement on a computer.

1. Reduce the prime implicant chart by eliminating the essential prime implicant rows and the corresponding columns.
2. Label the rows of the reduced prime implicant chart P_1, P_2, P_3, P_4 , etc.
3. Form a logical function P which is true when all the columns are covered. P consists of a product of sums where each sum term has the form $(P_{i0} + P_{i1} + \dots + P_{iN})$, where each P_{ij} represents a row covering column i .
4. Reduce P to a minimum sum of products by multiplying out and applying $X + XY = X$.
5. Each term in the result represents a solution, that is, a set of rows which covers all of the minterms in the table. To determine the minimum solutions, first find those terms which contain a minimum number of prime implicants.
6. Next, for each of the terms found in step five, count the number of literals in each prime implicant and find the total number of literals.
7. Choose the term or terms composed of the minimum total number of literals, and write out the corresponding sums of prime implicants.

Example of Petrick's method (copied from <http://www.mrc.uidaho.edu/mrc/people/jff/349/lect.10>)

Following is the function we want to reduce:

$$f(A, B, C) = \sum m(0, 1, 2, 5, 6, 7)$$

The prime implicant chart from the Quine-McCluskey algorithm is as follows:

	0	1	2	5	6	7
K (0, 1)	a'b'		X	X		
L (0, 2)	a'c'	X		X		
M (1, 5)	b'c		X		X	
N (2, 6)	bc'		X		X	
P (5, 7)	ac			X	X	
Q (6, 7)	ab				X	X

Based on the X marks in the table above, build a product of sums of the rows where each row is added, and columns are multiplied together:

$$(K+L)(K+M)(L+N)(M+P)(N+Q)(P+Q)$$

Use the distributive law to turn that expression into a sum of products. Also use the following equivalences to simplify the final expression: $X + XY = X$ and $XX = X$ and $X+X=X$

$$\begin{aligned} &= (K+L)(K+M)(L+N)(M+P)(N+Q)(P+Q) \\ &= (K+LM)(N+LQ)(P+MQ) \\ &= (KN+KLQ+LMN+LMQ)(P+MQ) \\ &= KNP + KLPQ + LMNP + LMPQ + KMNQ + KLMQ + LMNQ + LMQ \end{aligned}$$

Now use again the following equivalence to further reduce the equation: $X + XY = X$

$$= KNP + KLPQ + LMNP + LMQ + KMNQ$$

Choose products with fewest terms, in our example, there are two products with three terms:

KNP
LMQ

Choose term or terms with fewest total literals. In our example, the two products both expand to 6 literals total each:

KNP expands to $a'b' + bc' + ac$
LMQ expands to $a'c' + b'c + ab$

So either one can be used. In general, application of Petricks method is tedious for large charts, but it is easy to implement on a computer.

External links

- [1] Tutorial on Quine-McCluskey and Petrick's method (pdf).

References

[1] <http://www.simpogical.com/download>

Product term

In Boolean logic, a **product term** is a conjunction of literals, where each literal is either a variable or its negation. Examples of product terms include:

$$A \wedge B$$

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

$$\neg A$$

The terminology comes from the similarity of AND to multiplication as in the ring structure of Boolean rings.

Propositional formula

In propositional logic, a **propositional formula** is a type of syntactic formula which is well formed and has a truth value. If the values of all variables in a propositional formula are given, it determines a unique truth value. A propositional formula may also be called a **propositional expression**, a **sentence**, or a **sentential formula**.

A propositional formula is constructed from simple propositions, such as "five is greater than three" or propositional variables such as P and Q , using connectives such as NOT, AND, OR, and IMPLIES; for example:

$$(P \text{ AND NOT } Q) \text{ IMPLIES } (P \text{ OR } Q).$$

In mathematics, a propositional formula is often more briefly referred to as a "**proposition**", but, more precisely, a propositional formula is not a proposition but a formal expression that *denotes* a proposition, a formal object under discussion, just like an expression such as " $x + y$ " is not a value, but denotes a value. In some contexts, maintaining the distinction may be of importance.

Propositions

For the purposes of the propositional calculus, **propositions** (utterances, sentences, assertions) are considered to be either **simple** or **compound**.^[1] Compound propositions are considered to be linked by **sentential connectives**, some of the most common of which are "AND", "OR", "IF ... THEN ...", "NEITHER ... NOR...", "... IS EQUIVALENT TO ...". The linking semicolon ";", and connective "BUT" are considered to be expressions of "AND". A sequence of discrete sentences are considered to be linked by "AND"s, and formal analysis applies a recursive "parenthesis rule" with respect to sequences of simple propositions (see more below about well-formed formulas).

For example: The assertion: "This cow is blue. That horse is orange but this horse here is purple." is actually a compound proposition linked by "AND"s: (("This cow is blue" AND "that horse is orange") AND "this horse here is purple").

Simple propositions are declarative in nature, that is, they make assertions about the condition or nature of a *particular* object of sensation e.g. "This cow is blue", "There's a coyote!" ("That coyote IS *there*, behind the rocks.").^[2] Thus the simple "primitive" assertions must be about specific objects or specific states of mind. Each must have at least a **subject** (an immediate object of thought or observation), a verb (in the active voice and present tense preferred), and perhaps an adjective or adverb. "Dog!" probably implies "I see a dog" but should be rejected as too ambiguous.

Example: "That purple dog is running", "This cow is blue", "Switch M31 is closed", "This cap is off", "Tomorrow is Friday".

For the purposes of the propositional calculus a compound proposition can usually be reworded into a series of simple sentences, although the result will probably sound stilted.

Relationship between propositional and predicate formulas

The predicate calculus goes a step further than the propositional calculus to an "analysis of the *inner structure* of propositions"^[3] It breaks a simple sentence down into two parts (i) its **subject** (the object (singular or plural) of discourse) and (ii) a predicate—a verb or possibly verb-clause that asserts a quality or attribute of the object(s)). The predicate calculus then generalizes the "subject|predicate" form (where | symbolizes concatenation (stringing together) of symbols) into a form with the following blank-subject structure " ____|predicate", and the predicate in turn generalized to all things with that property.

Example: "This blue pig has wings" becomes two sentences in the *propositional calculus*: "This pig has wings" AND "This pig is blue", whose internal structure is not considered. In contrast, in the predicate calculus, the first sentence breaks into "this pig" as the subject, and "has wings" as the predicate. Thus it

asserts that object "this pig" is a member of the class (set, collection) of "winged things". The second sentence asserts that object "this pig" has an attribute "blue" and thus is a member of the class of "blue things". One might choose to write the two sentences connected with AND as:

$p|W \text{ AND } p|B$

The generalization of "this pig" to a (potential) member of two classes "winged things" and "blue things" means that it has a truth-relationship with both of these classes. In other words, given a **domain of discourse** "winged things", either we find p to be a member of this domain or not. Thus we have a relationship W (wingedness) between p (pig) and $\{ T, F \}$, $W(p)$ evaluates to $\{ T, F \}$. Likewise for B (blueness) and p (pig) and $\{ T, F \}$: $B(p)$ evaluates to $\{ T, F \}$. So we now can analyze the connected assertions " $B(p)$ AND $W(p)$ " for its overall truth-value, i.e.:

$(B(p) \text{ AND } W(p))$ evaluates to $\{ T, F \}$

In particular, simple sentences that employ notions of "all", "some", "a few", "one of", etc. are treated by the predicate calculus. Along with the new function symbolism " $F(x)$ " two new symbols are introduced: \forall (For all), and \exists (There exists ..., At least one of ... exists, etc.). The predicate calculus, but not the propositional calculus, can establish the formal validity the following statement:

"All blue pigs have wings but some pigs have no wings, hence some pigs are not blue".

Identity

Tarski asserts that the notion of IDENTITY (as distinguished from LOGICAL EQUIVALENCE) lies outside the propositional calculus; however, he notes that if a logic is to be of use for mathematics and the sciences it must contain a "theory" of IDENTITY.^[4] Some authors refer to "predicate logic with identity" to emphasize this extension. See more about this below.

An algebra of propositions, the propositional calculus

An **algebra** (and there are many different ones), loosely defined, is a method by which a collection of **symbols** called **variables** together with some other symbols such as parentheses $(,)$ and some sub-set of symbols such as $*$, $+$, \sim , $\&$, \vee , $=$, \equiv , \wedge , \neg are manipulated within a **system** of rules. These symbols, and **well-formed** strings of them, are said to represent **objects**, but in a specific algebraic system these objects do not have **meanings**. Thus work inside the algebra becomes an exercise in obeying certain **laws (rules)** of the algebra's syntax (symbol-formation) rather than in semantics (meaning) of the symbols. The meanings are to be found outside the algebra.

For a well-formed sequence of symbols in the algebra—a **formula**—to have some usefulness outside the algebra the symbols are assigned meanings and eventually the variables are assigned **values**; then by a series of rules the formula is **evaluated**.

When the values are restricted to just two and applied to the notion of **simple sentences** (e.g. spoken utterances or written assertions) linked by **propositional connectives** this whole algebraic system of symbols and rules and evaluation-methods is usually called the propositional calculus or the sentential calculus.

While some of the familiar rules of arithmetic algebra continue to hold in the algebra of propositions (e.g. the commutative and associative laws for AND and OR), some do not (e.g. the distributive laws for AND, OR and NOT).

Usefulness of propositional formulas

Analysis: In deductive reasoning, philosophers, rhetoricians and mathematicians reduce arguments to formulas and then study them (usually with truth tables) for correctness (soundness). For example: Is the following argument sound?

"Given that consciousness is sufficient for an artificial intelligence and only conscious entities can pass the Turing test, before we can conclude that a robot is an artificial intelligence the robot must pass the Turing test."

Engineers analyze the logic circuits they have designed using synthesis techniques and then apply various reduction and minimization techniques to simplify their designs.

Synthesis: Engineers in particular synthesize propositional formulas (that eventually end up as **circuits** of symbols) from truth tables. For example, one might write down a truth table for how binary addition should behave given the addition of variables "b" and "a" and "carry_in" "ci", and the results "carry_out" "co" and "sum" Σ :

Example: in row 5, $(b+a) + ci = (1+0) + 1 =$ the number "2". written as a binary number this is 10_2 , where "co"=1 and $\Sigma=0$ as shown in the right-most columns.

row	b	a	ci	$(b+a)+ci$	co	Σ
0	0	0	0	0	0	0
1	0	0	1	1	0	1
2	0	1	0	1	0	1
3	0	1	1	2	1	0
4	1	0	0	1	0	1
5	1	0	1	2	1	0
6	1	1	0	2	1	0
7	1	1	1	3	1	1

Propositional variables

The simplest type of propositional formula is a **propositional variable**. Propositions that are simple (atomic), symbolic expressions are often denoted by variables named *a*, *b*, or *A*, *B*, etc. A propositional variable is intended to represent an atomic proposition (assertion), such as "It is Saturday" = *a* (here the symbol = means "... is assigned the variable named ...") or "I only go to the movies on Monday" = *b*.

Truth-value assignments, formula evaluations

Evaluation of a propositional formula begins with assignment of a **truth value** to each variable. Because each variable represents a simple sentence, the truth values are being applied to the "truth" or "falsity" of these simple sentences.

Truth values in rhetoric, philosophy and mathematics: The truth values are only two: { TRUTH "T", FALSITY "F" }. An empiricist puts all propositions into two broad classes: *analytic*—true no matter what (e.g. tautology), and *synthetic*—derived from experience and thereby susceptible to confirmation by third parties (the verification theory of meaning).^[5] Empiricists hold that, in general, to arrive at the truth-value of a synthetic proposition, meanings (pattern-matching templates) must first be applied to the words, and then these meaning-templates must be matched against whatever it is that is being asserted. For example, my utterance "That cow is *blue!*" Is this statement a TRUTH? Truly I said it. And maybe I *am* seeing a blue cow—unless I am lying my statement is a TRUTH relative to the object of my (perhaps flawed) perception. But is the blue cow "really there"? What do you see when you look out the same window? In order to proceed with a verification, you will need a prior notion (a template) of both "cow"

and "blue", and an ability to match the templates against the object of sensation (if indeed there is one).

Truth values in engineering: Engineers try to avoid notions of truth and falsity that bedevil philosophers, but in the final analysis engineers must trust their measuring instruments. In their quest for robustness, engineers prefer to pull known objects from a small library—objects that have well-defined, predictable behaviors even in large combinations, (hence their name for the propositional calculus: "combinatorial logic"). The fewest behaviors of a single object are two (e.g. { OFF, ON }, { open, shut }, { UP, DOWN } etc.), and these are put in correspondence with { 0, 1 }. Such elements are called digital; those with a continuous range of behaviors are called analog. Whenever decisions must be made in an analog system, quite often an engineer will convert an analog behavior (the door is 45.32146% UP) to digital (e.g. DOWN=0) by use of a comparator.^[6]

Thus an assignment of **meaning** of the variables and the two value-symbols { 0, 1 } comes from "outside" the formula that represents the behavior of the (usually) compound object. An example is a garage door with two "limit switches", one for UP labelled SW_U and one for DOWN labelled SW_D, and whatever else is in the door's circuitry. Inspection of the circuit (either the diagram or the actual objects themselves—door, switches, wires, circuit board, etc.) might reveal that, on the circuit board "node 22" goes to +0 volts when the contacts of switch "SW_D" are mechanically in contact ("closed") and the door is in the "down" position (95% down), and "node 29" goes to +0 volts when the door is 95% UP and the contacts of switch SW_U are in mechanical contact ("closed").^[7] The engineer must define the meanings of these voltages and all possible combinations (all 4 of them), including the "bad" ones (e.g. both nodes 22 and 29 at 0 volts, meaning that the door is open and closed at the same time). The circuit mindlessly responds to whatever voltages it experiences without any awareness of TRUTH or FALSEHOOD, RIGHT or WRONG, SAFE or DANGEROUS.

Propositional connectives

Arbitrary propositional formulas are built from propositional variables and other propositional formulas using propositional connectives. Examples of connectives include:

- The unary negation connective. If α is a formula, then $\neg\alpha$ is a formula.
- The classical binary connectives \wedge , \vee , \rightarrow , \leftrightarrow . Thus, for example, if α and β are formulas, so is $(\alpha \rightarrow \beta)$.
- Other binary connectives, such as NAND, NOR, and XOR
- The ternary connective IF ... THEN ... ELSE ...
- Constant 0-ary connectives \top and \perp (alternately, constants { T, F }, { 1, 0 } etc.)
- The "theory-extension" connective EQUALS (alternately, IDENTITY, or the sign " = " as distinguished from the "logical connective" \leftrightarrow)

Connectives of rhetoric, philosophy and mathematics

The following are the connectives common to rhetoric, philosophy and mathematics together with their truth tables. The symbols used will vary from author to author and between fields of endeavor. In general the abbreviations "T" and "F" stand for the evaluations TRUTH and FALSEITY applied to the variables in the propositional formula (e.g. the assertion: "That cow is blue" will have the truth-value "T" for Truth or "F" for Falsity, as the case may be.).

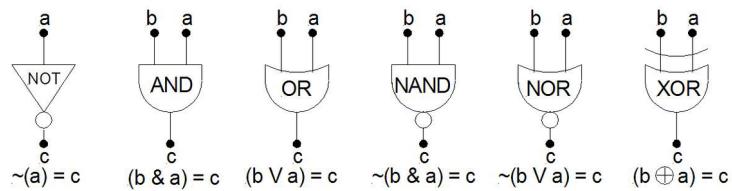
The connectives go by a number of different word-usages, e.g. "a IMPLIES b" is also said "IF a THEN b". Some of these are shown in the table.

					b only if a						
					b IS SUFFICIENT FOR a						
					a IS NECESSARY FOR b	b IF AND ONLY IF a; b IFF a					
				inclusive OR	IF b THEN a	b IS NECESSARY AND SUFFICIENT FOR a					
		negation	negation	conjunction	disjunction	implication	biconditional				
variable	variable	NOT b	NOT a	b AND a	b OR a	b IMPLIES a	b IS logically equivalent TO a ***	f IS A tautology	NEITHER a NOR b	b stroke a	exclusive OR
b	a	$\neg(b)$	$\neg(a)$	$(b \wedge a)$	$(b \vee a)$	$(b \rightarrow a)$	$(b \leftrightarrow a)$	$(f =$ formula)	$(a \text{ NOR } b)$	$(b a)$	various
F	F	T	T	F	F	T	T	T	T	T	F
F	T	T	F	F	T	T	F	T	F	T	T
T	F	F	T	F	T	F	F	T	F	T	T
T	T	F	F	T	T	T	T	T	F	F	F

Engineering connectives

In general, the engineering connectives are just the same as the mathematics connectives excepting they tend to evaluate with "1" = "T" and "0" = "F". This is done for the purposes of analysis/minimization and synthesis of formulas by use of the notion of *minterms* and Karnaugh maps (see below). Engineers also use the words **logical product** from Boole's notion ($a^*a = a$) and **logical sum** from Jevons' notion ($a+a = a$).^[8]

Symbols for common engineering connectives



Engineering symbols have varied over the years, but these are commonplace. Sometimes they appear simply as boxes with symbols in them. "a" and "b" are called "the inputs" and "c" is called "the output". An output will typically "connect to" an input (unless it is the final connective); this accomplishes the mathematical notion of **substitution**.

					logical product	logical sum			half-adder (no carry)
									exclusive OR
row number	variable	variable	NOT	NOT	AND	OR	NAND	NOR	XOR
$b*2^1+a*2^0$	b	a	$\sim(b)$	$\sim(a)$	$(b \& a)$	$(b \vee a)$	$\sim(b \& a)$	$\sim(b \vee a)$	\oplus
0	0	0	1	1	0	0	1	1	0
1	0	1	1	0	0	1	1	0	1
2	1	0	0	1	0	1	1	0	1
3	1	1	0	0	1	1	0	0	0

CASE connective: IF ... THEN ... ELSE ...

The IF ... THEN ... ELSE ... connective appears as the simplest form of CASE operator of recursion theory and computation theory and is the connective responsible for conditional goto's (jumps, branches). From this one connective all other connectives can be constructed (see more below). Although " IF c THEN b ELSE a " sounds like an implication it is, in its most reduced form, a **switch** that makes a decision and offers as outcome only one of two alternatives "a" or "b" (hence the name switch statement in the C programming language).^[9]

The following three propositions are equivalent (as indicated by the logical equivalence sign \equiv):

- (1) (IF 'counter is zero' THEN 'go to instruction b ' ELSE 'go to instruction a ') \equiv
- (2) (($c \rightarrow b$) & ($\sim c \rightarrow a$)) \equiv ((IF 'counter is zero' THEN 'go to instruction b ') AND (IF 'It is NOT the case that counter is zero' THEN 'go to instruction a ')) \equiv
- (3) (($c \& b$) V ($\sim c \& a$)) = " ('Counter is zero' AND 'go to instruction b ') OR ('It is NOT the case that 'counter is zero' AND 'go to instruction a ') "

Thus IF ... THEN ... ELSE—unlike implication—does not evaluate to an ambiguous "TRUTH" when the first proposition is false i.e. $c = F$ in $(c \rightarrow b)$. For example, most people would reject the following compound proposition as a nonsensical *non sequitur* because the second sentence is *not connected in meaning* to the first.^[10]

Example: The proposition " IF 'Winston Churchill was Chinese' THEN 'The sun rises in the east' " evaluates as a TRUTH given that 'Winston Church was Chinese' is a FALSEHOOD and 'The sun rises in the east' evaluates as a TRUTH.

In recognition of this problem, the sign \rightarrow of formal implication in the propositional calculus is called material implication to distinguish it from the everyday, intuitive implication.^[11]

The use of the IF ... THEN ... ELSE construction avoids controversy because it offers a completely deterministic choice between two stated alternatives; it offers two "objects" (the two alternatives b and a), and it *selects* between them exhaustively and unabiguously.^[12] In the truth table below, d1 is the formula: ((IF c THEN b) AND (IF NOT-c THEN a)). Its fully reduced form d2 is the formula: ((c AND b) OR (NOT-c AND a)). The two formulas are equivalent as shown by the columns " $=d1$ " and " $=d2$ ". Electrical engineers call the fully reduced formula the AND-OR-SELECT operator. The CASE (or SWITCH) operator is an extension of the same idea to n possible, but mutually exclusive outcomes. Electrical engineers call the CASE operator a multiplexer.

							d1								d2																						
row	c	b	a	((c	\rightarrow	b)	&	(\sim	(c)	\rightarrow	a))	=d1	((c	&	b)	V	(\sim	(c)	&	a))	=d2
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0							
1	0	0	1	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1							
2	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0							
3	0	1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	0	0	1	1	1	0	1	1	1	1	1	1	1	1							
4	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0							
5	1	0	1	0	1	0	0	0	1	1	1	1	0	0	0	1	0	0	0	0	0	1	0	1	0	1	0	0	0	0							
6	1	1	0	0	1	1	1	0	1	1	0	0	1	1	0	1	1	1	1	0	1	0	1	0	0	0	1	0	0	1							
7	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	0	1	0	1	0	1	0	1	1	0	1	1							

IDENTITY and evaluation

The first table of this section stars *** the entry logical equivalence to note the fact that "Logical equivalence" is not the same thing as "identity". For example, most would agree that the assertion "That cow is blue" is identical to the assertion "That cow is blue". On the other hand *logical* equivalence sometimes appears in speech as in this example: "The sun is shining' means 'I'm biking' ". Translated into a propositional formula the words become: "IF 'the sun is shining' THEN 'I'm biking', AND IF 'I'm biking' THEN 'the sun is shining'"^[13]..

"IF 's' THEN 'b' AND IF 'b' THEN 's' " is written as $((s \rightarrow b) \& (b \rightarrow s))$ or in an abbreviated form as $(s \leftrightarrow b)$.

As the rightmost symbol string is a **definition** for a new symbol in terms of the symbols on the left, the use of the IDENTITY sign = is appropriate:

$$((s \rightarrow b) \& (b \rightarrow s)) = (s \leftrightarrow b)$$

Different authors use different signs for logical equivalence: \leftrightarrow (e.g. Suppes, Goodstein, Hamilton), \equiv (e.g. Robbin), \Leftrightarrow (e.g. Bender and Williamson). Typically identity is written as the equals sign =. One exception to this rule is found in *Principia Mathematica*. For more about the philosophy of the notion of IDENTITY see Leibniz's law.

As noted above, Tarski considers IDENTITY to lie outside the propositional calculus, but he asserts that without the notion, "logic" is insufficient for mathematics and the deductive sciences. In fact the sign comes into the propositional calculus when a formula is to be evaluated.^[14]

In some systems there are no truth tables, but rather just formal axioms (e.g. strings of symbols from a set { \sim , \rightarrow , (,), variables p_1, p_2, p_3, \dots } and formula-formation rules (rules about how to make more symbol strings from previous strings by use of e.g. substitution and modus ponens). the result of such a calculus will be another formula (i.e. a well-formed symbol string). Eventually, however, if one wants to use the calculus to study notions of validity and truth, one must add axioms that define the behavior of the symbols called "the truth values" { T, F } (or { 1, 0 }, etc.) relative to the other symbols.

For example, Hamilton uses two symbols = and \neq when he defines the notion of a **valuation** v of any wffs A and B in his "formal statement calculus" L. A valuation v is a *function* from the wffs of his system L to the range (output) { T, F }, given that each variable p_1, p_2, p_3 in a wff is assigned an arbitrary truth value { T, F }.

- (i) $v(A) \neq v(\sim A)$
- (ii) $v(A \rightarrow B) = F$ if and only if $v(A) = T$ and $v(B) = F$

The two definitions (i) and (ii) define the equivalent of the truth tables for the \sim (NOT) and \rightarrow (IMPLICATION) connectives of his system. The first one derives $F \neq T$ and $T \neq F$, in other words " $v(A)$ does not mean $v(\sim A)$ ". Definition (ii) specifies the third row in the truth table, and the other three rows then come from an application of definition (i). In particular (ii) **assigns** the value F (or a meaning of "F") to the entire expression. The definitions also

serve as formation rules that allow substitution of a value previously derived into a formula:

		$v(A \rightarrow B)$		
($v(A)$	\rightarrow	$v(B)$)
	F	T	F	
	F	T	T	
	T	F	F	
	T	T	T	

Some formal systems specify these valuation axioms at the outset in the form of certain formulas such as the law of contradiction or laws of identity and nullity. The choice of which ones to use, together with laws such as commutation and distribution, is up to the system's designer as long as the set of axioms is **complete** (i.e. sufficient to form and to evaluate any well-formed formula created in the system).

More complex formulas

As shown above, the CASE (IF c THEN b ELSE a) connective is constructed either from the 2-argument connectives IF...THEN... and AND or from OR and AND and the 1-argument NOT. Connectives such as the n-argument AND ($a \& b \& c \& \dots \& n$), OR ($a \vee b \vee c \vee \dots \vee n$) are constructed from strings of two-argument AND and OR and written in abbreviated form without the parentheses. These, and other connectives as well, can then be used as building blocks for yet further connectives. Rhetoricians, philosophers, and mathematicians use truth tables and the various theorems to analyze and simplify their formulas.

Electrical engineering uses drawn symbols and connect them with lines that stand for the mathematical act of **substitution** and **replacement**. They then verify their drawings with truth tables and simplify the expressions as shown below by use of Karnaugh maps or the theorems. In this way engineers have created a host of "combinatorial logic" (i.e. connectives without feedback) such as "decoders", "encoders", "mutifunction gates", "majority logic", "binary adders", "arithmetic logic units", etc.

Definitions

A definition creates a new symbol and its behavior, often for the purposes of abbreviation. Once the definition is presented, either form of the equivalent symbol or formula can be used. The following symbolism $=_{Df}$ is following the convention of Reichenbach.^[15] Some examples of convenient definitions drawn from the symbol set { \sim , $\&$, $(,)$ } and variables. Each definition is producing a logically equivalent formula that can be used for substitution or replacement.

- definition of a new variable: $(c \& d) =_{Df} s$
- OR: $\sim(\sim a \& \sim b) =_{Df} (a \vee b)$
- IMPLICATION: $(\sim a \vee b) =_{Df} (a \rightarrow b)$
- XOR: $(\sim a \& b) \vee (a \& \sim b) =_{Df} (a \oplus b)$
- LOGICAL EQUIVALENCE: $((a \rightarrow b) \& (b \rightarrow a)) =_{Df} (a \equiv b)$

Axiom and definition *schemas*

The definitions above for OR, IMPLICATION, XOR, and logical equivalence are actually schemas (or "schemata"), that is, they are *models* (demonstrations, examples) for a general formula *format* but shown (for illustrative purposes) with specific letters a, b, c for the variables, whereas any variable letters can go in their places as long as the letter substitutions follow the rule of substitution below.

Example: In the definition $(\sim a \vee b) =_{\text{Df}} (a \rightarrow b)$, other variable-symbols such as "SW2" and "CON1" might be used, i.e. formally:

$$\begin{aligned} a &=_{\text{Df}} \text{SW2}, b =_{\text{Df}} \text{CON1}, \text{ so we would have as an } \textit{instance} \text{ of the definition schema } (\sim \text{SW2} \vee \text{CON1}) \\ &=_{\text{Df}} (\text{SW2} \rightarrow \text{CON1}) \end{aligned}$$

Substitution versus replacement

Substitution: The variable or sub-formula to be substituted with another variable, constant, or sub-formula must be replaced in all instances throughout the overall formula.

Example: $(c \& d) \vee (p \& \sim(c \& \sim d))$, but $(q_1 \& \sim q_2) \equiv d$. Now wherever variable "d" occurs, substitute $(q_1 \& \sim q_2)$:

$$(c \& (q_1 \& \sim q_2)) \vee (p \& \sim(c \& \sim(q_1 \& \sim q_2)))$$

Replacement: (i) the formula to be replaced must be within a tautology, i.e. *logically equivalent* (connected by \equiv or \leftrightarrow) to the formula that replaces it, and (ii) unlike substitution its permissible for the replacement to occur only in one place (i.e. for one formula).

Example: Use this set of formula schemas/equivalences: 1: $((a \vee 0) \equiv a)$. 2: $((a \& \sim a) \equiv 0)$. 3: $((\sim a \vee b) =_{\text{Df}} (a \rightarrow b))$. 6. $(\sim(\sim a) \equiv a)$

- start with "a": a
- Use 1 to replace "a" with $(a \vee 0)$: $(a \vee 0)$
- Use the notion of "schema" to substitute b for a in 2: $((a \& \sim a) \equiv 0)$
- Use 2 to replace 0 with $(b \& \sim b)$: $((a \vee (b \& \sim b)) \equiv 0)$
- (see below for how to distribute " $a \vee$ " over $(b \& \sim b)$, etc

Inductive definition

The classical presentation of propositional logic (see Enderton 2002) uses the connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. The set of formulas over a given set of propositional variables is inductively defined to be the smallest set of expressions such that:

- Each propositional variable in the set is a formula,
- $(\neg \alpha)$ is a formula whenever α is, and
- $(\alpha \Box \beta)$ is a formula whenever α and β are formulas and \Box is one of the binary connectives $\wedge, \vee, \rightarrow, \leftrightarrow$.

This inductive definition can be easily extended to cover additional connectives.

The inductive definition can also be rephrased in terms of a closure operation (Enderton 2002). Let V denote a set of propositional variables and let X_V denote the set of all strings from an alphabet including symbols in V , left and right parentheses, and all the logical connectives under consideration. Each logical connective corresponds to a formula building operation, a function from XX_V to XX_V :

- Given a string z , the operation $\mathcal{E}_{\neg}(z)$ returns $(\neg z)$.
- Given strings y and z , the operation $\mathcal{E}_{\wedge}(y, z)$ returns $(y \wedge z)$. There are similar operations $\mathcal{E}_{\vee}, \mathcal{E}_{\rightarrow},$ and $\mathcal{E}_{\leftrightarrow}$ corresponding to the other binary connectives.

The set of formulas over V is defined to be the smallest subset of XX_V containing V and closed under all the formula building operations.

Parsing formulas

The following "laws" of the propositional calculus are used to "reduce" complex formulas. The "laws" can be easily verified with truth tables. For each law, the principal (outermost) connective is associated with logical equivalence \equiv or identity $=$. A complete analysis of all 2^n combinations of truth-values for its n distinct variables will result in a column of 1's (T's) underneath this connective. This finding makes each law, by definition, a tautology. And, for a given law, because its formula on the left and right are equivalent (or identical) they can be substituted for one another.

Example: The following truth table is De Morgan's law for the behavior of NOT over OR: $\sim(a \vee b) \equiv (\sim a \wedge \sim b)$. To the left of the principal connective \equiv (yellow column labelled "taut") the formula $\sim(b \vee a)$ evaluates to (1, 0, 0, 0) under the label "P". On the right of "taut" the formula $(\sim b) \wedge (\sim a)$ also evaluates to (1, 0, 0, 0) under the label "Q". As the two columns have equivalent evaluations, the logical equivalence \equiv under "taut" evaluates to (1, 1, 1, 1), i.e. P \equiv Q. Thus either formula can be substituted for the other if it appears in a larger formula.

			P				taut				Q									
b	a	(\sim	(b	\vee	a)	\equiv	(\sim	(b)	\wedge	\sim	(a))
0	0		1		0	0	0		1		1		0		1	1		0		
0	1		0		0	1	1		1		1		0		0	0		1		
1	0		0		1	1	0		1		0		1		0	1		0		
1	1		0		1	1	1		1		0		1		0	0		1		

Enterprising readers might challenge themselves to invent an "axiomatic system" that uses the symbols { V, &, ~, (,), variables a, b, c }, the formation rules specified above, as few as possible of the laws listed below, and then derive as theorems the others as well as the truth-table valuations for V, &, and ~. One set attributed to Huntington (1904) (Suppes:204) uses 8 of the laws defined below.

Note that if used in an axiomatic system, the symbols 1 and 0 (or T and F) are considered to be wffs and thus obey all the same rules as the variables. Thus the laws listed below are actually axiom schemas, that is, they stand in place of an infinite number of instances. Thus $(x \vee y) \equiv (y \vee x)$ might be used in one instance, $(p \vee 0) \equiv (0 \vee p)$ and in another instance $(1 \vee q) \equiv (q \vee 1)$, etc.

Connective seniority (symbol rank)

In general, to avoid confusion during analysis and evaluation of propositional formulas make liberal use parentheses. However, quite often authors leave them out. To parse a complicated formula one first needs to know the **seniority**, or **rank** that each of the connectives (excepting *) has over the other connectives. To "well-form" a formula start with the connective with the highest rank and add parentheses around its components, then move down in rank (paying close attention to the connective's **scope** over which it is working). From most- to least-senior, with the predicate signs $\forall x$ and $\exists x$, the IDENTITY = and arithmetic signs added for completeness:^[16]

- (LOGICAL EQUIVALENCE), → (IMPLICATION), & (AND), V (OR), ~ (NOT), $\forall x$ (FOR ALL x),
- $\exists x$ (THERE EXISTS AN x), = (IDENTITY), + (arithmetic sum), *(arithmetic multiply), ' (s, arithmetic successor).

Thus the formula can be parsed—but note that, because NOT does not obey the distributive law, the parentheses around the inner formula ($\sim c \wedge \sim d$) is mandatory:

Example: " d & c V w " rewritten is ((d & c) V w)

Example: " a & a → b ≡ a & ~a V b " rewritten (rigorously) is

- \equiv has seniority: $((a \& a \rightarrow b) \equiv (a \& \sim a V b))$
- \rightarrow has seniority: $((a \& (a \rightarrow b)) \equiv (a \& \sim a V b))$
- $\&$ has seniority both sides: $((((a) \& (a \rightarrow b))) \equiv ((a) \& (\sim a V b)))$
- \sim has seniority: $((((a) \& (a \rightarrow b))) \equiv ((a) \& (\sim(a) V b)))$
- check 9 (-parenthesis and 9) -parenthesis: $((((a) \& (a \rightarrow b))) \equiv ((a) \& (\sim(a) V b)))$

Example:

$$d \& c V p \& \sim(c \& \sim d) \equiv c \& d V p \& c V p \& \sim d \text{ rewritten is } ((d \& c) V (p \& \sim(c \& \sim(d)))) \equiv ((c \& d) V (p \& c) V (p \& \sim(d)))$$

Commutative and associative laws

Both AND and OR obey the commutative law and associative law:

- Commutative law for OR: $(a V b) \equiv (b V a)$
- Commutative law for AND: $(a \& b) \equiv (b \& a)$
- Associative law for OR: $((a V b) V c) \equiv (a V (b V c))$
- Associative law for AND: $((a \& b) \& c) \equiv (a \& (b \& c))$

Omitting parentheses in strings of AND and OR: The connectives are considered to be unary (one-variable, e.g. NOT) and binary (i.e. two-variable AND, OR, IMPLIES). For example:

$$(c \& d) V (p \& c) V (p \& \sim d) \text{ above should be written } ((c \& d) V (p \& c)) V (p \& \sim(d)) \text{ or possibly } ((c \& d) V (p \& c) V (p \& \sim(d)))$$

However, a truth-table demonstration shows that the form without the extra parentheses is perfectly adequate.

Omitting parentheses with regards to a single-variable NOT: While $\sim(a)$ where a is a single variable is perfectly clear, $\sim a$ is adequate and is the usual way this literal would appear. When the NOT is over a formula with more than one symbol, then the parentheses are mandatory, e.g. $\sim(a V b)$

Distributive laws

OR distributes over AND and AND distributes over OR. NOT does not distribute over AND nor OR. See below about De Morgan's law:

- Distributive law for OR: $(c V (a \& b)) \equiv ((c V a) \& (c V b))$
- Distributive law for AND: $(c \& (a V b)) \equiv ((c \& a) V (c \& b))$

De Morgan's laws

NOT, when distributed over OR or AND, does something peculiar (again, these can be verified with a truth-table):

- De Morgan's law for OR: $\sim(a V b) \equiv (\sim a \& \sim b)$
- De Morgan's law for AND: $\sim(a \& b) \equiv (\sim a V \sim b)$

Laws of absorption

Absorption, in particular the first one, cause the "laws" of logic to differ from the "laws" of arithmetic:

- Absorption (idempotency) for OR: $(a \vee a) \equiv a$
- Absorption (idempotency) for AND: $(a \wedge a) \equiv a$

Laws of evaluation: Identity, nullity, and complement

The sign " $=$ " (as distinguished from logical equivalence \equiv , alternately \leftrightarrow or \Leftrightarrow) symbolizes the assignment of value or meaning. Thus the string $(a \wedge \sim(a))$ symbolizes "1", i.e. it **means** the same thing as symbol "1" ". In some "systems" this will be an axiom (definition) perhaps shown as $((a \wedge \sim(a)) =_{\text{Df}} 1)$; in other systems, it may be derived in the truth table below:

			c					taut	\equiv	c	
a	((a	&	\sim	(a))	=	0
0			0	0	1		0		1	0	
1			1	0	0		1		1	0	

- Commutation of equality: $(a = b) \equiv (b = a)$
- Identity for OR: $(a \vee 0) = a$ or $(a \vee F) = a$
- Identity for AND: $(a \wedge 1) = a$ or $(a \wedge T) = a$
- Nullity for OR: $(a \vee 1) = 1$ or $(a \vee T) = T$
- Nullity for AND: $(a \wedge 0) = 0$ or $(a \wedge F) = F$
- Complement for OR: $(a \vee \sim a) = 1$ or $(a \vee \sim a) = T$, law of excluded middle
- Complement for AND: $(a \wedge \sim a) = 0$ or $(a \wedge \sim a) = F$, law of contradiction

Double negative (Involution)

- $\sim(\sim a) = a$

Well-formed formulas (wffs)

A key property of formulas is that they can be uniquely parsed to determine the structure of the formula in terms of its propositional variables and logical connectives. When formulas are written in infix notation, as above, unique readability is ensured through an appropriate use of parentheses in the definition of formulas. Alternatively, formulas can be written in Polish notation or reverse Polish notation, eliminating the need for parentheses altogether.

The inductive definition of infix formulas in the previous section can be converted to a formal grammar in Backus-Naur form:

```

<formula> ::= <propositional variable>
| (  $\neg$  <formula> )
| ( <formula>  $\wedge$  <formula> )
| ( <formula>  $\vee$  <formula> )
| ( <formula>  $\rightarrow$  <formula> )
| ( <formula>  $\leftrightarrow$  <formula> )

```

It can be shown that any expression matched by the grammar has a balanced number of left and right parentheses, and any nonempty initial segment of a formula has more left than right parentheses.^[17] This fact can be used to give an algorithm for parsing formulas. For example, suppose that an expression x begins with $(\neg$. Starting after the

second symbol, match the shortest subexpression y of x that has balanced parentheses. If x is a formula, there is exactly one symbol left after this expression, this symbol is a closing parenthesis, and y itself is a formula. This idea can be used to generate a recursive descent parser for formulas.

Example of parenthesis counting:

This method locates as "1" the **principal connective** -- the connective under which the overall evaluation of the formula occurs for the outer-most parentheses (which are often omitted).^[18] It also locates the inner-most connective where one would begin evaluation of the formula without the use of a truth table, e.g. at "level 6".

	start	(((c	&	d)	V	(p	&	~	((c	&	~	(d)))))	=	(((c	&	d)	V	(p	&	d))	V	(p	&	~	(c)))))
count	0	1	2	3	3	3	2	2	3	3	3	3	4	5	5	5	5	6	5	4	3	3	1	1	2	3	4	4	4	4	3	3	4	4	4	4	3	2	2	3	3	3	3	3	2	1	0					

Wffs versus valid formulas in inferences

The notion of **valid argument** is usually applied to inferences in arguments, but arguments reduce to propositional formulas and can be evaluated the same as any other propositional formula. Here a **valid** inference means: "The formula that represents the inference evaluates to "truth" beneath its principal connective, no matter what truth-values are assigned to its variables", i.e. the formula is a tautology.^[19] Quite possibly a formula will be *well-formed* but not **valid**. Another way of saying this is: "Being well-formed is *necessary* for a formula to be valid but it is not *sufficient*." The only way to find out if it is *both* well-formed *and* valid is to submit it to verification with a truth table or by use of the "laws":

Example 1: What does one make of the following difficult-to-follow assertion? Is it valid? "If it's sunny, but if the frog is croaking then it's not sunny, then it's the same as saying that the frog isn't croaking." Convert this to a propositional formula as follows:

" IF (a AND (IF b THEN NOT-a) THEN NOT-a" where " a " represents "its sunny" and " b " represents "the frog is croaking":

$$((a) \& ((b) \rightarrow \sim(a)) \equiv \sim(b))$$

This is well-formed, but is it *valid*? In other words, when evaluated will this yield a tautology (all T) beneath the logical-equivalence symbol \equiv ? The answer is NO, it is not valid. However, if reconstructed as an *implication* then the argument *is* valid.

"Saying it's sunny, but if the frog is croaking then it's not sunny, *implies* that the frog isn't croaking."

Other circumstances may be preventing the frog from croaking: perhaps a crane ate it.

Example 2 (from Reichenbach via Bertrand Russell):

"If pigs have wings, some winged animals are good to eat. Some winged animals are good to eat, so pigs have wings."

$((a) \rightarrow (b)) \& (b) \rightarrow (a)$ is well formed, but an invalid argument as shown by the red evaluation under the principal implication:

W	G							arg						
a	b	(((a	->	b)	&	b)	->	a)
0	0				0	1	0		0	0		1	0	
0	1				0	1	1		1	1		0	0	
1	0				1	0	0		0	0		1	1	
1	1				1	1	1		1	1		1	1	

Reduced sets of connectives

A set of logical connectives is called **complete** if every propositional formula is tautologically equivalent to a formula with just the connectives in that set. There are many complete sets of connectives, including $\{\wedge, \neg\}$, $\{\vee, \neg\}$, and $\{\rightarrow, \neg\}$. There are two binary connectives that are complete on their own, corresponding to NAND and NOR, respectively.^[20] Some pairs are not complete, for example $\{\wedge, \vee\}$.

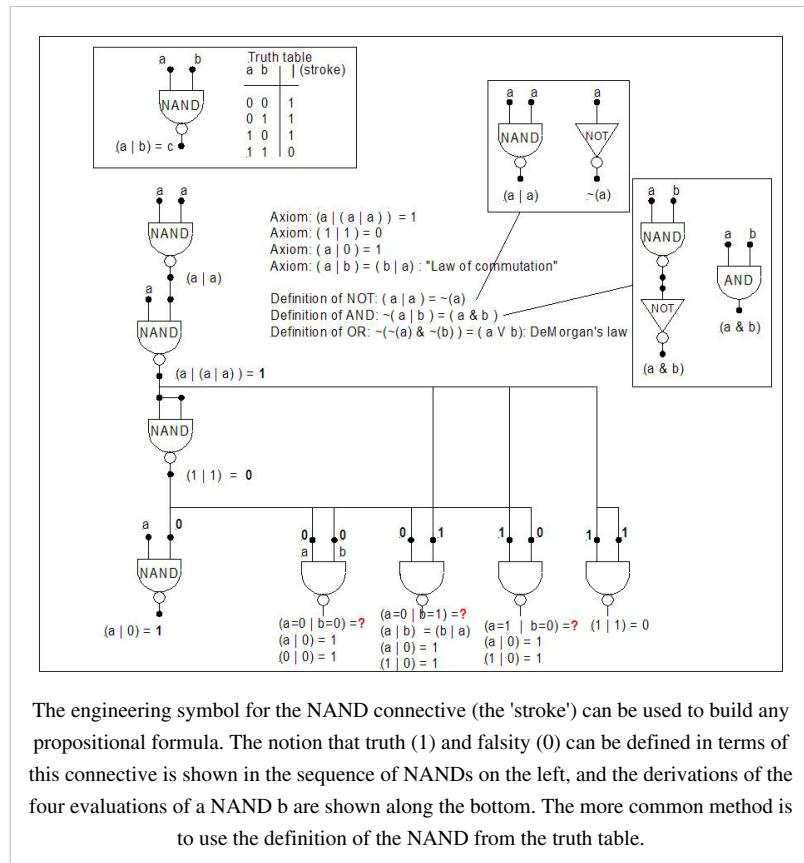
The stroke (NAND)

The binary connective corresponding to NAND is called the Sheffer stroke, and written with a vertical bar | or vertical arrow ↑. The completeness of this connective was noted in *Principia Mathematica* (1927:xvii). Since it is complete on its own, all other connectives can be expressed using only the stroke. For example, where the symbol "≡" represents *logical equivalence*:

$$\begin{aligned} \neg p &\equiv p|p \\ p \rightarrow q &\equiv p|q \\ p \vee q &\equiv \neg p|\neg q \\ p \wedge q &\equiv \neg(p|q) \end{aligned}$$

In particular, the zero-ary connectives \top (representing truth) and \perp (representing falsity) can be expressed using the stroke:

$$\begin{aligned} \top &\equiv (a|(a|a)) \\ \perp &\equiv (\top|\top) \end{aligned}$$



The engineering symbol for the NAND connective (the 'stroke') can be used to build any propositional formula. The notion that truth (1) and falsity (0) can be defined in terms of this connective is shown in the sequence of NANDs on the left, and the derivations of the four evaluations of a NAND b are shown along the bottom. The more common method is to use the definition of the NAND from the truth table.

IF ... THEN ... ELSE

This connective together with { 0, 1 }, (or { F, T } or { \perp , \top }) forms a complete set. In the following the IF...THEN...ELSE relation $(c, b, a) = d$ represents $(c \rightarrow b) \vee (\sim c \rightarrow a) \equiv ((c \& b) \vee (\sim c \& a)) = d$

(c, b, a) :

$$(c, 0, 1) \equiv \sim c$$

$$(c, b, 1) \equiv (c \rightarrow b)$$

$$(c, c, a) \equiv (c \vee a)$$

$$(c, b, c) \equiv (c \& b)$$

Example: The following shows how a theorem-based proof of " $(c, b, 1) \equiv (c \rightarrow b)$ " would proceed, below the proof is its truth-table verification. (Note: $(c \rightarrow b)$ is *defined* to be $(\sim c \vee b)$):

- Begin with the reduced form: $((c \& b) \vee (\sim c \& a))$
- Substitute "1" for a: $((c \& b) \vee (\sim c \& 1))$
- Identity $(\sim c \& 1) = \sim c$: $((c \& b) \vee (\sim c))$
- Law of commutation for \vee : $((\sim c) \vee (c \& b))$
- Distribute " $\sim c \vee$ " over $(c \& b)$: $((\sim c) \vee c) \& ((\sim c) \vee b)$
- Law of excluded middle $((\sim c) \vee c) = 1$: $((1) \& ((\sim c) \vee b))$
- Distribute " $(1) \&$ " over $((\sim c) \vee b)$: $((1) \& ((\sim c) \vee b)) \equiv ((1 \& 1) \vee (1 \& b))$
- Commutivity and Identity $((1 \& \sim c) = (\sim c \& 1) = \sim c)$, and $((1 \& b) \equiv (b \& 1) \equiv b)$: $(\sim c \vee b) \equiv b$
- $(\sim c \vee b)$ is defined as $c \rightarrow b$ Q. E. D.

In the following truth table the column labelled "taut" for tautology evaluates **logical equivalence** (symbolized here by \equiv) between the two columns labelled d. Because all four rows under "taut" are 1's, the equivalence indeed represents a tautology.

									d										taut					d					
rows	c	b	a	((c	&	b)	V	(~	(c)	&	a))	=	(~	(c)	V	b))
0,1	0	0	1			0	0	0		1		1		0		1	1			1		1		0		1	0		
2,3	0	1	1			0	0	1		1		1		0		1	1			1		1		0		1	1		
4,5	1	0	1			1	0	0		0		0		1		0	1			1		0		1		0	0		
6,7	1	1	1			1	1	1		1		0		1		0	1			1		0		1		1	1		

Normal forms

An arbitrary propositional formula may have a very complicated structure. It is often convenient to work with formulas that have simpler forms, known as **normal forms**. Some common normal forms include conjunctive normal form and disjunctive normal form. Any propositional formula can be reduced to its conjunctive or disjunctive normal form.

Reduction to normal form

Reduction to normal form is relatively simple once a truth table for the formula is prepared. But further attempts to minimize the number of **literals** (see below) requires some tools: reduction by De Morgan's laws and truth tables can be unwieldy, but Karnaugh maps are very suitable a small number of variables (5 or less). Some sophisticated tabular methods exist for more complex circuits with multiple outputs but these are beyond the scope of this article; for more see Quine–McCluskey algorithm.

Literal, term and alterm

In electrical engineering a variable x or its negation $\sim(x)$ is lumped together into a single notion called a literal. A string of literals connected by ANDs is called a **term**. A string of literals connected by OR is called an **alterm**. Typically the literal $\sim(x)$ is abbreviated $\sim x$. Sometimes the $\&$ -symbol is omitted altogether in the manner of algebraic multiplication.

Example: a, b, c, d are variables. $((a \& \sim(b)) \& \sim(c)) \& d$ is a term. This can be abbreviated as $(a \& \sim b \& \sim c \& d)$, or $a \sim b \sim c d$.

Example: p, q, r, s are variables. $((p \& \sim(q)) \& r) \& \sim(s)$ is an alterm. This can be abbreviated as $(p \vee \sim q \vee r \vee \sim s)$.

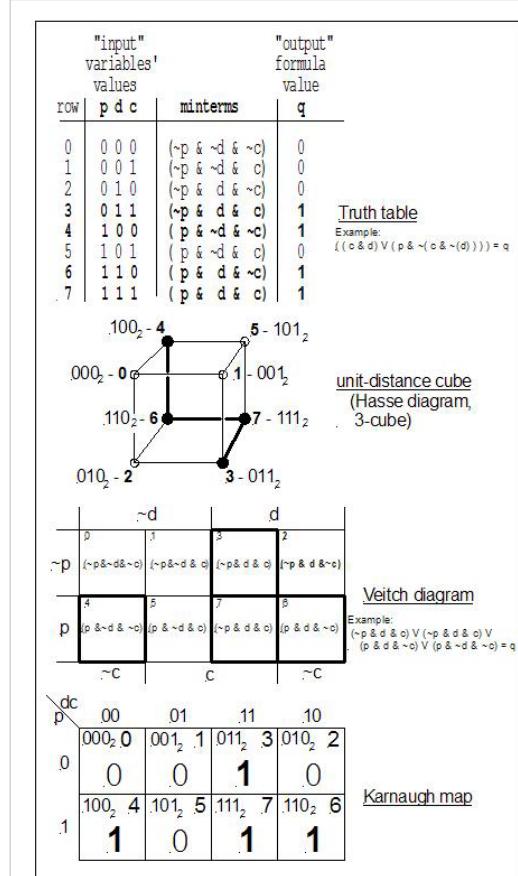
Minterms

In the same way that a 2^n -row truth table displays the evaluation of a propositional formula for all 2^n possible values of its variables, n variables produces a 2^n -square Karnaugh map (even though we cannot draw it in its full-dimensional realization). For example, 3 variables produces $2^3 = 8$ rows and 8 Karnaugh squares; 4 variables produces 16 truth-table rows and 16 squares and therefore 16 minterms. Each Karnaugh-map square and its corresponding truth-table evaluation represents one minterm.

Any propositional formula can be reduced to the "logical sum" (OR) of the active (i.e. "1"- or "T"-valued) minterms. When in this form the formula is said to be in disjunctive normal form. But even though it is in this form, it is not necessarily minimized with respect to either the number of terms or the number of literals.

In the following table, observe the peculiar numbering of the rows: (0, 1, 3, 2, 6, 7, 5, 4, 0). The first column is the decimal equivalent of the binary equivalent of the digits "cba", in other words:

Example: $cba_2 = c*2^2 + b*2^1 + a*2^0$



A truth table will contain 2^n rows, where n is the number of variables (e.g. three variables "p", "d", "c" produce 2^3 rows). Each row represents a minterm. Each minterm can be found on the Hasse diagram, on the Veitch diagram, and on the Karnaugh map. (The evaluations of "p" shown in the truth table are not shown in the Hasse, Veitch and Karnaugh diagrams; these are shown in the Karnaugh map of the following section.)

$$cba = (c=1, b=0, a=0) = 101_2 = 1*2^2 + 0*2^1 + 1*2^0 = 5_{10}$$

This numbering comes about because as one moves down the table from row to row only one variable at a time changes its value. Gray code is derived from this notion. This notion can be extended to three and four-dimensional hypercubes called Hasse diagrams where each corner's variables change only one at a time as one moves around the edges of the cube. Hasse diagrams (hypercubes) flattened into two dimensions are either Veitch diagrams or Karnaugh maps (these are virtually the same thing).

When working with Karnaugh maps one must always keep in mind that the top edge "wrap arounds" to the bottom edge, and the left edge wraps around to the right edge—the Karnaugh diagram is really a three- or four- or n-dimensional flattened object.

decimal equivalent of (c, b, a)	c	b	a	minterm
0	0	0	0	($\sim c \ \& \ \sim b \ \& \ \sim a$)
1	0	0	1	($\sim c \ \& \ \sim b \ \& \ a$)
3	0	1	1	($\sim c \ \& \ b \ \& \ a$)
2	0	1	0	($\sim c \ \& \ b \ \& \ \sim a$)
6	1	1	0	($c \ \& \ b \ \& \ \sim a$)
7	1	1	1	($c \ \& \ b \ \& \ a$)
5	1	0	1	($c \ \& \ \sim b \ \& \ a$)
4	1	0	0	($c \ \& \ \sim b \ \& \ \sim a$)
0	0	0	0	($\sim a \ \& \ \sim b \ \& \ \sim c$)

Reduction by use of the map method (Veitch, Karnaugh)

Veitch improved the notion of Venn diagrams by converting the circles to abutting squares, and Karnaugh simplified the Veitch diagram by converting the minterms, written in their literal-form (e.g. $\sim abc\sim d$) into numbers.^[21] The method proceeds as follows:

(1) Produce the formula's truth table

Produce the formula's truth table. Number its rows using the binary-equivalents of the variables (usually just sequentially 0 through n-1) for n variables.

Technically, the propositional function has been reduced to its (unminimized) conjunctive normal form: each row has its minterm expression and these can be OR'd to produce the formula in its (unminimized) conjunctive normal form.

Example: $((c \ \& \ d) \vee (p \ \& \ \sim(c \ \& \ (\sim d)))) = q$ in conjunctive normal form is:

$$((\sim p \ \& \ d \ \& \ c) \vee (p \ \& \ d \ \& \ c) \vee (p \ \& \ d \ \& \ \sim c) \vee (p \ \& \ \sim d \ \& \ \sim c)) = q$$

However, this formula be reduced both in the number of terms (from 4 to 3) and in the total count of its literals (12 to 6).

row	Minterms	p	d	c	((c	&	d)	V	(p	&	~	((c	&	~	(d)))	Active minterms	Formula in conjunctive normal form
0	$(\sim p \ \& \ \sim d \ \& \ \sim c)$	0	0	0		0	0	0	0	0	0	0	0	1		0	0	1	0	0	0	0	0				
1	$(\sim p \ \& \ \sim d \ \& \ c)$	0	0	1		1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0				
2	$(\sim p \ \& \ d \ \& \ \sim c)$	0	1	0		0	0	1	0	0	0	0	0	1		0	0	0	1	0	0	0	0				
3	$(\sim p \ \& \ d \ \& \ c)$	0	1	1		1	1	1	1	0	0	0	1		1	0	0	0	1	0	0	0	0	$(\sim p \ \& \ d \ \& \ c)$			
4	$(p \ \& \ \sim d \ \& \ \sim c)$	1	0	0		0	0	0	0	1		1	1	1		0	0	1	0	0	0	0	0	$(\sim p \ \& \ d \ \& \ c)$			
5	$(p \ \& \ \sim d \ \& \ c)$	1	0	1		1	0	0	0	0	1	1	0	0		1	1	1	0	0	0	0	0				
6	$(p \ \& \ d \ \& \ \sim c)$	1	1	0		0	0	1	1	0	1	1	1	1		0	0	0	1	0	0	0	0	$(p \ \& \ d \ \& \ \sim c)$			
7	$(p \ \& \ d \ \& \ c)$	1	1	1		0	1	1	1	1	1	1	1	1		1	0	0	1	0	0	0	0	$(p \ \& \ d \ \& \ c)$			
										q														$= (\sim p \ \& \ d \ \& \ c) \ V \ (\sim p \ \& \ d \ \& \ \sim c) \ V \ (p \ \& \ d \ \& \ \sim c) \ V \ (p \ \& \ d \ \& \ c)$			

(2) Create the formula's Karnaugh map

Use the values of the formula (e.g. "p") found by the truth-table method and place them in their into their respective (associated) Karnaugh squares (these are numbered per the Gray code convention). If values of "d" for "don't care" appear in the table, this adds flexibility during the reduction phase.

(3) Reduce minterms

Minterms of adjacent (abutting) 1-squares (T-squares) can be reduced with respect to the number of their literals, and the number terms also will be reduced in the process. Two abutting squares (2 x 1 horizontal or 1 x 2 vertical, even the edges represent abutting squares) lose one literal, four squares in a 4 x 1 rectangle (horizontal or vertical) or 2 x 2 square (even the four corners represent abutting squares) lose two literals, eight squares in a rectangle lose 3 literals, etc. (One seeks out the largest square or rectangles and ignores the smaller squares or rectangles contained totally within it.) This process continues until all abutting squares are accounted for, at which point the propositional formula is minimized.

For example, squares #3 and #7 abut. These two abutting squares can lose one literal (e.g. "p" from squares #3 and #7), four squares in a rectangle or square lose two literals, eight squares in a rectangle lose 3 literals, etc. (One seeks out the largest square or rectangles.) This process continues until all abutting squares are accounted for, at which point the propositional formula is said to be minimized.

Example: The map method usually is done by inspection. The following example expands the algebraic method to show the "trick" behind the combining of terms on a Karnaugh map:

Minterms #3 and #7 abut, #7 and #6 abut, and #4 and #6 abut (because the table's edges wrap around). So each of these pairs can be reduced.

Observe that by the Idempotency law ($A \vee A = A$), we can create more terms. Then by association and distributive laws the variables to disappear can be paired, and then "disappeared" with the Law of contradiction ($x \wedge \neg x = 0$). The following uses brackets [and] only to keep track of the terms; they have no special significance:

- Put the formula in conjunctive normal form with the formula to be reduced:

$$q = ((\neg p \wedge d \wedge c) \vee (p \wedge d \wedge c) \vee (p \wedge d \wedge \neg c) \vee (p \wedge \neg d \wedge \neg c)) = (\#3 \vee \#7 \vee \#6 \vee \#4)$$

- Idempotency (absorption) [$A \vee A = A$]:

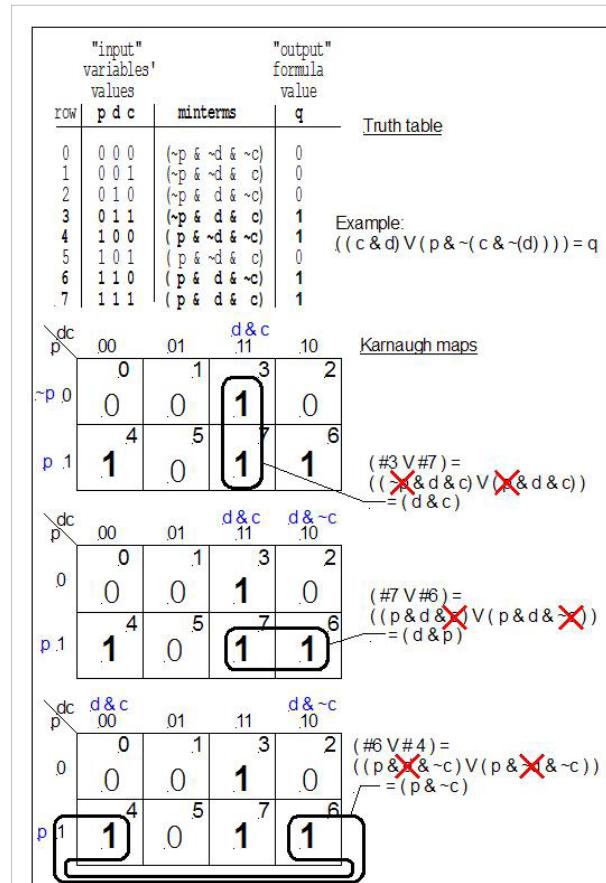
$$(\#3 \vee [\#7 \vee \#7] \vee [\#6 \vee \#6] \vee \#4)$$

- Associative law ($x \vee (y \vee z) = (x \vee y) \vee z$)

$$([\#3 \vee \#7] \vee [\#7 \vee \#6] \vee [\#6 \vee \#4])$$

$$[(\neg p \wedge d \wedge c) \vee (p \wedge d \wedge c)] \vee [(\neg p \wedge d \wedge c) \vee (p \wedge d \wedge \neg c)] \vee [(\neg p \wedge d \wedge c) \vee (p \wedge \neg d \wedge \neg c)].$$

- Distributive law ($x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$):



Steps in the reduction using a Karnaugh map. The final result is the OR (logical "sum") of the three reduced terms.

$$([(d \& c) V (\sim p \& p)] V [(p \& d) V (\sim c \& c)] V [(p \& \sim c) V (c \& \sim c)])$$

- Commutative law and law of contradiction $(x \& \sim x) = (\sim x \& x) = 0$:

$$([(d \& c) V (0)] V [(p \& d) V (0)] V [(p \& \sim c) V (0)])$$

- Law of identity $(x V 0) = x$ leading to the reduced form of the formula:

$$q = ((d \& c) V (p \& d) V (p \& \sim c))$$

(4) Verify reduction with a truth table

row	Minterms	p	d	c	$((d \& c) V (p \& d) V (p \& \sim c))$	$\sim ((d \& c) V (p \& d) V (p \& \sim c))$	
0	$(\sim p \& \sim d \& \sim c)$	0	0	0	0	0	1
1	$(\sim p \& \sim d \& c)$	0	0	1	0	0	0
2	$(\sim p \& d \& \sim c)$	0	1	0	0	0	1
3	$(\sim p \& d \& c)$	0	1	1	1	0	0
4	$(p \& \sim d \& \sim c)$	1	0	0	0	1	1
5	$(p \& \sim d \& c)$	1	0	1	0	1	0
6	$(p \& d \& \sim c)$	1	1	0	1	1	1
7	$(p \& d \& c)$	1	1	1	1	1	0
					q		

Impredicative propositions

Given the following examples-as-definitions, what does one make of the subsequent reasoning:

- (1) "This sentence is simple." (2) "This sentence is complex, and it is conjoined by AND."

Then assign the variable "s" to the left-most sentence "This sentence is simple". Define "compound" $c = \text{"not simple"}$ $\sim s$, and assign $c = \sim s$ to "This sentence is compound"; assign "j" to "It [this sentence] is conjoined by AND". The second sentence can be expressed as:

$$(\text{NOT}(s) \text{ AND } j)$$

If truth values are to be placed on the sentences $c = \sim s$ and j , then all are clearly FALSEHOODS: e.g. "This sentence is complex" is a FALSEHOOD (it is *simple*, by definition). So their conjunction (AND) is a falsehood. But when taken in its assembled form, the sentence a TRUTH.

This is an example of the paradoxes that result from an impredicative definition—that is, when an object m has a property P , but the object m is defined in terms of property P .^[22] The best advice for a rhetorician or one involved in deductive analysis is avoid impredicative definitions but at the same time be on the lookout for them because they can indeed create paradoxes. Engineers, on the other hand, put them to work in the form of propositional formulas with feedback.

Propositional formula with "feedback"

The notion of a propositional formula appearing as one of its own variables requires a formation rule that allows the assignment of the formula to a variable. In general there is no stipulation (either axiomatic or truth-table systems of objects and relations) that forbids this from happening.^[23]

The simplest case occurs when an OR formula becomes one its own inputs e.g. $p = q$. Begin with $(p \vee s) = q$, then let $p = q$. Observe that q 's "definition" depends on itself " q " as well as on " s " and the OR connective; this definition of q is thus **impredicative**. Either of two conditions can result:^[24] oscillation or memory.

It helps to think of the formula as a black box. Without knowledge of what is going on "inside" the formula-"box" from the outside it would appear that the output is no longer a function of the inputs alone. That is, sometimes one looks at q and sees 0 and other times 1. To avoid this problem one has to know the **state** (condition) of the "hidden" variable p inside the box (i.e. the value of q fed back and assigned to p). When this is known the apparent inconsistency goes away.

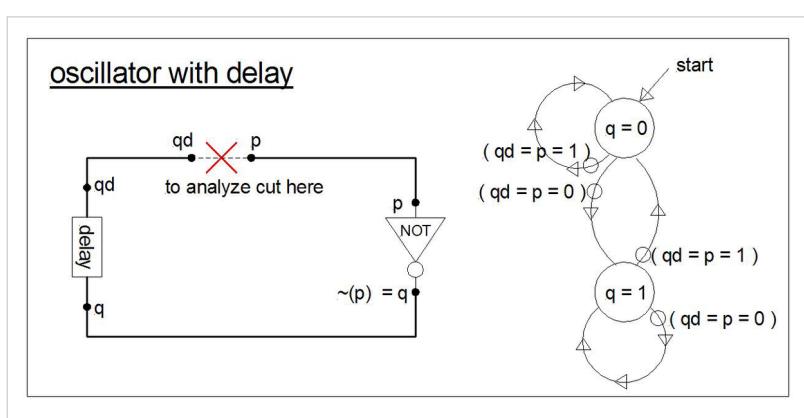
To understand [predict] the behavior of formulas with feedback requires the more sophisticated analysis of sequential circuits. Propositional formulas with feedback lead, in their simplest form, to state machines; they also lead to memories in the form of Turing tapes and counter-machine counters. From combinations of these elements one can build any sort of bounded computational model (e.g. Turing machines, counter machines, register machines, Macintosh computers, etc.).

Oscillation

In the abstract (ideal) case the simplest oscillating formula is a NOT fed back to itself: $\sim(\sim(p=q)) = q$. Analysis of an abstract (ideal) propositional formula in a truth-table reveals an inconsistency for both $p=1$ and $p=0$ cases: When $p=1$, $q=0$, this cannot be because $p=q$; ditto for when $p=0$ and $q=1$.

p	q	$\sim(p)$	$\sim(\sim(p))$	$=$	q
0	1	1	0	1	q & p inconsistent
1	0	0	1	0	q & p inconsistent

Oscillation with delay: If an delay^[25] (ideal or non-ideal) is inserted in the abstract formula between p and q then p will oscillate between 1 and 0: 101010...101... *ad infinitum*. If either of the delay and NOT are not abstract (i.e. not ideal), the type of analysis to be used will be dependent upon the exact nature of the objects that make up the oscillator; such things fall outside mathematics and into engineering.



Analysis requires a delay to be inserted and then the loop cut between the delay and the input "p". The delay must be viewed as a kind of proposition that has "qd" (q-delayed) as output for "q" as input. This new proposition adds another column to the truth table. The inconsistency is now between "qd" and "p" as shown in red; two stable states resulting:

			q				
qd	p	(\sim	(p)	=
							q
0	0	1		0		1	state 1
0	1	0		1		0	qd & p inconsistent
1	0	1		0		1	qd & p inconsistent
1	1	0		1		0	state 0

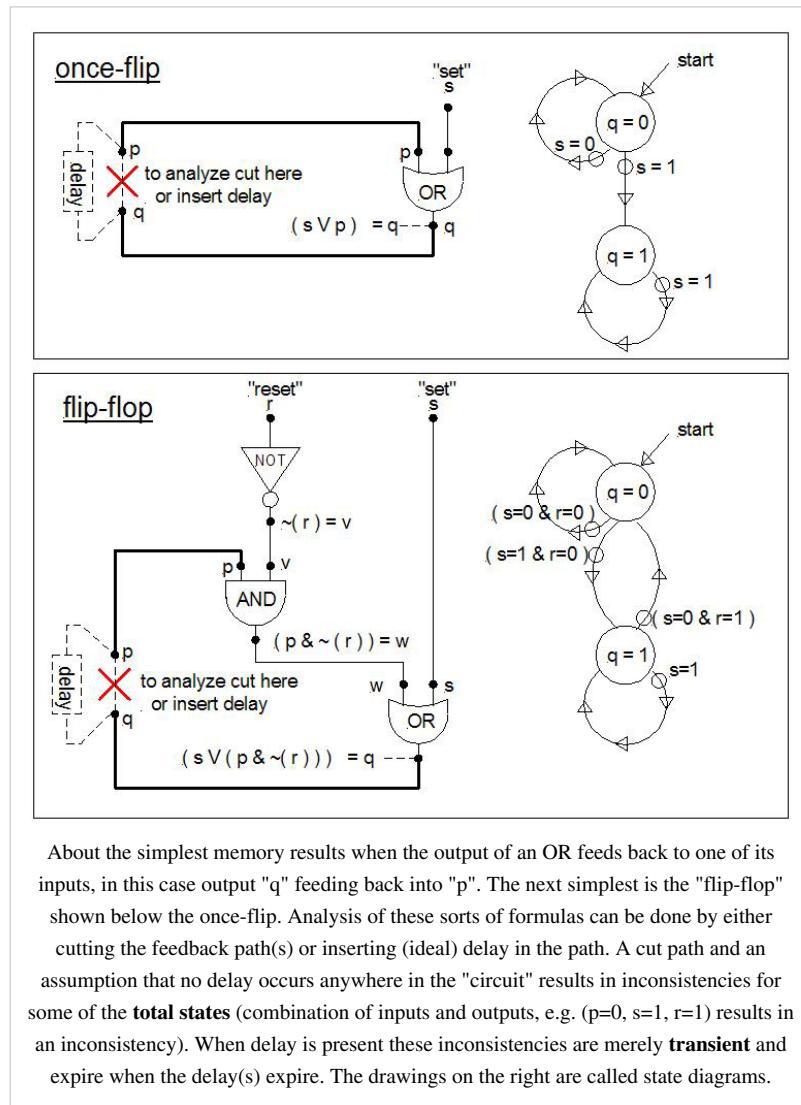
Memory

Without delay, inconsistencies must be eliminated from a truth table analysis. With the notion of "delay", this condition presents itself as a momentary inconsistency between the feed-back output variable q and $p = q_{\text{delayed}}$.

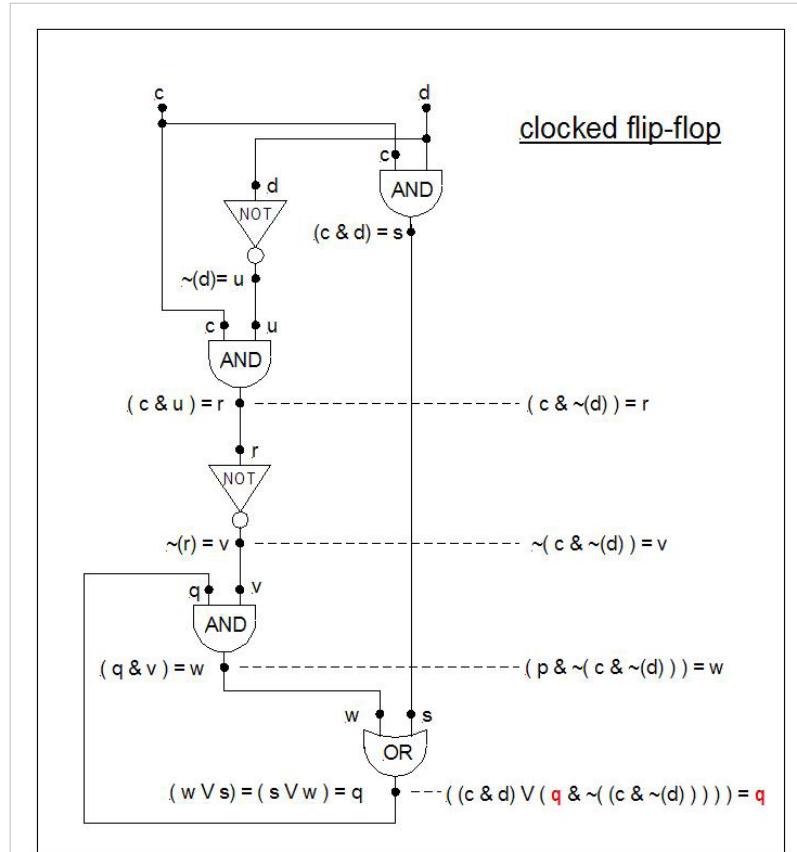
A truth table reveals the rows where inconsistencies occur between $p = q_{\text{delayed}}$ at the input and q at the output. After "breaking" the feed-back,^[26] the truth table construction proceeds in the conventional manner. But afterwards, in every row the output q is compared to the now-independent input p and any inconsistencies between p and q are noted (i.e. $p=0$ together with $q=1$, or $p=1$ and $q=0$); when the "line" is "remade" both are rendered impossible by the Law of contradiction $\sim(p \& \sim p)$. Rows revealing inconsistencies are either considered **transient states** or just eliminated as inconsistent and hence "impossible".

Once-flip memory

About the simplest memory results when the output of an OR feeds back to one of its inputs, in this case output "q" feeds back into "p". Given that the formula is first evaluated (initialized) with $p=0 \& q=0$, it will "flip" once when "set" by $s=1$. Thereafter, output



"q" will sustain "q" in the "flipped" condition (state q=1). This behavior, now time-dependent, is shown by the state diagram to the right of the once-flip.



A "clocked flip-flop" memory ("c" is the "clock" and "d" is the "data"). The data can change at any time when clock $c=0$; when clock $c=1$ the output q "tracks" the value of data d . When c goes from 1 to 0 it "traps" $d = q$'s value and this continues to appear at q no matter what d does (as long as c remains 0).

			q				
p	s	(s	V	p)	=
0	0	0	0	0	0	0	state 0, $s=0$
0	1	1	1	0			q & p inconsistent
1	0	0	1	1	1	1	state 1 with $s = 0$
1	1	1	1	1	1	1	state 1 with $s = 1$

Flip-flop memory

The next simplest case is the "set-reset" flip-flop shown below the once-flip. Given that $r=0 \& s=0$ and $q=0$ at the outset, it is "set" ($s=1$) in a manner similar to the once-flip. It however has a provision to "reset" $q=0$ when " $r=1$ ". And additional complication occurs if both set=1 and reset=1. In this formula, the set=1 forces the output $q=1$ so when and if $(s=0 \& r=1)$ the flip-flop will be reset. Or, if $(s=1 \& r=0)$ the flip-flop will be set. In the abstract (ideal) instance in which $s=1 \Rightarrow s=0 \& r=1 \Rightarrow r=0$ simultaneously, the formula q will be indeterminate (undecidable). Due to delays in "real" OR, AND and NOT the result will be unknown at the outset but thereafter predictable.

				q										
p	s	r	(s	V	(p	&	~	(r))	=
0	0	0	0	0	0	0	0	1	0	0	0	0	0	state 0 with (s=0 & r=0)
0	0	1	0	0	0	0	0	0	1	0	0	0	0	state 0 with (s=0 & r=1)
0	1	0	1	1	0	0	0	1	0	0	0	0	0	q & p inconsistent
0	1	1	1	1	0	0	0	0	1	0	0	0	0	q & p inconsistent
1	0	0	0	1	1	1	1	1	0	0	0	0	1	state 1 with (s=0 & r=0)
1	0	1	0	0	0	1	0	0	0	1	0	0	0	q & p inconsistent
1	1	0	1	1	1	1	1	1	0	0	0	0	1	state 1 with (s=1 & r=0)
1	1	1	1	1	1	1	0	0	0	1	0	0	1	state 1 with s & r simultaneously 1

Clocked flip-flop memory

The formula known as "clocked flip-flop" memory ("c" is the "clock" and "d" is the "data") is given below. It works as follows: When $c = 0$ the data d (either 0 or 1) cannot "get through" to affect output q . When $c = 1$ the data d "gets through" and output q "follows" d 's value. When c goes from 1 to 0 the last value of the data remains "trapped" at output "q". As long as $c=0$, d can change value without causing q to change.

Example: $((c \& d) \vee (p \& (\sim(c \& \sim(d)))) = q$, but now let $p = q$:

Example: $((c \& d) \vee (q \& (\sim(c \& \sim(d)))) = q$

The state diagram is similar in shape to the flip-flop's state diagram, but with different labelling on the **transitions**.

Historical development

Bertrand Russell (1912:74) lists three laws of thought that derive from Aristotle: (1) The law of identity: "Whatever is, is.", (2) The law of contradiction: "Nothing cannot both be and not be", and (3) The law of excluded middle: "Everything must be or not be."

Example: Here O is an expression about an objects BEING or QUALITY:

- (1) Law of Identity: $O = O$
- (2) Law of contradiction: $\sim(O \ \& \ \sim(O))$
- (3) Law of excluded middle: $(O \vee \sim(O))$

The use of the word "everything" in the law of excluded middle renders Russell's expression of this law open to debate. If restricted to an expression about BEING or QUALITY with reference to a finite collection of objects (a finite "universe of discourse") -- the members of which can be investigated one after another for the presence or absence of the assertion--then the law is considered intuitionistically appropriate. Thus an assertion such as: "This object must either BE or NOT BE (in the collection)", or "This object must either have this QUALITY or NOT have this QUALITY (relative to the objects in the collection)" is acceptable. See more at Venn diagram.

Although a propositional calculus originated with Aristotle, the notion of an *algebra* applied to propositions had to wait until the early 19th century. In an (adverse) reaction to the 2000 year tradition of Aristotle's syllogisms, John Locke's *Essay concerning human understanding* (1690) used the word semiotics (theory of the use of symbols). By 1826 Richard Whately had critically analyzed the syllogistic logic with a sympathy toward Locke's semiotics. George Bentham's work (1827) resulted in the notion of "quantification of the predicate" (1827) (nowadays symbolized as $\forall \equiv$ "for all"). A "row" instigated by William Hamilton over a priority dispute with Augustus De Morgan "inspired George Boole to write up his ideas on logic, and to publish them as MAL [Mathematical Analysis of Logic] in 1847" (Grattan-Guinness and Bornet 1997:xxviii).

About his contribution Grattan-Guinness and Bornet comment:

"Boole's principal single innovation was [the] law [$x^n = x$] for logic: it stated that the mental acts of choosing the property x and choosing x again and again is the same as choosing x once... As consequence of it he formed the equations $x \cdot (1-x) = 0$ and $x + (1-x) = 1$ which for him expressed respectively the law of contradiction and the law of excluded middle" (p. xxviiff). For Boole "1" was the universe of discourse and "0" was nothing.

Gottlob Frege's massive undertaking (1879) resulted in a formal calculus of propositions, but his symbolism is so daunting that it had little influence excepting on one person: Bertrand Russell. First as the student of Alfred North Whitehead he studied Frege's work and suggested a (famous and notorious) emendation with respect to it (1904) around the problem of an antinomy that he discovered in Frege's treatment (cf Russell's paradox). Russell's work led to a collaboration with Whitehead that, in the year 1912, produced the first volume of *Principia Mathematica* (PM). It is here that what we consider "modern" propositional logic first appeared. In particular, PM introduces NOT and OR and the assertion symbol \vdash as primitives. In terms of these notions they define IMPLICATION \rightarrow (def. *1.01: $\sim p \vee q$), then AND (def. *3.01: $\sim(\sim p \vee \sim q)$), then EQUIVALENCE $p \leftrightarrow q$ (*4.01: $(p \rightarrow q) \ \& \ (q \rightarrow p)$).

- Henry M. Sheffer (1921) and Jean Nicod demonstrate that only one connective, the "stroke" \mid is sufficient to express all propositional formulas.
- Emil Post (1921) develops the truth-table method of analysis in his "Introduction to a general theory of elementary propositions". He notes Nicod's stroke \mid .
- Whitehead and Russell add an introduction to their 1927 re-publication of PM adding, in part, a favorable treatment of the "stroke".

Computation and switching logic:

- William Eccles and F. W. Jordan (1919) describe a "trigger relay" made from a vacuum tube.
- George Stibitz (1937) invents the binary adder using mechanical relays. He builds this on his kitchen table.

Example: Given binary bits a_i and b_i and carry-in (c_{in_i}), their summation Σ_i and carry-out (c_{out_i}) are:

- $((a_i \text{ XOR } b_i) \text{ XOR } c_{in_i}) = \Sigma_i$
- $(a_i \& b_i) \vee c_{in_i} = c_{out_i}$
- Alan Turing builds a multiplier using relays (1937–1938). He has to hand-wind his own relay coils to do this.
- Textbooks about "switching circuits" appear in early 1950s.
- Willard Quine 1952 and 1955, E. W. Veitch 1952, and M. Karnaugh (1953) develop map-methods for simplifying propositional functions.
- George H. Mealy (1955) and Edward F. Moore (1956) address the theory of sequential (i.e. switching-circuit) "machines".
- E. J. McCluskey and H. Shorr develop a method for simplifying propositional (switching) circuits (1962).

Footnotes

- [1] Hamilton 1978:1
- [2] PM p. 91 eschews "the" because they require a clear-cut "object of sensation"; they stipulate the use of "this"
- [3] (italics added) Reichenbach p.80.
- [4] Tarski p.54-68. Suppes calls IDENTITY a "further rule of inference" and has a brief development around it; Robbin, Bender and Williamson, and Goodstein introduce the sign and its usage without comment or explanation. Hamilton p. 37 employs two signs ≠ and = with respect to the **valuation** of a formula in a formal calculus. Kleene p. 70 and Hamilton p. 52 place it in the predicate calculus, in particular with regards to the arithmetic of natural numbers.
- [5] Empiricists eschew the notion of *a priori* (built-in, born-with) knowledge. "Radical reductionists" such as John Locke and David Hume "held that every idea must either originate directly in sense experience or else be compounded of ideas thus originating"; quoted from Quine reprinted in 1996 *The Emergence of Logical Empiricism*, Garland Publishing Inc. <http://www.marxists.org/reference/subject/philosophy/works/us/quine.htm>
- [6] Neural net modelling offers a good mathematical model for a comparator as follows: Given a signal S and a threshold "thr", subtract "thr" from S and substitute this difference d to a sigmoid function: For large "gains" k, e.g. k=100, $1/(1 + e^{-k*(d)}) = 1/(1 + e^{-k*(S-thr)}) = \{\approx 0, \approx 1\}$. For example, if "The door is DOWN" means "The door is less than 50% of the way up", then a threshold thr=0.5 corresponding to 0.5*5.0 = +2.50 volts could be applied to a "linear" measuring-device with an output of 0 volts when fully closed and +5.0 volts when fully open.
- [7] In actuality the digital 1 and 0 are defined over non-overlapping ranges e.g. { "1" = +5/+0.2/-1.0 volts, 0 = +0.5/-0.2 volts }. When a value falls outside the defined range(s) the value becomes "u" -- unknown; e.g. +2.3 would be "u".
- [8] While the notion of logical product is not so peculiar (e.g. $0*0=0, 0*1=0, 1*0=0, 1*1=1$), the notion of $(1+1=1)$ is peculiar; in fact $(a + b) = (a + (b - a*b))$ where "+" is the "logical sum" but + and - are the true arithmetic counterparts. Occasionally all four notions do appear in a formula: A AND B = $1/2 * (A + B - (A \text{ XOR } B))$ (cf p. 146 in John Wakerly 1978, *Error Detecting Codes, Self-Checking Circuits and Applications*, North-Holland, New York, ISBN 0-444-00259-6 pbk.)
- [9] A careful look at its Karnaugh map shows that IF...THEN...ELSE can also be expressed, in a rather round-about way, in terms of two exclusive-ORs: $((b \text{ AND } (c \text{ XOR } a)) \text{ OR } (a \text{ AND } (c \text{ XOR } b))) = d$.
- [10] Robbin p. 3.
- [11] Rosenbloom p. 30 and p. 54ff discusses this problem of implication at some length. Most philosophers and mathematicians just accept the material definition as given above. But some do not, including the intuitionists; they consider it a form of the law of excluded middle misapplied.
- [12] Indeed, exhaustive selection between alternatives -- **mutual exclusion** -- is required by the definition that Kleene gives the CASE operator (Kleene 1952:229)
- [13] The use of quote marks around the expressions is not accidental. Tarski comments on the use of quotes in his "18. Identity of things and identity of their designations; use of quotation marks" p. 58ff.
- [14] Hamilton p. 37. Bender and Williamson p. 29 state "In what follows, we'll replace "equals" with the symbol "↔" (equivalence) which is usually used in logic. We use the more familiar "≡" for assigning meaning and values."
- [15] Reichenbach p. 20-22 and follows the conventions of PM. The symbol $=_{Df}$ is in the metalanguage and is not a formal symbol with the following meaning: "by symbol 's' is to have the same meaning as the formula '(c & d)'".
- [16] Rosenbloom 1950:32. Kleene 1952:73-74 ranks all 11 symbols.
- [17] cf Minsky 1967:75, section 4.2.3 "The method of parenthesis counting". Minsky presents a state machine that will do the job, and by use of induction (recursive definition) Minsky proves the "method" and presents a theorem as the result. A fully generalized "parenthesis grammar" requires an infinite state machine (e.g. a Turing machine) to do the counting.
- [18] Robbin p. 7
- [19] cf Reichenbach p. 68 for a more involved discussion: "If the inference is valid and the premises are true, the inference is called *conclusive*".
- [20] As well as the first three, Hamilton pp.19-22 discusses logics built from only \perp (NAND), and \downarrow (NOR).

- [21] Wickes 1967:36ff. Wickes offers a good example of 8 of the 2×4 (3-variable maps) and 16 of the 4×4 (4-variable) maps. As an arbitrary 3-variable map could represent any one of $2^8 = 256$ 2×4 maps, and an arbitrary 4-variable map could represent any one of $2^{16} = 65,536$ different formula-evaluations, writing down every one is infeasible.
- [22] This definition is given by Stephen Kleene. Both Kurt Gödel and Kleene believed that the classical paradoxes are uniformly examples of this sort of definition. But Kleene went on to assert that the problem has not been solved satisfactorily and impredicative definitions can be found in analysis. He gives as example the definition of the least upper bound (l.u.b.) \mathbf{u} of \mathbf{M} . Given a Dedekind cut of the number line \mathbf{C} and the two parts into which the number line is cut, i.e. \mathbf{M} and $(\mathbf{C} - \mathbf{M})$, l.u.b. = \mathbf{u} is defined in terms of the notion \mathbf{M} , whereas \mathbf{M} is defined in terms of \mathbf{C} . Thus the definition of \mathbf{u} , an element of \mathbf{C} , is defined in terms of the totality \mathbf{C} and this makes its definition impredicative. Kleene asserts that attempts to argue this away can be used to uphold the impredicative definitions in the paradoxes.(Kleene 1952:43).
- [23] McCluskey comments that "it could be argued that the analysis is still incomplete because the word statement "The outputs are equal to the previous values of the inputs" has not been obtained"; he goes on to dismiss such worries because "English is not a formal language in a mathematical sense, [and] it is not really possible to have a *formal* procedure for obtaining word statements" (p. 185).
- [24] More precisely, given enough "loop gain", either **oscillation** or **memory** will occur (cf McCluskey p. 191-2). In abstract (idealized) mathematical systems adequate loop gain is not a problem.
- [25] The notion of delay and the principle of local causation as caused ultimately by the speed of light appears in Robin Gandy (1980), "Church's thesis and Principles for Mechanisms", in J. Barwise, H. J. Keisler and K. Kunen, eds., *The Kleene Symposium*, North-Holland Publishing Company (1980) 123-148. Gandy considered this to be the most important of his principles: "Contemporary physics rejects the possibility of instantaneous action at a distance" (p. 135). Gandy was Alan Turing's student and close friend.
- [26] McKlusky p. 194-5 discusses "breaking the loop" and inserts "amplifiers" to do this; Wickes (p. 118-121) discuss inserting delays. McCluskey p. 195ff discusses the problem of "races" caused by delays.

References

- Bender, Edward A. and Williamson, S. Gill, 2005, *A Short Course in Discrete Mathematics*, Dover Publications, Mineola NY, ISBN 0-486-43946-1. This text is used in a "lower division two-quarter [computer science] course" at UC San Diego.
- Enderton, H. B., 2002, *A Mathematical Introduction to Logic*. Harcourt/Academic Press. ISBN 0-12-238452-0
- Goodstein, R. L., (Pergamon Press 1963), 1966, (Dover edition 2007), *Boolean Algebra*, Dover Publications, Inc. Minola, New York, ISBN 0-486-45894-6. Emphasis on the notion of "algebra of classes" with set-theoretic symbols such as \cap , \cup , ' (NOT), \subset (IMPLIES). Later Goldstein replaces these with &, \vee , \neg , \rightarrow (respectively) in his treatment of "Sentence Logic" pp. 76–93.
- Ivor Grattan-Guinness and Gérard Bornet 1997, *George Boole: Selected Manuscripts on Logic and its Philosophy*, Birkhäuser Verlag, Basel, ISBN 0-876-5456-9 (Boston).
- A. G. Hamilton 1978, *Logic for Mathematicians*, Cambridge University Press, Cambridge UK, ISBN 0-521-21838-1.
- E. J. McCluskey 1965, *Introduction to the Theory of Switching Circuits*, McGraw-Hill Book Company, New York. No ISBN. Library of Congress Catalog Card Number 65-17394. McCluskey was a student of Willard Quine and developed some notable theorems with Quine and on his own. For those interested in the history, the book contains a wealth of references.
- Marvin L. Minsky 1967, *Computation: Finite and Infinite Machines*, Prentice-Hall, Inc, Englewood Cliffs, N.J.. No ISBN. Library of Congress Catalog Card Number 67-12342. Useful especially for computability, plus good sources.
- Paul C. Rosenbloom 1950, Dover edition 2005, *The Elements of Mathematical Logic*, Dover Publications, Inc., Mineola, New York, ISBN 0-486-44617-4.
- Joel W. Robbin 1969, 1997, *Mathematical Logic: A First Course*, Dover Publications, Inc., Mineola, New York, ISBN 0-486-45018-X (pbk.).
- Patrick Suppes 1957 (1999 Dover edition), *Introduction to Logic*, Dover Publications, Inc., Mineola, New York. ISBN 0-486-40687-3 (pbk.). This book is in print and readily available.
- On his page 204 in a footnote he references his set of axioms to E. V. Huntington, "Sets of Independent Postulates for the Algebra of Logic", *Transactions of the American Mathematical Society*, Vol. 5 91904) pp. 288-309.
- Alfred Tarski 1941 (1995 Dover edition), *Introduction to Logic and to the Methodology of Deductive Sciences*, Dover Publications, Inc., Mineola, New York. ISBN 0-486-28462-X (pbk.). This book is in print and readily

available.

- Jean van Heijenoort 1967, 3rd printing with emendations 1976, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, Harvard University Press, Cambridge, Massachusetts. ISBN 0-674-32449-8 (pbk.) Translation/reprints of Frege (1879), Russell's letter to Frege (1902) and Frege's letter to Russell (1902), Richard's paradox (1905), Post (1921) can be found here.
- Alfred North Whitehead and Bertrand Russell 1927 2nd edition, paperback edition to *53 1962, *Principia Mathematica*, Cambridge University Press, no ISBN. In the years between the first edition of 1912 and the 2nd edition of 1927, H. M. Sheffer 1921 and M. Jean Nicod (no year cited) brought to Russell's and Whitehead's attention that what they considered their primitive propositions (connectives) could be reduced to a single \perp , nowadays known as the "stroke" or NAND (NOT-AND, NEITHER ... NOR...). Russell-Whitehead discuss this in their "Introduction to the Second Edition" and makes the definitions as discussed above.
- William E. Wickes 1968, *Logic Design with Integrated Circuits*, John Wiley & Sons, Inc., New York. No ISBN. Library of Congress Catalog Card Number: 68-21185. Tight presentation of engineering's analysis and synthesis methods, references McCluskey 1965. Unlike Suppes, Wickes' presentation of "Boolean algebra" starts with a set of postulates of a truth-table nature and then derives the customary theorems of them (p. 18ff).

Stone duality

In mathematics, there is an ample supply of categorical dualities between certain categories of topological spaces and categories of partially ordered sets. Today, these dualities are usually collected under the label **Stone duality**, since they form a natural generalization of Stone's representation theorem for Boolean algebras. These concepts are named in honor of Marshall Stone. Stone-type dualities also provide the foundation for pointless topology and are exploited in theoretical computer science for the study of formal semantics.

This article gives pointers to special cases of Stone duality and explains a very general instance thereof in detail.

Overview of Stone-type dualities

Probably the most general duality which is classically referred to as "Stone duality" is the duality between the category **Sob** of sober spaces with continuous functions and the category **SFrm** of spatial frames with appropriate frame homomorphisms. The dual category of **SFrm** is the category of locales denoted by **SLoc**. The categorical equivalence of **Sob** and **SLoc** is the basis for the mathematical area of pointless topology, that is devoted to the study of **Loc** – the category of all locales of which **SLoc** is a full subcategory. The involved constructions are characteristic for this kind of duality, and are detailed below.

Now one can easily obtain a number of other dualities by restricting to certain special classes of sober spaces:

- The category **CohSp** of coherent sober spaces (and coherent maps) is equivalent to the category **CohLoc** of coherent (or spectral) locales (and coherent maps), on the assumption of the Boolean prime ideal theorem (in fact, this statement is equivalent to that assumption). The significance of this result stems from the fact that **CohLoc** in turn is dual to the category **DLat** of distributive lattices. Hence, **DLat** is dual to **CohSp** – one obtains Stone's representation theorem for distributive lattices.
- When restricting further to coherent sober spaces which are Hausdorff, one obtains the category **Stone** of so-called Stone spaces. On the side of **DLat**, the restriction yields the subcategory **Bool** of Boolean algebras. Thus one obtains Stone's representation theorem for Boolean algebras.
- Stone's representation for distributive lattices can be extended via an equivalence of coherent spaces and Priestley spaces (ordered topological spaces, that are compact and totally order-disconnected). One obtains a representation of distributive lattices via ordered topologies: Priestley's representation theorem for distributive lattices.

Many other Stone-type dualities could be added to these basic dualities.

Duality of sober spaces and spatial locales

This section motivates and explains one of the most basic constructions of Stone duality: the duality between topological spaces which are *sober* and frames (i.e. complete Heyting algebras) which are *spatial*. This classical piece of mathematics requires a substantial amount of abstraction that usually tends to puzzle beginners. It should therefore be considered as graduate level mathematics. Some prior exposure to the basics of category theory is recommended, although a deep understanding of the concepts of adjunction and duality may well arise from examples such as the result below. Furthermore, concepts of topology and order theory are naturally involved as well, where the latter is probably more important for a thorough understanding.

The lattice of open sets

The starting point for the theory is the fact that every topological space is characterized by a set of points X and a system $\Omega(X)$ of open sets of elements from X , i.e. a subset of the powerset of X . It is known that $\Omega(X)$ has certain special properties: it is a complete lattice within which suprema and finite infima are given by set unions and finite set intersections, respectively. Furthermore, it contains both X and the empty set. Since the embedding of $\Omega(X)$ into the powerset lattice of X preserves finite infima and arbitrary suprema, $\Omega(X)$ inherits the following distributivity law:

$$x \wedge \bigvee S = \bigvee \{ x \wedge s : s \in S \},$$

for every element (open set) x and every subset S of $\Omega(X)$. Hence $\Omega(X)$ is not an arbitrary complete lattice but a *complete Heyting algebra* (also called *frame* or *locale* – the various names are primarily used to distinguish several categories that have the same class of objects but different morphisms: frame morphisms, locale morphisms and homomorphisms of complete Heyting algebras). Now an obvious question is: To what extent is a topological space characterized by its locale of open sets?

As already hinted at above, one can go even further. The category **Top** of topological spaces has as morphisms the continuous functions, where a function f is continuous if the inverse image $f^{-1}(O)$ of any open set in the codomain of f is open in the domain of f . Thus any continuous function f from a space X to a space Y defines an inverse mapping f^{-1} from $\Omega(Y)$ to $\Omega(X)$. Furthermore, it is easy to check that f^{-1} (like any inverse image map) preserves finite intersections and arbitrary unions and therefore is a *morphism of frames*. If we define $\Omega(f) = f^{-1}$ then Ω becomes a contravariant functor from the category **Top** to the category **Frm** of frames and frame morphisms. Using the tools of category theory, the task of finding a characterization of topological spaces in terms of their open set lattices is equivalent to finding a functor from **Frm** to **Top** which is adjoint to Ω .

Points of a locale

The goal of this section is to define a functor pt from **Frm** to **Top** that in a certain sense "inverts" the operation of Ω by assigning to each locale L a set of points $\text{pt}(L)$ (hence the notation pt) with a suitable topology. But how can we recover the set of points just from the locale, though it is not given as a lattice of sets? It is certain that one cannot expect in general that pt can reproduce all of the original elements of a topological space just from its lattice of open sets – for example all sets with the indiscrete topology yield (up to isomorphism) the same locale, such that the information on the specific set is no longer present. However, there is still a reasonable technique for obtaining "points" from a locale, which indeed gives an example of a central construction for Stone-type duality theorems.

Let us first look at the points of a topological space X . One is usually tempted to consider a point of X as an element x of the set X , but there is in fact a more useful description for our current investigation. Any point x gives rise to a continuous function p_x from the one element topological space 1 (all subsets of which are open) to the space X by defining $p_x(1) = x$. Conversely, any function from 1 to X clearly determines one point: the element that it "points" to. Therefore the set of points of a topological space is equivalently characterized as the set of functions from 1 to X .

When using the functor Ω to pass from **Top** to **Frm**, all set-theoretic elements of a space are lost, but – using a fundamental idea of category theory – one can as well work on the function spaces. Indeed, any "point" $p_x : 1 \rightarrow X$ in

Top is mapped to a morphism $\Omega(p_x): \Omega(X) \rightarrow \Omega(1)$. The open set lattice of the one-element topological space $\Omega(1)$ is just (isomorphic to) the two-element locale $2 = \{0, 1\}$ with $0 < 1$. After these observations it appears reasonable to define the set of points of a locale L to be the set of frame morphisms from L to 2 . Yet, there is no guarantee that every point of the locale $\Omega(X)$ is in one-to-one correspondence to a point of the topological space X (consider again the indiscrete topology, for which the open set lattice has only one "point").

Before defining the required topology on $\text{pt}(X)$, it is worthwhile to clarify the concept of a point of a locale further. The perspective motivated above suggests to consider a point of a locale L as a frame morphism p from L to 2 . But these morphisms are characterized equivalently by the inverse images of the two elements of 2 . From the properties of frame morphisms, one can derive that $p^{-1}(0)$ is a lower set (since p is monotone), which contains a greatest element $a_p = \bigvee p^{-1}(0)$ (since p preserves arbitrary suprema). In addition, the principal ideal $p^{-1}(0)$ is a prime ideal since p preserves finite infima and thus the principal a_p is a meet-prime element. Now the set-inverse of $p^{-1}(0)$ given by $p^{-1}(1)$ is a completely prime filter because $p^{-1}(0)$ is a principal prime ideal. It turns out that all of these descriptions uniquely determine the initial frame morphism. We sum up:

A point of a locale L is equivalently described as:

- a frame morphism from L to 2
- a principal prime ideal of L
- a meet-prime element of L
- a completely prime filter of L .

All of these descriptions have their place within the theory and it is convenient to switch between them as needed.

The functor **pt**

Now that a set of points is available for any locale, it remains to equip this set with an appropriate topology in order to define the object part of the functor **pt**. This is done by defining the open sets of $\text{pt}(L)$ as

$$\varphi(a) = \{p \in \text{pt}(L) \mid p(a) = 1\},$$

for every element a of L . Here we viewed the points of L as morphisms, but one can of course state a similar definition for all of the other equivalent characterizations. It can be shown that setting $\Omega(\text{pt}(L)) = \{\varphi(a) \mid a \in L\}$ does really yield a topological space $(\text{pt}(L), \Omega(\text{pt}(L)))$. It is common to abbreviate this space as $\text{pt}(L)$.

Finally **pt** can be defined on morphisms of **Frm** rather canonically by defining, for a frame morphism g from L to M , $\text{pt}(g): \text{pt}(M) \rightarrow \text{pt}(L)$ as $\text{pt}(g)(p) = p \circ g$. In words, we obtain a morphism from L to 2 (a point of L) by applying the morphism g to get from L to M before applying the morphism p that maps from M to 2 . Again, this can be formalized using the other descriptions of points of a locale as well – for example just calculate $(p \circ g)^{-1}(0)$.

The adjunction of **Top** and **Loc**

As noted several times before, **pt** and Ω usually are not inverses. In general neither is X homeomorphic to $\text{pt}(\Omega(X))$ nor is L order-isomorphic to $\Omega(\text{pt}(L))$. However, when introducing the topology of $\text{pt}(L)$ above, a mapping φ from L to $\Omega(\text{pt}(L))$ was applied. This mapping is indeed a frame morphism. Conversely, we can define a continuous function ψ from X to $\text{pt}(\Omega(X))$ by setting $\psi(x) = \Omega(p_x)$, where p_x is just the characteristic function for the point x from 1 to X as described above. Another convenient description is given by viewing points of a locale as meet-prime elements. In this case we have $\psi(x) = X \setminus \text{Cl}\{x\}$, where $\text{Cl}\{x\}$ denotes the topological closure of the set $\{x\}$ and \setminus is just set-difference.

At this point we already have more than enough data to obtain the desired result: the functors Ω and **pt** define an adjunction between the categories **Top** and **Loc** = **Frm**^{op}, where **pt** is right adjoint to Ω and the natural transformations ψ and φ^{op} provide the required unit and counit, respectively.

The duality theorem

The above adjunction is not an equivalence of the categories **Top** and **Loc** (or, equivalently, a duality of **Top** and **Frm**). For this it is necessary that both ψ and φ are isomorphisms in their respective categories.

For a space X , $\psi: X \rightarrow \text{pt}(\Omega(X))$ is a homeomorphism if and only if it is bijective. Using the characterization via meet-prime elements of the open set lattice, one sees that this is the case if and only if every meet-prime open set is of the form $X \setminus \text{Cl}\{x\}$ for a unique x . Alternatively, every join-prime closed set is the closure of a unique point, where "join-prime" can be replaced by (join-) irreducible since we are in a distributive lattice. Spaces with this property are called **sober**.

Conversely, for a locale L , $\varphi: L \rightarrow \Omega(\text{pt}(L))$ is always surjective. It is additionally injective if and only if any two elements a and b of L for which a is not less-or-equal to b can be separated by points of the locale, formally:

if not $a \leq b$, then there is a point p in $\text{pt}(L)$ such that $p(a) = 1$ and $p(b) = 0$.

If this condition is satisfied for all elements of the locale, then the locale is **spatial**, or said to have enough points.

Finally, one can verify that for every space X , $\Omega(X)$ is spatial and for every locale L , $\text{pt}(L)$ is sober. Hence, it follows that the above adjunction of **Top** and **Loc** restricts to an equivalence of the full subcategories **Sob** of sober spaces and **SLoc** of spatial locales. This main result is completed by the observation that for the functor $\text{pt} \circ \Omega$, sending each space to the points of its open set lattice is right adjoint to the inclusion functor from **Sob** to **Top**. For a space X , $\text{pt}(\Omega(X))$ is called its **soberification**. The case of the functor $\Omega \circ \text{pt}$ is symmetric but a special name for this operation is not commonly used.

References

- Burris, Stanley N., and H.P. Sankappanavar, H. P., 1981. *A Course in Universal Algebra*.^[1] Springer-Verlag. ISBN 3-540-90578-2. (available free online at the website mentioned)
- P. T. Johnstone, *Stone Spaces*, Cambridge Studies in Advanced Mathematics 3, Cambridge University Press, Cambridge, 1982. ISBN 0-521-23893-5.
- Pedicchio, Maria Cristina; Tholen, Walter, eds. (2004). *Categorical foundations. Special topics in order, topology, algebra, and sheaf theory*. Encyclopedia of Mathematics and Its Applications 97. Cambridge: Cambridge University Press. ISBN 0-521-83414-7. Zbl 1034.18001^[2].
- Vickers, Steven (1989). *Topology via logic*. Cambridge Tracts in Theoretical Computer Science 5. Cambridge: Cambridge University Press. ISBN 0-521-36062-5. Zbl 0668.54001^[3].

References

- [1] <http://www.thoralf.uwaterloo.ca/htdocs/ualg.html>
[2] <http://www.zentralblatt-math.org/zmath/en/search/?format=complete&q=an:1034.18001>
[3] <http://www.zentralblatt-math.org/zmath/en/search/?format=complete&q=an:0668.54001>

Stone functor

In mathematics, the **Stone functor** is a functor $S: \mathbf{Top}^{\text{op}} \rightarrow \mathbf{Bool}$, where **Top** is the category of topological spaces and **Bool** is the category of Boolean algebras and Boolean homomorphisms. It assigns to each topological space X the Boolean algebra $S(X)$ of its clopen subsets, and to each morphism $f^{\text{op}}: X \rightarrow Y$ in **Top**^{op} (i.e., a continuous map $f: Y \rightarrow X$) the homomorphism $S(f): S(X) \rightarrow S(Y)$ given by $S(f)(Z) = f^{-1}[Z]$.

References

- *Abstract and Concrete Categories. The Joy of Cats* [1]. Jiri Adámek, Horst Herrlich, George E. Strecker.
- Peter T. Johnstone, *Stone Spaces*. (1982) Cambridge university Press ISBN 0-521-23893-5

References

[1] <http://katmat.math.uni-bremen.de/acc/acc.pdf>

True quantified Boolean formula

In computational complexity theory, the language **TQBF** is a formal language consisting of the **true quantified Boolean formulas**. A (fully) quantified Boolean formula is a formula in quantified propositional logic where every variable is quantified (or bound), using either existential or universal quantifiers, at the beginning of the sentence. Such a formula is equivalent to either true or false (since there are no free variables). If such a formula evaluates to true, then that formula is in the language TQBF. It is also known as **QSAT** (Quantified SAT).

Overview

In computational complexity theory, the **quantified Boolean formula problem** (**QBF**) is a generalization of the Boolean satisfiability problem in which both existential quantifiers and universal quantifiers can be applied to each variable. Put another way, it asks whether a quantified sentential form over a set of Boolean variables is true or false. For example, the following is an instance of QBF:

$$\forall x \exists y \exists z ((x \vee z) \wedge y)$$

QBF is the canonical complete problem for PSPACE, the class of problems solvable by a deterministic or nondeterministic Turing machine in polynomial space and unlimited time. Given the formula in the form of an abstract syntax tree, the problem can be solved easily by a set of mutually recursive procedures which evaluate the formula. Such an algorithm uses space proportional to the height of the tree, which is linear in the worst case, but uses time exponential in the number of quantifiers.

Provided that MA ⊥ PSPACE, which is widely believed, QBF cannot be solved, nor can a given solution even be verified, in either deterministic or probabilistic polynomial time (in fact, unlike the satisfiability problem, there's no known way to specify a solution succinctly). It is trivial to solve using an alternating Turing machine in linear time, which is no surprise since in fact AP = PSPACE, where AP is the class of problems alternating machines can solve in polynomial time.

When the seminal result IP = PSPACE was shown (see interactive proof system), it was done by exhibiting an interactive proof system that could solve QBF by solving a particular arithmetization of the problem.

QBF formulas have a number of useful canonical forms. For example, it can be shown that there is a polynomial-time many-one reduction that will move all quantifiers to the front of the formula and make them alternate between universal and existential quantifiers. There is another reduction that proved useful in the IP =

PSPACE proof where no more than one universal quantifier is placed between each variable's use and the quantifier binding that variable. This was critical in limiting the number of products in certain subexpressions of the arithmetization.

Prenex normal form

A fully quantified Boolean formula can be assumed to have a very specific form, called prenex normal form. It has two basic parts: a portion containing only quantifiers and a portion containing an unquantified Boolean formula usually denoted as ϕ . If there are n Boolean variables, the entire formula can be written as

$$\exists x_1 \forall x_2 \exists x_3 \cdots Q_n x_n \phi(x_1, x_2, x_3, \dots, x_n)$$

where every variable falls within the scope of some quantifier. By introducing dummy variables, any formula in prenex normal form can be converted into a sentence where existential and universal quantifiers alternate. Using the dummy variable y_1 ,

$$\exists x_1 \exists x_2 \phi(x_1, x_2) \mapsto \exists x_1 \forall y_1 \exists x_2 \phi(x_1, x_2)$$

The second sentence has the same truth value but follows the restricted syntax. Assuming fully quantified Boolean formulas to be in prenex normal form is a frequent feature of proofs.

Solving

There is a simple recursive algorithm for determining whether a TQBF is true. Given some QBF

$$Q_1 x_1 Q_2 x_2 \cdots Q_n x_n \phi(x_1, x_2, \dots, x_n).$$

If the formula contains no quantifiers, we can just return the formula. Otherwise, we take off the first quantifier and check both possible values for the first variable:

$$A = Q_2 x_2 \cdots Q_n x_n \phi(0, x_2, \dots, x_n),$$

$$B = Q_2 x_2 \cdots Q_n x_n \phi(1, x_2, \dots, x_n).$$

If $Q_1 = \exists$, then return $A \vee B$. If $Q_1 = \forall$, then return $A \wedge B$.

How fast does this algorithm run? For every quantifier in the initial QBF, the algorithm makes two recursive calls on only a linearly smaller subproblem. This gives the algorithm an exponential runtime $O(2^n)$.

How much space does this algorithm use? Within each invocation of the algorithm, it needs to store the intermediate results of computing A and B. Every recursive call takes off one quantifier, so the total recursive depth is linear in the number of quantifiers. Formulas that lack quantifiers can be evaluated in space logarithmic in the number of variables. The initial QBF was fully quantified, so there are at least as many quantifiers as variables. Thus, this algorithm uses $O(n + \log n) = O(n)$ space. This makes the TQBF language part of the PSPACE complexity class.

PSPACE-completeness

The TQBF language serves in complexity theory as the canonical PSPACE-complete problem. Being PSPACE-complete means that a language is in PSPACE and that the language is also PSPACE-hard. The algorithm above shows that TQBF is in PSPACE. Showing that TQBF is PSPACE-hard requires showing that any language in the complexity class PSPACE can be reduced to TQBF in polynomial time. I.e.,

$$\forall L \in \text{PSPACE}, L \leq_p \text{TQBF}.$$

This means that, for a PSPACE language L, whether an input x is in L can be decided by checking whether $f(x)$ is in TQBF, for some function f that is required to run in polynomial time (relative to the length of the input). Symbolically,

$$x \in L \iff f(x) \in \text{TQBF}.$$

Proving that TQBF is PSPACE-hard, requires specification of f .

So, suppose that L is a PSPACE language. This means that L can be decided by a polynomial space deterministic Turing machine (DTM). This is very important for the reduction of L to TQBF, because the configurations of any such Turing Machine can be represented as Boolean formulas, with Boolean variables representing the state of the machine as well as the contents of each cell on the Turing Machine tape, with the position of the Turing Machine head encoded in the formula by the formula's ordering. In particular, our reduction will use the variables c_1 and c_2 , which represent two possible configurations of the DTM for L , and a natural number t , in constructing a QBF $\phi_{c_1, c_2, t}$ which is true if and only if the DTM for L can go from the configuration encoded in c_1 to the configuration encoded in c_2 in no more than t steps. The function f , then, will construct from the DTM for L a QBF $\phi_{c_{start}, c_{accept}, T}$, where c_{start} is the DTM's starting configuration, c_{accept} is the DTM's accepting configuration, and T is the maximum number of steps the DTM could need to move from one configuration to the other. We know that $T = O(\exp(n))$, where n is the length of the input, because this bounds the total number of possible configurations of the relevant DTM. Of course, it cannot take the DTM more steps than there are possible configurations to reach c_{accept} unless it enters a loop, in which case it will never reach c_{accept} anyway. At this stage of the proof, we have already reduced the question of whether an input formula w (encoded, of course, in c_{start}) is in L to the question of whether the QBF $\phi_{c_{start}, c_{accept}, T}$, i.e., $f(w)$, is in TQBF. The remainder of this proof proves that f can be computed in polynomial time.

For $t = 1$, computation of $\phi_{c_1, c_2, t}$ is straightforward—either one of the configurations changes to the other in one step or it does not. Since the Turing Machine that our formula represents is deterministic, this presents no problem. For $t > 1$, computation of $\phi_{c_1, c_2, t}$ involves a recursive evaluation, looking for a so-called "middle point" m_1 . In this case, we rewrite the formula as follows:

$$\phi_{c_1, c_2, t} = \exists m_1 (\phi_{c_1, m_1, \lceil t/2 \rceil} \wedge \phi_{m_1, c_2, \lceil t/2 \rceil}).$$

This converts the question of whether c_1 can reach c_2 in t steps to the question of whether c_1 reaches a middle point m_1 in $t/2$ steps, which itself reaches c_2 in $t/2$ steps. The answer to the latter question of course gives the answer to the former.

Now, t is only bounded by T , which is exponential (and so not polynomial) in the length of the input. Additionally, each recursive layer virtually doubles the length of the formula. (The variable m_1 is only one midpoint—for greater t , there are more stops along the way, so to speak.) So the time required to recursively evaluate $\phi_{c_1, c_2, t}$ in this manner could be exponential as well, simply because the formula could become exponentially large. This problem is solved by universally quantifying using variables c_3 and c_4 over the configuration pairs (e.g., $\{(c_1, m_1), (m_1, c_2)\}$), which prevents the length of the formula from expanding due to recursive layers. This yields the following interpretation of $\phi_{c_1, c_2, t}$:

$$\phi_{c_1, c_2, t} = \exists m_1 \forall (c_3, c_4) \in \{(c_1, m_1), (m_1, c_2)\} (\phi_{c_3, c_4, \lceil t/2 \rceil}).$$

This version of the formula can indeed be computed in polynomial time, since any one instance of it can be computed in polynomial time. The universally quantified ordered pair simply tells us that whichever choice of (c_3, c_4) is made, $\phi_{c_1, c_2, t} \iff \phi_{c_3, c_4, \lceil t/2 \rceil}$.

Thus, $\forall L \in \text{PSPACE}, L \leq_p \text{TQBF}$, so TQBF is PSPACE-hard. Together with the above result that TQBF is in PSPACE, this completes the proof that TQBF is a PSPACE-complete language.

(This proof follows Sipser 2006 pp. 310–313 in all essentials. Papadimitriou 1994 also includes a proof.)

Miscellany

- One important subproblem in TQBF is the Boolean satisfiability problem. In this problem, you wish to know whether a given Boolean formula ϕ can be made true with some assignment of variables. This is equivalent to the TQBF using only existential quantifiers:

$$\exists x_1 \cdots \exists x_n \phi(x_1, \dots, x_n)$$

This is also an example of the larger result $\text{NP} \subseteq \text{PSPACE}$ which follows directly from the observation that a polynomial time verifier for a proof of a language accepted by a NTM (Non-deterministic Turing machine) requires polynomial space to store the proof.

- Any class in the polynomial hierarchy (PH) has TQBF as a hard problem. In other words, for the class comprising all languages L for which there exists a poly-time TM V, a verifier, such that for all input x and some constant i,

$$x \in L \Leftrightarrow \exists y_1 \forall y_2 \cdots Q_i y_i V(x, y_1, y_2, \dots, y_i) = 1$$

which has a specific QBF formulation that is given as

$$\exists \phi \text{ such that } \exists \vec{x}_1 \forall \vec{x}_2 \cdots Q_i \vec{x}_i \phi(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_i) = 1$$

where the \vec{x}_i 's are vectors of Boolean variables.

- It is important to note that while TQBF the language is defined as the collection of true quantified Boolean formulas, the abbreviation TQBF is often used (even in this article) to stand for a totally quantified Boolean formula, often simply called a QBF (quantified Boolean formula, understood as "fully" or "totally" quantified). It is important to distinguish contextually between the two uses of the abbreviation TQBF in reading the literature.
- A TQBF can be thought of as a game played between two players, with alternating moves. Existentially quantified variables are equivalent to the notion that a move is available to a player at a turn. Universally quantified variables mean that the outcome of the game does not depend on what move a player makes at that turn. Also, a TQBF whose first quantifier is existential corresponds to a formula game in which the first player has a winning strategy.
- A TQBF for which the quantified formula is in 2-CNF may be solved in linear time, by an algorithm involving strong connectivity analysis of its implication graph. The 2-satisfiability problem is a special case of TQBF for these formulas, in which every quantifier is existential.
- There is a systematic treatment of restricted versions of quantified boolean formulas (giving Schaefer-type classifications) provided in an expository paper by Hubie Chen.

Notes and references

- Fortnow & Homer (2003) provides some historical background for PSPACE and TQBF.
- Zhang (2003) provides some historical background of Boolean formulas.
- Arora, Sanjeev. (2001). *COS 522: Computational Complexity* (<http://www.cs.princeton.edu/~arora/pubs/aroracom.ps>). Lecture Notes, Princeton University. Retrieved October 10, 2005.
- Fortnow, Lance & Steve Homer. (2003, June). A short history of computational complexity (<http://people.cs.uchicago.edu/~fortnow/beatcs/column80.pdf>). *The Computational Complexity Column*, 80. Retrieved October 9, 2005.
- Papadimitriou, C. H. (1994). *Computational Complexity*. Reading: Addison-Wesley.
- Sipser, Michael. (2006). *Introduction to the Theory of Computation*. Boston: Thomson Course Technology.
- Zhang, Lintao. (2003). *Searching for truth: Techniques for satisfiability of boolean formulas* (http://research.microsoft.com/users/lintaoz/thesis_lintao_zhang.pdf). Retrieved October 10, 2005.

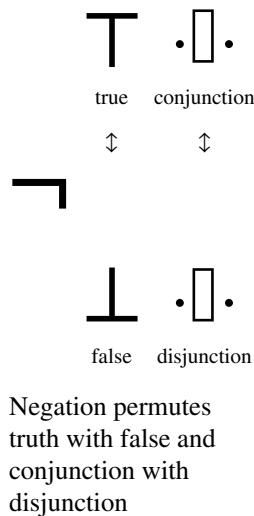
External links

- The Quantified Boolean Formulas Library (QBFLIB) (<http://www.qbflib.org>)

Truth value

In logic and mathematics, a **truth value**, sometimes called a **logical value**, is a value indicating the relation of a proposition to truth.

Classical logic



Negation permutes
truth with false and
conjunction with
disjunction

In classical logic, with its intended semantics, the truth values are **true** (1 or T) and **false** (0 or F); that is, classical logic is a two-valued logic. This set of two values is also called the Boolean domain. Corresponding semantics of logical connectives are truth functions, whose values are expressed in the form of truth tables. Logical biconditional becomes the equality binary relation, and negation becomes a bijection which permutes true and false. Conjunction and disjunction are dual with respect to negation, which is expressed by De Morgan's laws:

$$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$$

$$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$$

Propositional variables become variables in the Boolean domain. Assigning values for propositional variables is referred to as valuation.

Multi-valued logic

Multi-valued logics (such as fuzzy logic and relevance logic) allow for more than two truth values, possibly containing some internal structure. For example, on the unit interval [0,1] such structure is a total order; this may be expressed as existence of various degrees of truth.

Algebraic semantics

Not all logical systems are truth-valuational in the sense that logical connectives may be interpreted as truth functions. For example, intuitionistic logic lacks a complete set of truth values because its semantics, the Brouwer–Heyting–Kolmogorov interpretation, is specified in terms of provability conditions, and not directly in terms of the necessary truth of formulae.

But even non-truth-valuational logics can associate values with logical formulae, as is done in algebraic semantics. The algebraic semantics of intuitionistic logic is given in terms of Heyting algebras, compared to Boolean algebra semantics of classical propositional calculus.

In other theories

Intuitionistic type theory uses types in the place of truth values.

Topos theory uses truth values in a special sense: the truth values of a topos are the global elements of the subobject classifier. Having truth values in this sense does not make a logic truth valuational.

External links

- Truth Values^[1] entry by Yaroslav Shramko, Heinrich Wansing in the *Stanford Encyclopedia of Philosophy*

Vector logic

Vector logic^{[1][2]} is an algebraic model of elementary logic based on matrix algebra. Vector logic assumes that the truth values map on vectors, and that the monadic and dyadic operations are executed by matrix operators.

Overview

Classic binary logic is represented by a small set of mathematical functions depending on one (monadic) or two (dyadic) variables. In the binary set, the value 1 corresponds to *true* and the value 0 to *false*. A two-valued vector logic requires a correspondence between the truth-values *true* (t) and *false* (f), and two q -dimensional normalized column vectors composed by real numbers s and n , hence:

$$t \mapsto s \quad \text{and} \quad f \mapsto n$$

(where $q \geq 2$ is an arbitrary natural number, and “normalized” means that the length of the vector is 1; usually s and n are orthogonal vectors). This correspondence generates a space of vector truth-values: $V_2 = \{s, n\}$. The basic logical operations defined using this set of vectors lead to matrix operators.

The operations of vector logic are based on the scalar product between q -dimensional column vectors: $u^T v = \langle u, v \rangle$: the orthonormality between vectors s and n implies that $\langle u, v \rangle = 1$ if $u = v$, and $\langle u, v \rangle = 0$ if $u \neq v$.

Monadic operators

The monadic operators result from the application $Mon : V_2 \rightarrow V_2$, and the associated matrices have q rows and q columns. The two basic monadic operators for this two-valued vector logic are the identity and the negation:

- **Identity:** A logical identity $ID(p)$ is represented by matrix $I = ss^T + nn^T$. This matrix operates as follows:
 $Ip = p$, $p \in V_2$; due to the orthogonality of s respect to n , we have
 $Is = ss^T s + nn^T s = s\langle s, s \rangle + n\langle n, s \rangle = s$, and conversely $In = n$.
- **Negation:** A logical negation $\neg p$ is represented by matrix $N = ns^T + sn^T$. Consequently, $Ns = n$ and $Nn = s$.
The involutory behavior of the logical negation, namely that $\neg(\neg p)$ equals p , corresponds with the fact that $N^2 = I$. Is important to note that this vector logic identity matrix is not generally an identity matrix in the sense of matrix algebra.

Dyadic operators

The 16 two-valued dyadic operators correspond to functions of the type $Dyad : V_2 \otimes V_2 \rightarrow V_2$; the dyadic matrices have q rows and q^2 columns. The matrices that execute these dyadic operations are based on the properties of the Kronecker product.

Two properties of this product are essential for the formalism of vector logic:

1. **The mixed-product property** If \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{D} are matrices of such size that one can form the matrix products \mathbf{AC} and \mathbf{BD} , then

$$(A \otimes B)(C \otimes D) = AC \otimes BD$$

2. **Distributive transpose** The operation of transposition is distributive over the Kronecker product:

$$(A \otimes B)^T = A^T \otimes B^T.$$

Using these properties, expressions for dyadic logic functions can be obtained:

- **Conjunction.** The conjunction ($p \wedge q$) is executed by a matrix that acts on two vector truth-values: $C(u \otimes v)$. This matrix reproduces the features of the classical conjunction truth-table in its formulation:

$$C = s(s \otimes s)^T + n(s \otimes n)^T + n(n \otimes s)^T + n(n \otimes n)^T$$

and verifies

$$C(s \otimes s) = s, \text{ and}$$

$$C(s \otimes n) = C(n \otimes s) = C(n \otimes n) = n.$$

- **Disjunction.** The disjunction ($p \vee q$) is executed by the matrix

$$D = s(s \otimes s)^T + s(s \otimes n)^T + s(n \otimes s)^T + n(n \otimes n)^T, \text{ resulting in}$$

$$D(s \otimes s) = D(s \otimes n) = D(n \otimes s) = s \text{ and}$$

$$D(n \otimes n) = n.$$

- **Implication.** The implication corresponds in classical logic to the expression $p \rightarrow q \equiv \neg p \vee q$. The vector logic version of this equivalence leads to a matrix that represents this implication in vector logic: $L = D(N \otimes I)$.

The explicit expression for this implication is:

$$L = s(s \otimes s)^T + n(s \otimes n)^T + s(n \otimes s)^T + n(n \otimes n)^T,$$

and the properties of classical implication are satisfied:

$$L(s \otimes s) = L(n \otimes s) = L(n \otimes n) = s \text{ and}$$

$$L(s \otimes n) = n.$$

- **Equivalence and Exclusive or.** In vector logic the equivalence $p \equiv q$ is represented by the following matrix:

$$E = s(s \otimes s)^T + n(s \otimes n)^T + n(n \otimes s)^T + s(n \otimes n)^T \text{ with}$$

$$E(s \otimes s) = E(n \otimes n) = s \text{ and}$$

$$E(s \otimes n) = E(n \otimes s) = n.$$

The Exclusive or is the negation of the equivalence, $\neg(p \equiv q)$; it corresponds with the matrix $X = NE$ given by

$$X = n(s \otimes s)^T + s(s \otimes n)^T + s(n \otimes s)^T + n(n \otimes n)^T,$$

$$\text{with } X(s \otimes s) = X(n \otimes n) = n \text{ and}$$

$$X(s \otimes n) = X(n \otimes s) = s.$$

- **NAND and NOR**

The matrices S and P correspond to the Sheffer (NAND) and the Peirce (NOR) operations, respectively:

$$S = NC$$

$$P = ND$$

De Morgan's law

In the two-valued logic, the conjunction and the disjunction operations satisfy the De Morgan's law: $p \wedge q \equiv \neg(\neg p \vee \neg q)$, and its dual: $p \vee q \equiv \neg(\neg p \wedge \neg q)$. For the two-valued vector logic this Law is also verified:

$$C(u \otimes v) = ND(Nu \otimes Nv), \text{ where } u \text{ and } v \text{ are two logic vectors.}$$

The Kronecker product implies the following factorization:

$$C(u \otimes v) = ND(N \otimes N)(u \otimes v).$$

Then it can be proved that in the two-dimensional vector logic the De Morgan's law is a law involving operators, and not only a law concerning operations:^[3]

$$C = ND(N \otimes N)$$

Law of contraposition

In the classical propositional calculus, the Law of Contraposition $p \rightarrow q \equiv \neg q \rightarrow \neg p$ is proved because the equivalence holds for all the possible combinations of truth-values of p and q .^[4] Instead, in vector logic, the law of contraposition emerges from a chain of equalities within the rules of matrix algebra and Kronecker products, as shown in what follows:

$$\begin{aligned} L(u \otimes v) &= D(N \otimes I)(u \otimes v) = D(Nu \otimes v) = D(Nu \otimes NNv) = \\ &D(NNv \otimes Nu) = D(N \otimes I)(Nv \otimes Nu) = L(Nv \otimes Nu) \end{aligned}$$

This result is based in the fact that D , the disjunction matrix, represents a commutative operation.

Many-valued two-dimensional logic

Many-valued logic was developed by many researchers, particularly by Jan Łukasiewicz and allows extending logical operations to truth-values that include uncertainties.^[5] In the case of two-valued vector logic, uncertainties in the truth values can be introduced using vectors with s and n weighted by probabilities.

Let $f = \epsilon s + \delta n$, with $\epsilon, \delta \in [0, 1]$, $\epsilon + \delta = 1$ be this kind of "probabilistic" vectors. Here, the many-valued character of the logic is introduced *a posteriori* via the uncertainties introduced in the inputs.

Scalar projections of vector outputs

The outputs of this many-valued logic can be projected on scalar functions and generate a particular class of probabilistic logic with similarities with the many-valued logic of Reichenbach.^{[6][7][8]} Given two vectors $u = \alpha s + \beta n$ and $v = \alpha' s + \beta' n$ and a dyadic logical matrix G , a scalar probabilistic logic is provided by the projection over vector s :

$$Val(\text{scalars}) = s^T G(\text{vectors})$$

Here are the main results of these projections:

$$\begin{aligned} NOT(\alpha) &= s^T Nu = 1 - \alpha \\ OR(\alpha, \alpha') &= s^T D(u \otimes v) = \alpha + \alpha' - \alpha\alpha' \\ AND(\alpha, \alpha') &= s^T C(u \otimes v) = \alpha\alpha' \\ IMPL(\alpha, \alpha') &= s^T L(u \otimes v) = 1 - \alpha(1 - \alpha') \\ XOR(\alpha, \alpha') &= s^T X(u \otimes v) = \alpha + \alpha' - 2\alpha\alpha' \end{aligned}$$

The associated negations are:

$$\begin{aligned} NOR(\alpha, \alpha') &= 1 - OR(\alpha, \alpha') \\ NAND(\alpha, \alpha') &= 1 - AND(\alpha, \alpha') \\ EQUI(\alpha, \alpha') &= 1 - XOR(\alpha, \alpha') \end{aligned}$$

If the scalar values belong to the set $\{0, \frac{1}{2}, 1\}$, this many-valued scalar logic is for many of the operators almost identical to the 3-valued logic of Łukasiewicz. Also, it has been proved that when the monadic or dyadic operators act over probabilistic vectors belonging to this set, the output is also an element of this set.

History

The approach has been inspired in neural network models based on the use of high-dimensional matrices and vectors.^{[9][10]} Vector logic is a direct translation into a matrix-vector formalism of the classical Boolean polynomials.^[11] This kind of formalism has been applied to develop a fuzzy logic in terms of complex numbers.^[12] Other matrix and vector approaches to logical calculus have been developed in the framework of quantum physics, computer science and optics.^{[13][14][15]} Early attempts to use linear algebra to represent logic operations can be referred to Peirce and Copilowich.^[16] The Indian biophysicist G.N. Ramachandran developed a formalism using algebraic matrices and vectors to represent many operations of classical Indian logic.^[17]

Boolean polynomials

George Boole established the development of logical operations as polynomials. For the case of monadic operators (such as identity or negation), the Boolean polynomials look as follows:

$$f(x) = f(1)x + f(0)(1 - x)$$

The four different monadic operations result from the different binary values for the coefficients. Identity operation requires $f(1) = 1$ and $f(0) = 0$, and negation occurs if $f(1) = 0$ and $f(0) = 1$. For the 16 dyadic operators, the Boolean polynomials are of the form:

$$f(x, y) = f(1, 1)xy + f(1, 0)x(1 - y) + f(0, 1)(1 - x)y + f(0, 0)(1 - x)(1 - y)$$

The dyadic operations can be translated to this polynomial format when the coefficients f take the values indicated in the respective truth tables. For instance: the NAND operation requires that:

$$f(1, 1) = 0 \text{ and } f(1, 0) = f(0, 1) = f(0, 0) = 1.$$

These Boolean polynomials can be immediately extended to any number of variables, producing a large potential variety of logical operators. In vector logic, the matrix-vector structure of logical operators is an exact translation to the format of liner algebra of these Boolean polynomials, where the x and $1-x$ correspond to vectors s and n respectively (the same for y and $1-y$). In the example of NAND, $f(1,1)=n$ and $f(1,0)=f(0,1)=f(0,0)=s$ and the matrix version becomes:

$$S = n(s \otimes s)^T + s[(s \otimes n)^T + (n \otimes s)^T + (n \otimes n)^T]$$

Extensions

- Vector logic can be extended to include many truth values since large dimensional vector spaces allow to create many orthogonal truth values and the corresponding logical matrices.
- Logical modalities can be fully represented in this context, with recursive process inspired in neural models^[18]
- Some cognitive problems about logical computations can be analyzed using this formalism, in particular recursive decisions. Any logical expression of classical propositional calculus can be naturally represented by a tree structure. This fact is retained by vector logic, and has been partially used in neural models focused in the investigation of the branched structure of natural languages.^{[19][20][21][22][23][24]}
- The computation via reversible operations as the Fredkin gate can be implemented in vector logic. This implementations provides explicit expressions for matrix operators that produce the input format and the output filtering necessary for obtaining computations
- Elementary cellular automata can be analyzed using the operator structure of vector logic; this analysis leads to a spectral decomposition of the laws governing the its dynamics^{[25][26]}

References

- [1] Mizraji, E. (1992). Vector logics: the matrix-vector representation of logical calculus. (<http://www.sciencedirect.com/science/article/pii/016501149290216Q>) *Fuzzy Sets and Systems*, 50, 179–185, 1992
- [2] Mizraji, E. (2008) Vector logic: a natural algebraic representation of the fundamental logical gates. (<http://logcom.oxfordjournals.org/content/18/1/97.full.pdf>) *Journal of Logic and Computation*, 18, 97–121, 2008
- [3] Mizraji, E. (1996) The operators of vector logic. *Mathematical Logic Quarterly*, 42, 27–39
- [4] Suppes, P. (1957) *Introduction to Logic*, Van Nostrand Reinhold, New York.
- [5] Łukasiewicz, J. (1980) *Selected Works*. L. Borkowski, ed., pp. 153–178. North-Holland, Amsterdam, 1980
- [6] Rescher, N. (1969) *Many-Valued Logic*. McGraw–Hill, New York
- [7] Blanché, R. (1968) *Introduction à la Logique Contemporaine*, Armand Colin, Paris
- [8] Klir, G.J., Yuan, G. (1995) *Fuzzy Sets and Fuzzy Logic*. Prentice–Hall, New Jersey
- [9] Kohonen, T. (1977) *Associative Memory: A System-Theoretical Approach*. Springer-Verlag, New York
- [10] Mizraji, E. (1989) Context-dependent associations in linear distributed memories (<http://link.springer.com/article/10.1007/BF02458441>). *Bulletin of Mathematical Biology*, 50, 195–205
- [11] Boole, G. (1854) *An Investigation of the Laws of Thought, on which are Founded the Theories of Logic and Probabilities*. Macmillan, London, 1854; Dover, New York Reedition, 1958
- [12] Dick, S. (2005) Towards complex fuzzy logic. *IEEE Transactions on Fuzzy Systems*, 15, 405–414, 2005
- [13] Mittelstaedt, P. (1968) *Philosophische Probleme der Modernen Physik*, Bibliographisches Institut, Mannheim
- [14] Stern, A. (1988) *Matrix Logic: Theory and Applications*. North-Holland, Amsterdam
- [15] Westphal, J., Hardy, J. (2005) Logic as a vector system. *Journal of Logic and Computation*, 15, 751–765
- [16] Copilovich, I.M. (1948) Matrix development of the calculus of relations. *Journal of Symbolic Logic*, 13, 193–203
- [17] Jain, M.K. (2011) Logic of evidence-based inference propositions, *Current Science*, 1663–1672, 100
- [18] Mizraji, E. (1994) Modalities in vector logic (<http://projecteuclid.org/DPubS?verb=Display&version=1.0&service=UI&handle=euclid.ndjfl/1094061864&page=record>). *Notre Dame Journal of Formal Logic*, 35, 272–283
- [19] Mizraji, E., Lin, J. (2002) The dynamics of logical decisions. *Physica D*, 168–169, 386–396
- [20] beim Graben, P., Potthast, R. (2009). Inverse problems in dynamic cognitive modeling. *Chaos*, 19, 015103
- [21] beim Graben, P., Pinotsis, D., Saddy, D., Potthast, R. (2008). Language processing with dynamic fields. *Cogn. Neurodyn.*, 2, 79–88
- [22] beim Graben, P., Gerth, S., Vasishth, S. (2008) Towards dynamical system models of language-related brain potentials. *Cogn. Neurodyn.*, 2, 229–255
- [23] beim Graben, P., Gerth, S. (2012) Geometric representations for minimalist grammars. *Journal of Logic, Language and Information*, 21, 393–432 .
- [24] Binazzi, A. (2012) Cognizione logica e modelli mentali. (<http://www.fupress.net/index.php/sf/article/view/11649>) *Studi sulla formazione*, 1–2012, pag. 69–84
- [25] Mizraji, E. (2006) The parts and the whole: inquiring how the interaction of simple subsystems generates complexity. *International Journal of General Systems*, 35, pp. 395–415.
- [26] Arruti, C., Mizraji, E. (2006) Hidden potentialities. *International Journal of General Systems*, 35, 461–469.

Visualization

Binary decision diagram

In the field of computer science, a **binary decision diagram (BDD)** or **branching program**, like a negation normal form (NNF) or a propositional directed acyclic graph (PDAG), is a data structure that is used to represent a Boolean function. On a more abstract level, BDDs can be considered as a compressed representation of sets or relations. Unlike other compressed representations, operations are performed directly on the compressed representation, i.e. without decompression.

Definition

A Boolean function can be represented as a rooted, directed, acyclic graph, which consists of several decision nodes and terminal nodes. There are two types of terminal nodes called 0-terminal and 1-terminal. Each decision node N is labeled by Boolean variable V_N and has two child nodes called low child and high child. The edge from node V_N to a low (or high) child represents an assignment of V_N to 0 (resp. 1). Such a **BDD** is called 'ordered' if different variables appear in the same order on all paths from the root. A BDD is said to be 'reduced' if the following two rules have been applied to its graph:

- Merge any isomorphic subgraphs.
- Eliminate any node whose two children are isomorphic.

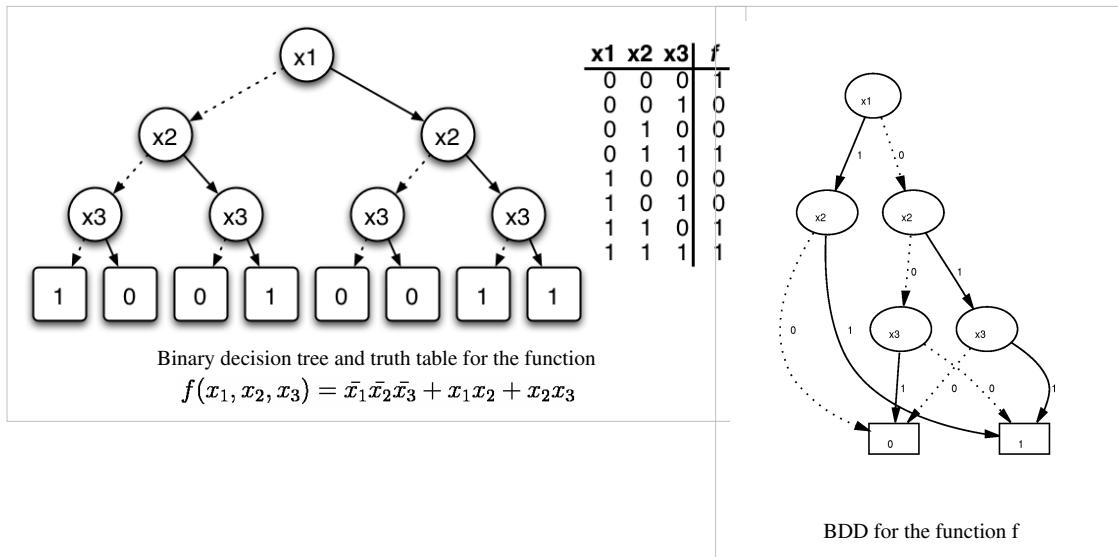
In popular usage, the term **BDD** almost always refers to **Reduced Ordered Binary Decision Diagram (ROBDD)** in the literature, used when the ordering and reduction aspects need to be emphasized). The advantage of an ROBDD is that it is canonical (unique) for a particular function and variable order.^[1] This property makes it useful in functional equivalence checking and other operations like functional technology mapping.

A path from the root node to the 1-terminal represents a (possibly partial) variable assignment for which the represented Boolean function is true. As the path descends to a low (or high) child from a node, then that node's variable is assigned to 0 (resp. 1).

Example

The left figure below shows a binary decision *tree* (the reduction rules are not applied), and a truth table, each representing the function $f(x_1, x_2, x_3)$. In the tree on the left, the value of the function can be determined for a given variable assignment by following a path down the graph to a terminal. In the figures below, dotted lines represent edges to a low child, while solid lines represent edges to a high child. Therefore, to find $(x_1=0, x_2=1, x_3=1)$, begin at x_1 , traverse down the dotted line to x_2 (since x_1 has an assignment to 0), then down two solid lines (since x_2 and x_3 each have an assignment to one). This leads to the terminal 1, which is the value of $f(x_1=0, x_2=1, x_3=1)$.

The binary decision *tree* of the left figure can be transformed into a binary decision *diagram* by maximally reducing it according to the two reduction rules. The resulting **BDD** is shown in the right figure.



History

The basic idea from which the data structure was created is the Shannon expansion. A switching function is split into two sub-functions (cofactors) by assigning one variable (cf. *if-then-else normal form*). If such a sub-function is considered as a sub-tree, it can be represented by a *binary decision tree*. Binary decision diagrams (BDD) were introduced by Lee,^[2] and further studied and made known by Akers^[3] and Boute.^[4]

The full potential for efficient algorithms based on the data structure was investigated by Randal Bryant at Carnegie Mellon University: his key extensions were to use a fixed variable ordering (for canonical representation) and shared sub-graphs (for compression). Applying these two concepts results in an efficient data structure and algorithms for the representation of sets and relations.^{[5][6]} By extending the sharing to several BDDs, i.e. one sub-graph is used by several BDDs, the data structure *Shared Reduced Ordered Binary Decision Diagram* is defined.^[7] The notion of a BDD is now generally used to refer to that particular data structure.

In his video lecture *Fun With Binary Decision Diagrams (BDDs)*, Donald Knuth calls BDDs "one of the only really fundamental data structures that came out in the last twenty-five years" and mentions that Bryant's 1986 paper was for some time one of the most-cited papers in computer science.

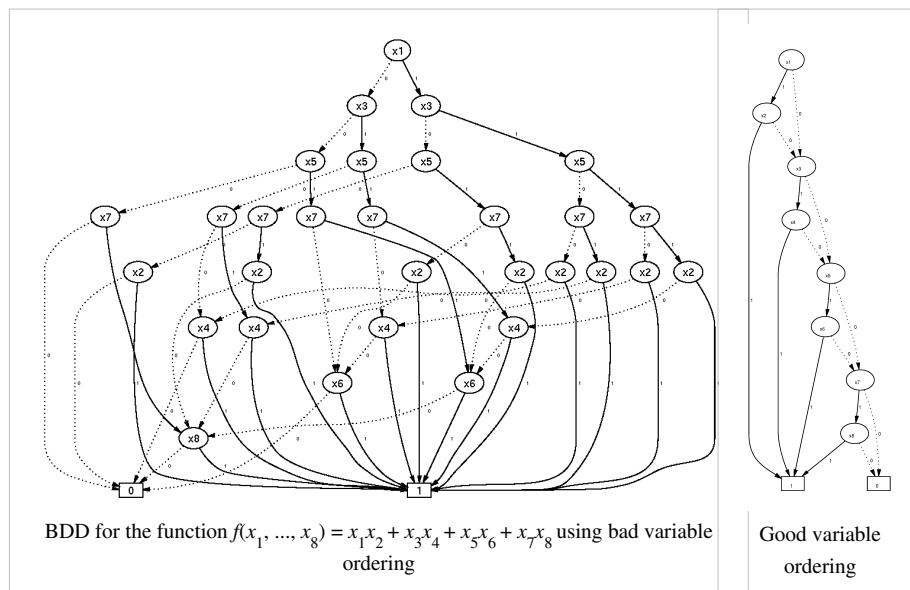
Applications

BDDs are extensively used in CAD software to synthesize circuits (logic synthesis) and in formal verification. There are several lesser known applications of BDD, including Fault tree analysis, Bayesian Reasoning, Product Configuration, and Private information retrieval^{[8][9][citation needed]}.

Every arbitrary BDD (even if it is not reduced or ordered) can be directly implemented by replacing each node with a 2 to 1 multiplexer; each multiplexer can be directly implemented by a 4-LUT in a FPGA. It is not so simple to convert from an arbitrary network of logic gates to a BDD^[citation needed] (unlike the and-inverter graph).

Variable ordering

The size of the BDD is determined both by the function being represented and the chosen ordering of the variables. There exist Boolean functions $f(x_1, \dots, x_n)$ for which depending upon the ordering of the variables we would end up getting a graph whose number of nodes would be linear (in n) at the best and exponential at the worst case (e.g., a ripple carry adder). Let us consider the Boolean function $f(x_1, \dots, x_{2n}) = x_1x_2 + x_3x_4 + \dots + x_{2n-1}x_{2n}$. Using the variable ordering $x_1 < x_3 < \dots < x_{2n-1} < x_2 < x_4 < \dots < x_{2n}$, the BDD needs 2^{n+1} nodes to represent the function. Using the ordering $x_1 < x_2 < x_3 < x_4 < \dots < x_{2n-1} < x_{2n}$, the BDD consists of $2n+2$ nodes.



It is of crucial importance to care about variable ordering when applying this data structure in practice. The problem of finding the best variable ordering is NP-hard.^[10] For any constant $c > 1$ it is even NP-hard to compute a variable ordering resulting in an OBDD with a size that is at most c times larger than an optimal one.^[11] However there exist efficient heuristics to tackle the problem.^[citation needed]

There are functions for which the graph size is always exponential — independent of variable ordering. This holds e.g. for the multiplication function (an indication^[citation needed] as to the apparent complexity of factorization).

Researchers have of late suggested refinements on the BDD data structure giving way to a number of related graphs, such as BMD (Binary Moment Diagrams), ZDD (Zero Suppressed Decision Diagram), FDD (Free Binary Decision Diagrams), PDD (Parity decision Diagrams), and MTBDDs (Multiple terminal BDDs).

Logical operations on BDDs

Many logical operations on BDDs can be implemented by polynomial-time graph manipulation algorithms.

- conjunction
- disjunction
- negation
- existential abstraction
- universal abstraction

However, repeating these operations several times, for example forming the conjunction or disjunction of a set of BDDs, may in the worst case result in an exponentially big BDD. This is because any of the preceding operations for two BDDs may result in a BDD with a size proportional to the product of the BDDs' sizes, and consequently for

several BDDs the size may be exponential.

References

- [1] Graph-Based Algorithms for Boolean Function Manipulation, Randal E. Bryant, 1986
- [2] C. Y. Lee. "Representation of Switching Circuits by Binary-Decission Programs". Bell Systems Technical Journal, 38:985–999, 1959.
- [3] Sheldon B. Akers. Binary Decision Diagrams, IEEE Transactions on Computers, C-27(6):509–516, June 1978.
- [4] Raymond T. Boute, "The Binary Decision Machine as a programmable controller". EUROMICRO Newsletter, Vol. 1(2):16–22, January 1976.
- [5] Randal E. Bryant. " Graph-Based Algorithms for Boolean Function Manipulation (<http://www.cs.cmu.edu/~bryant/pubdir/ieetc86.ps>)". IEEE Transactions on Computers, C-35(8):677–691, 1986.
- [6] R. E. Bryant, " Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams" (<http://www.cs.cmu.edu/~bryant/pubdir/acmcs92.ps>), ACM Computing Surveys, Vol. 24, No. 3 (September, 1992), pp. 293–318.
- [7] Karl S. Brace, Richard L. Rudell and Randal E. Bryant. " Efficient Implementation of a BDD Package" (<http://portal.acm.org/citation.cfm?id=123222&coll=portal&dl=ACM>). In Proceedings of the 27th ACM/IEEE Design Automation Conference (DAC 1990), pages 40–45. IEEE Computer Society Press, 1990.
- [8] R.M. Jensen. "CLab: A C+ + library for fast backtrack-free interactive product configuration" (<http://www.cs.cmu.edu/~runej/data/papers/JSW04.pdf>). Proceedings of the Tenth International Conference on Principles and Practice of Constraint Programming, 2004.
- [9] H.L. Lipmaa. "First CPIR Protocol with Data-Dependent Computation" (<http://eprint.iacr.org/2009/395.pdf>). ICISC 2009.
- [10] Beate Bollig, Ingo Wegener, IEEE Transactions on Computers, 45(9):993–1002, September 1996.
- [11] Detlef Sieling. "The nonapproximability of OBDD minimization." Information and Computation 172, 103–138. 2002.
- R. Ubar, "Test Generation for Digital Circuits Using Alternative Graphs (in Russian)", in Proc. Tallinn Technical University, 1976, No.409, Tallinn Technical University, Tallinn, Estonia, pp. 75–81.

Further reading

- D. E. Knuth, "The Art of Computer Programming Volume 4, Fascicle 1: Bitwise tricks & techniques; Binary Decision Diagrams" (Addison–Wesley Professional, March 27, 2009) viii+260pp, ISBN 0-321-58050-8. Draft of Fascicle 1b (<http://www-cs-faculty.stanford.edu/~knuth/fasc1b.ps.gz>) available for download.
- H. R. Andersen " An Introduction to Binary Decision Diagrams, (http://configit.com/configit_wordpress/wp-content/uploads/2013/07/bdd-eap.pdf)" Lecture Notes, 1999, IT University of Copenhagen.
- Ch. Meinel, T. Theobald, " Algorithms and Data Structures in VLSI-Design: OBDD – Foundations and Applications" (http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_ITS/books/OBDD-Book.pdf), Springer-Verlag, Berlin, Heidelberg, New York, 1998. Complete textbook available for download.
- Rüdiger Ebendt; Görschwin Fey; Rolf Drechsler (2005). *Advanced BDD optimization*. Springer. ISBN 978-0-387-25453-1.
- Bernd Becker; Rolf Drechsler (1998). *Binary Decision Diagrams: Theory and Implementation*. Springer. ISBN 978-1-4419-5047-5.

External links

Available OBDD Packages

- ABCD (<http://fmv.jku.at/abcd/>): The ABCD package by Armin Biere, Johannes Kepler Universität, Linz.
- CMU BDD (<http://www-2.cs.cmu.edu/~modelcheck/bdd.html>), BDD package, Carnegie Mellon University, Pittsburgh
- BuDDy (<http://buddy.sourceforge.net/manual/>): A BDD package by Jørn Lind-Nielsen
- Biddy (<http://lms.uni-mb.si/biddy/>): Academic multiplatform BDD package, University of Maribor
- CUDD (<http://vlsi.colorado.edu/~fabio/CUDD/>): BDD package, University of Colorado, Boulder
- JavaBDD (<http://javabdd.sourceforge.net>), a Java port of BuDDy that also interfaces to CUDD, CAL, and JDD
- JDD (<http://javaddlib.sourceforge.net/jdd/>) is a pure java implementation of BDD and ZBDD. JBDD (<http://javaddlib.sourceforge.net/jbdd/>) by the same author has a similar API but is a Java interface to BuDDy and CUDD

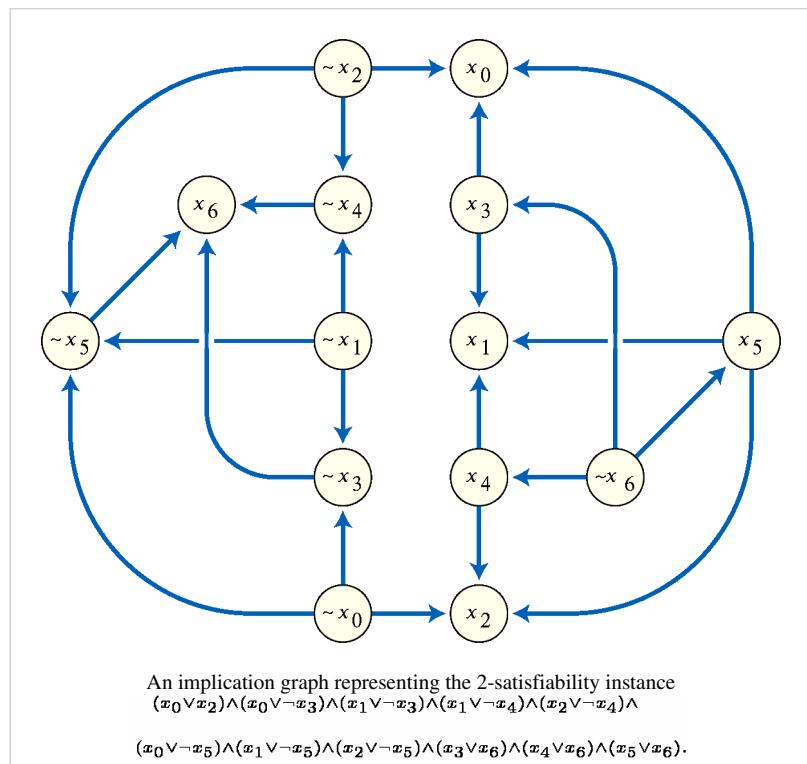
- The Berkeley CAL (http://embedded.eecs.berkeley.edu/Research/cal_bdd/) package which does breadth-first manipulation
- DDD (<http://ddd.lip6.fr>): A C++ library with support for integer valued and hierarchical decision diagrams.
- JINC (<http://www.jossowski.de/projects/jinc/jinc.html>): A C++ library developed at University of Bonn, Germany, supporting several BDD variants and multi-threading.
- Fun With Binary Decision Diagrams (BDDs) (<http://myvideos.stanford.edu/player/slplayer.aspx?coll=ea60314a-53b3-4be2-8552-dcf190ca0c0b&co=18bcd3a8-965a-4a63-a516-a1ad74af1119&o=true>), lecture by Donald Knuth

Implication graph

In mathematical logic, an **implication graph** is a skew-symmetric directed graph $G(V, E)$ composed of vertex set V and directed edge set E . Each vertex in V represents the truth status of a Boolean literal, and each directed edge from vertex u to vertex v represents the material implication "If the literal u is true then the literal v is also true". Implication graphs were originally used for analyzing complex Boolean expressions.

Applications

A 2-satisfiability instance in conjunctive normal form can be transformed into an implication graph by replacing each of its disjunctions by a pair of implications. An instance is satisfiable if and only if no literal and its negation belong to the same strongly connected component of its implication graph; this characterization can be used to solve 2-satisfiability instances in linear time.

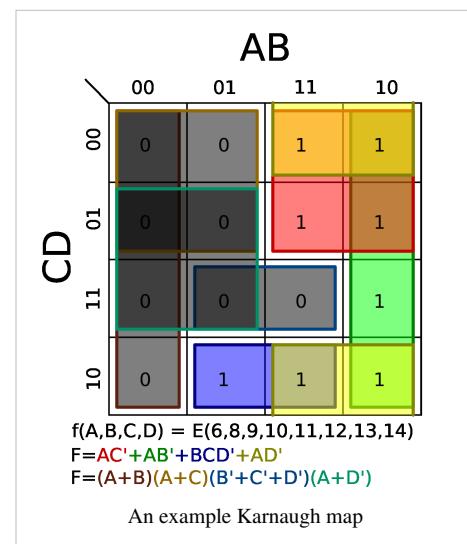


References

Karnaugh map

The **Karnaugh map**, also known as the **K-map**, is a method to simplify boolean algebra expressions. Maurice Karnaugh introduced it in 1953 as a refinement of Edward Veitch's 1952 **Veitch diagram**. The Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability. It also permits the rapid identification and elimination of potential race conditions.

The required boolean results are transferred from a truth table onto a two-dimensional grid where the cells are ordered in Gray code, and each cell position represents one combination of input conditions, while each cell value represents the corresponding output value. Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table. These terms can be used to write a minimal boolean expression representing the required logic.



Karnaugh maps are used to simplify real-world logic requirements so that they can be implemented using a minimum number of physical logic gates. A sum-of-products expression can always be implemented using AND gates feeding into an OR gate, and a product-of-sums expression leads to OR gates feeding an AND gate. Karnaugh maps can also be used to simplify logic expressions in software design. Boolean conditions, as used for example in conditional statements, can get very complicated, which makes the code difficult to read and to maintain. Once minimised, canonical sum-of-products and product-of-sums expressions can be implemented directly using AND and OR logic operators.

Example

Karnaugh maps are used to facilitate the simplification of Boolean algebra functions. Take the Boolean or binary function described by the following truth table.

Truth table of a function

	A	B	C	D	f(A, B, C, D)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1

12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

Following are two different notations describing the same function in unsimplified Boolean algebra, using the Boolean variables A, B, C, D , and their inverses.

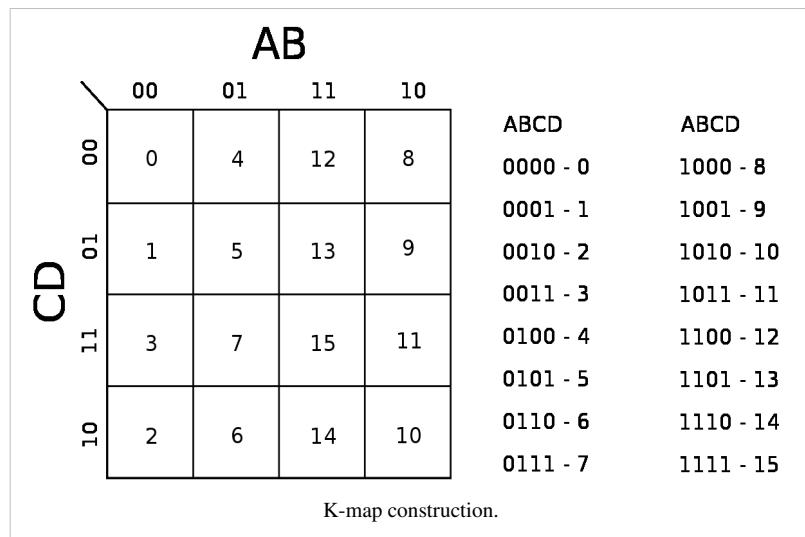
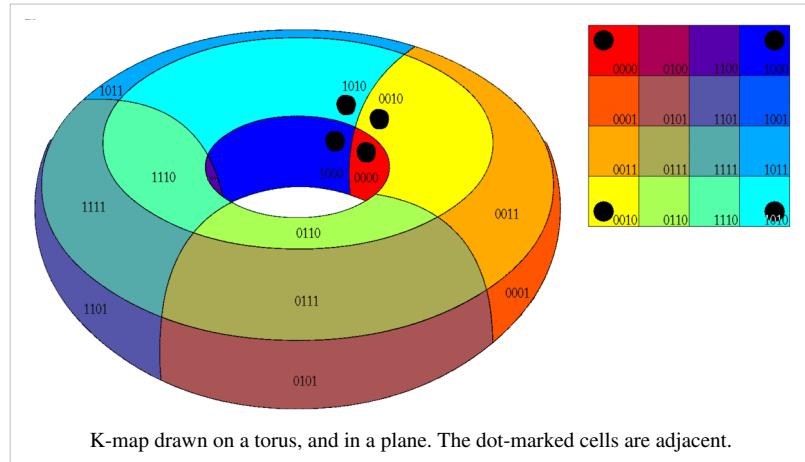
- $f(A, B, C, D) = \sum m(6, 8, 9, 10, 11, 12, 13, 14)$ Note: The values inside \sum are the minterms to map (i.e. rows that have output 1 in the truth table).
- $f(A, B, C, D) = (\overline{A}BC\overline{D}) + (A\overline{B}\overline{C}\overline{D}) + (A\overline{B}\overline{C}D) + (A\overline{B}C\overline{D}) + (A\overline{B}CD) + (ABC\overline{D}) + (AB\overline{C}D) + (ABCD)$

Karnaugh map

In this case, the four input variables can be combined in 16 different ways, so the truth table has 16 rows, and the Karnaugh map has 16 positions. The Karnaugh map is therefore arranged in a 4×4 grid.

The row and column values (shown across the top, and down the left side of the Karnaugh map) are ordered in Gray code rather than binary numerical order. Gray code ensures that only one variable changes between each pair of adjacent cells. Each cell of the completed Karnaugh map contains a binary digit representing the function's output for that combination of inputs.

After the Karnaugh map has been constructed it is used to find one of the simplest possible forms—a canonical form—for the information in the truth table. Adjacent 1s in the Karnaugh map represent opportunities to simplify the expression. The minterms ('minimal terms') for the final expression are found by encircling groups of 1s in the map. Minterm groups must be rectangular and must have an area that is a power of two (i.e. 1, 2, 4, 8...). Minterm rectangles should be as large as possible without containing any 0s. Groups may overlap in order to make each one larger. The optimal groupings in this example are marked by the green, red and blue lines, and the red and green groups overlap. The red group is a 2×2 square, the green group is a 4×1 rectangle, and the overlap area is indicated in brown.



The grid is toroidally connected, which means that rectangular groups can wrap across the edges (see picture). Cells on the extreme right are actually 'adjacent' to those on the far left; similarly, so are those at the very top and those at

the bottom. Therefore $A\bar{D}$ can be a valid term—it includes cells 12 and 8 at the top, and wraps to the bottom to include cells 10 and 14—as is $\bar{B}\bar{D}$, which includes the four corners.

Solution

Once the Karnaugh map has been constructed and the adjacent 1s linked by rectangular and square boxes, the algebraic minterms can be found by examining which variables stay the same within each box.

For the red grouping:

- The variable A is the same and is equal to 1 throughout the box, therefore it should be included in the algebraic representation of the red minterm.
- Variable B does not maintain the same state (it shifts from 1 to 0), and should therefore be excluded.
- C does not change. It is always 0 so its complement, NOT-C, should be included thus, \bar{C} .
- D changes, so it is excluded as well.

Thus the first minterm in the Boolean sum-of-products expression is $A\bar{C}$.

For the green grouping, A and B maintain the same state, while C and D change. B is 0 and has to be negated before it can be included. Thus the second term is $A\bar{B}$.

In the same way, the blue grouping gives the term $BC\bar{D}$.

The solutions of each grouping are combined thus $A\bar{C} + A\bar{B} + BC\bar{D}$.

Thus the Karnaugh map has guided a simplification of $f(A, B, C, D) = (\bar{A}BC\bar{D}) + (\bar{A}\bar{B}C\bar{D}) + (A\bar{B}CD) + (A\bar{B}C\bar{D}) + (A\bar{B}CD) + (ABC\bar{D}) + (AB\bar{C}D) + (ABC\bar{D})$ to

$$f(A, B, C, D) = A\bar{C} + A\bar{B} + BC\bar{D}$$

It would also have been possible to derive this simplification by carefully applying the axioms of boolean algebra, but the time it takes to find it grows exponentially with the number of terms.

Inverse

The inverse of a function is solved in the same way by grouping the 0s instead.

The three terms to cover the inverse are all shown with grey boxes with different colored borders:

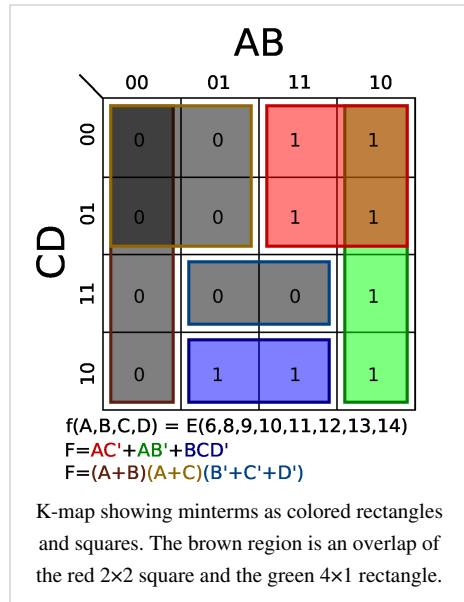
- brown— $\bar{A}\bar{B}$
- gold— $\bar{A}\bar{C}$
- blue— BCD

This yields the inverse:

$$\bar{F} = \bar{A}\bar{B} + \bar{A}\bar{C} + BCD$$

Through the use of De Morgan's laws, the product of sums can be determined:

$$\begin{aligned}\bar{F} &= \overline{\bar{A}\bar{B} + \bar{A}\bar{C} + BCD} \\ F &= (A + B)(A + C)(\bar{B} + \bar{C} + \bar{D})\end{aligned}$$



Don't cares

Karnaugh maps also allow easy minimizations of functions whose truth tables include "don't care" conditions. A "don't care" condition is a combination of inputs for which the designer doesn't care what the output is. Therefore "don't care" conditions can either be included in or excluded from any circled group, whichever makes it larger. They are usually indicated on the map with a dash or X.

The example on the right is the same as the example above but with the value of F for $ABCD = 1111$ replaced by a "don't care". This allows the red term to expand all the way down and, thus, removes the green term completely.

This yields the new minimum equation:

$$F = A + BCD.$$

Note that the first term is just A not $A\bar{C}$. In this case, the don't care has dropped a term (the green); simplified another (the red); and removed the race hazard (the yellow as shown in a following section).

The inverse case is simplified as follows

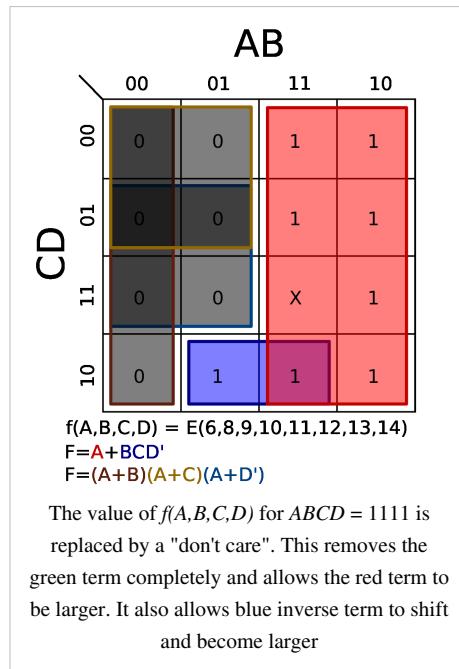
$$\bar{F} = \overline{AB} + \overline{AC} + \overline{AD}$$

Race hazards

Elimination

Karnaugh maps are useful for detecting and eliminating race hazards. Race hazards are very easy to spot using a Karnaugh map, because a race condition may exist when moving between any pair of adjacent, but disjointed, regions circled on the map.

- In the example above, a potential race condition exists when C is 1 and D is 0, A is 1, and B changes from 1 to 0 (moving from the blue state to the green state). For this case, the output is defined to remain unchanged at 1, but because this transition is not covered by a specific term in the equation, a potential for a *glitch* (a momentary transition of the output to 0) exists.
- There is a second potential glitch in the same example that is more difficult to spot: when D is 0 and A and B are both 1, with C changing from 1 to 0 (moving from the blue state to the red state). In this case the glitch wraps around from the top of the map to the bottom.

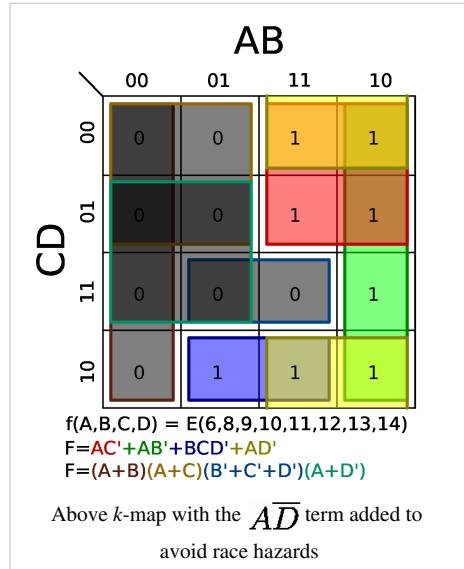


Whether these glitches will actually occur depends on the physical nature of the implementation, and whether we need to worry about it depends on the application.

In this case, an additional term of \overline{AD} would eliminate the potential race hazard, bridging between the green and blue output states or blue and red output states: this is shown as the yellow region (which wraps around from the bottom to the top of the right half) in the diagram to the right.

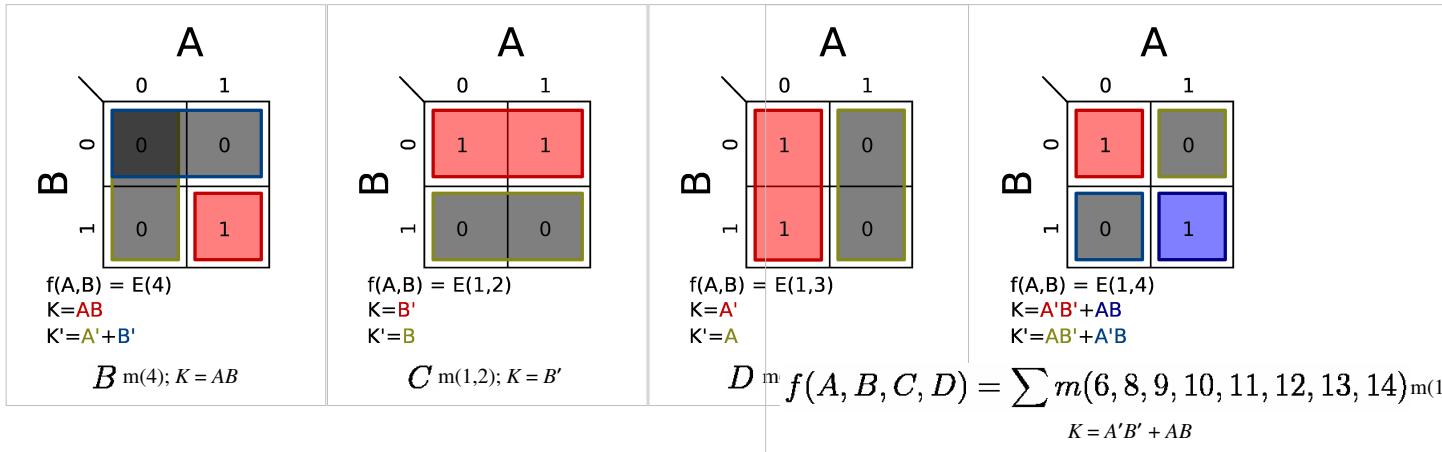
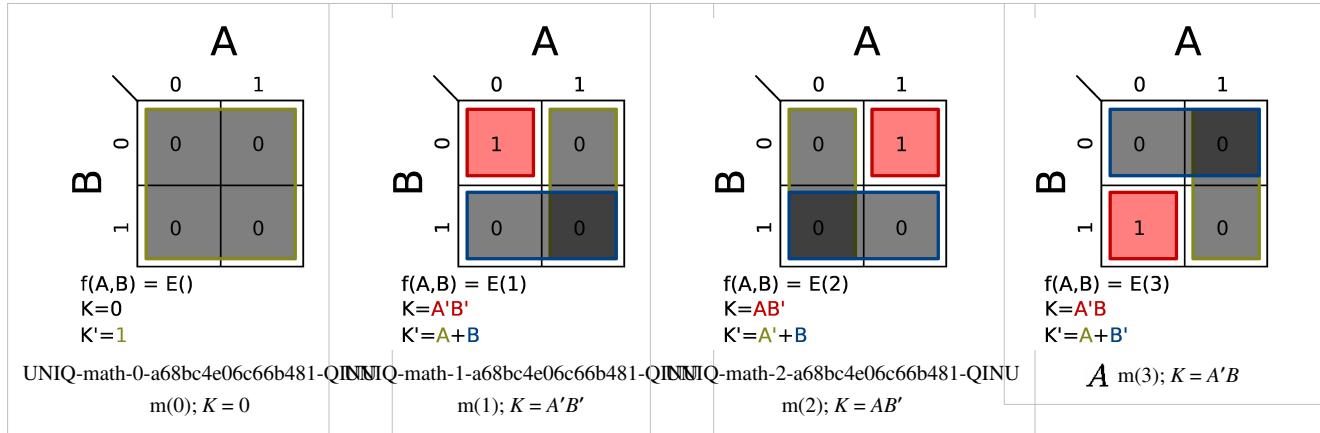
The term is redundant in terms of the static logic of the system, but such redundant, or consensus terms, are often needed to assure race-free dynamic performance.

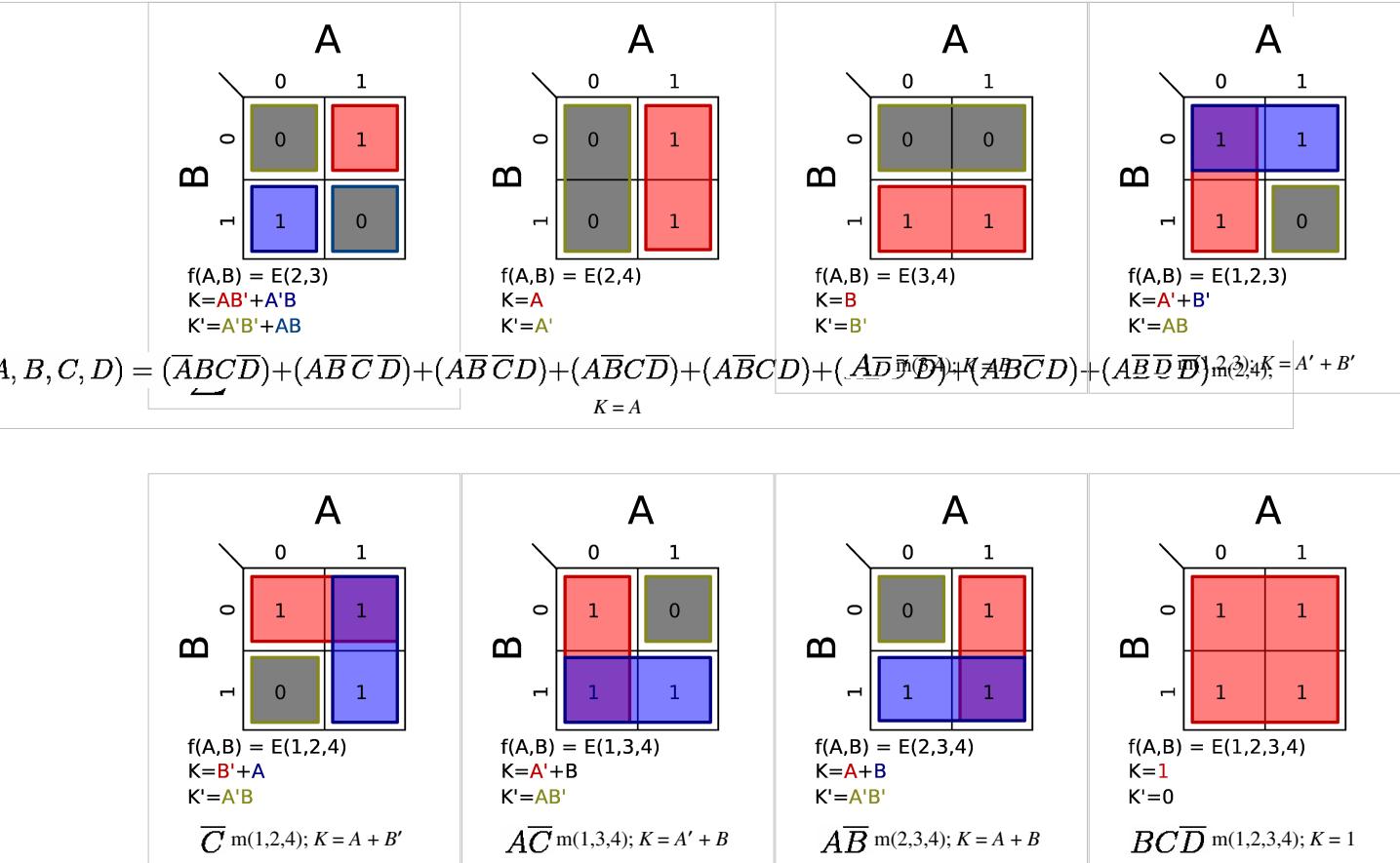
Similarly, an additional term of \overline{AD} must be added to the inverse to eliminate another potential race hazard. Applying De Morgan's laws creates another product of sums expression for F , but with a new factor of $(A + \overline{D})$.



2-variable map examples

The following are all the possible 2-variable, 2×2 Karnaugh maps. Listed with each is the minterms as a function of $\sum m()$ and the race hazard free (*see previous section*) minimum equation.





References

Further reading

- Karnaugh, Maurice (November 1953). "The Map Method for Synthesis of Combinational Logic Circuits". *Transactions of the American Institute of Electrical Engineers part I* **72** (9): 593–599. doi: 10.1109/TCE.1953.6371932 (<http://dx.doi.org/10.1109/TCE.1953.6371932>).
- Katz, Randy (1998) [1994]. *Contemporary Logic Design*. The Benjamin/Cummings. pp. 70–85. doi: 10.1016/0026-2692(95)90052-7 ([http://dx.doi.org/10.1016/0026-2692\(95\)90052-7](http://dx.doi.org/10.1016/0026-2692(95)90052-7)). ISBN 0-8053-2703-7.
- Veitch, Edward W. (1952). "A Chart Method for Simplifying Truth Functions". *ACM Annual Conference/Annual Meeting: Proceedings of the 1952 ACM Annual Meeting (Pittsburg)* (ACM, NY): pp. 127–133. doi: 10.1145/609784.609801 (<http://dx.doi.org/10.1145/609784.609801>).
- Vingron, Dr. Shimon Peter (2004) [2004]. "Karnaugh Maps". *Switching Theory: Insight Through Predicate Logic*. Berlin, Heidelberg, New York: Springer-Verlag. pp. 57–76. ISBN 3-540-40343-4.
- Wickes, William E. (1968). *Logic Design with Integrated Circuits*. New York: John Wiley & Sons. pp. 36–49. Library of Congress Catalog Number: 68-21185. "A refinement of the Venn diagram in that circles are replaced by squares and arranged in a form of matrix. The Veitch diagram labels the squares with the minterms. Karnaugh assigned 1s and 0s to the squares and their labels and deduced the numbering scheme in common use."

External links

- Quine–McCluskey algorithm implementation with a search of all solutions ([http://frederic.carpon.perso.sfr.fr/Quine-McCluskey_\(frederic_carpon_implementation\).php](http://frederic.carpon.perso.sfr.fr/Quine-McCluskey_(frederic_carpon_implementation).php)), by Frédéric Carpon.
- Detect Overlapping Rectangles (http://gandraxa.com/detect_overlapping_subrectangles.xml), by Herbert Glarner.
- Using Karnaugh maps in practical applications (<http://www.sccs.swarthmore.edu/users/06/adem/engin/e15/lab1/>), Circuit design project to control traffic lights.
- K-Map Tutorial for 2,3,4 and 5 variables (<http://fullchipdesign.com/kmap2v.htm>)
- Karnaugh Map Example (http://www.generalnumbers.com/karnaugh_application1.html)

Propositional directed acyclic graph

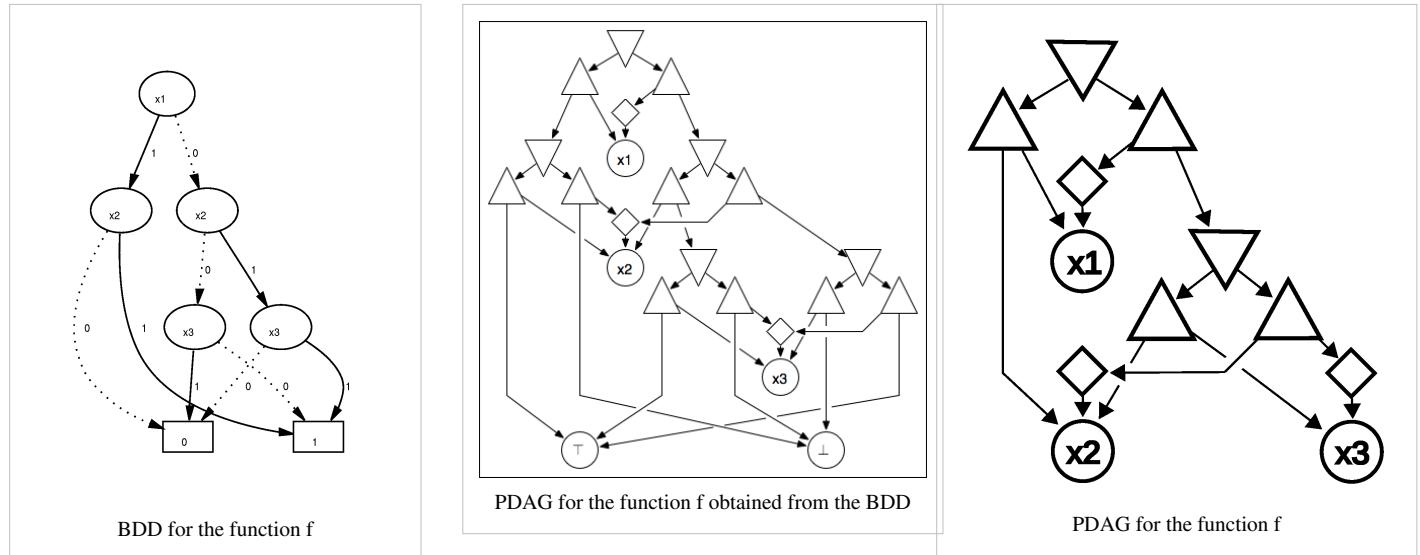
A **propositional directed acyclic graph (PDAG)** is a data structure that is used to represent a Boolean function. A Boolean function can be represented as a rooted, directed acyclic graph of the following form:

- Leaves are labeled with \top (true), \perp (false), or a Boolean variable.
- Non-leaves are \triangle (logical and), ∇ (logical or) and \Diamond (logical not).
- \triangle - and ∇ -nodes have at least one child.
- \Diamond -nodes have exactly one child.

Leaves labeled with \top (\perp) represent the constant Boolean function which always evaluates to 1 (0). A leaf labeled with a Boolean variable x is interpreted as the assignment $x = 1$, i.e. it represents the Boolean function which evaluates to 1 if and only if $x = 1$. The Boolean function represented by a \triangle -node is the one that evaluates to 1, if and only if the Boolean function of all its children evaluate to 1. Similarly, a ∇ -node represents the Boolean function that evaluates to 1, if and only if the Boolean function of at least one child evaluates to 1. Finally, a \Diamond -node represents the complementary Boolean function of its child, i.e. the one that evaluates to 1, if and only if the Boolean function of its child evaluates to 0.

PDAG, BDD, and NNF

Every **binary decision diagram (BDD)** and every **negation normal form (NNF)** are also a PDAG with some particular properties. The following pictures represent the Boolean function $f(x_1, x_2, x_3) = -x_1 * -x_2 * -x_3 + x_1 * x_2 + x_2 * x_3$:



References

- M. Wachter & R. Haenni, "Propositional DAGs: a New Graph-Based Language for Representing Boolean Functions", KR'06, 10th International Conference on Principles of Knowledge Representation and Reasoning, Lake District, UK, 2006.
- M. Wachter & R. Haenni, "Probabilistic Equivalence Checking with Propositional DAGs", Technical Report iam-2006-001, Institute of Computer Science and Applied Mathematics, University of Bern, Switzerland, 2006.
- M. Wachter, R. Haenni & J. Jonczy, "Reliability and Diagnostics of Modular Systems: a New Probabilistic Approach", DX'06, 18th International Workshop on Principles of Diagnosis, Peñaranda de Duero, Burgos, Spain, 2006.

Quine–McCluskey algorithm

The **Quine–McCluskey algorithm** (or **the method of prime implicants**) is a method used for minimization of boolean functions which was developed by W.V. Quine and Edward J. McCluskey in 1956. It is functionally identical to Karnaugh mapping, but the tabular form makes it more efficient for use in computer algorithms, and it also gives a deterministic way to check that the minimal form of a Boolean function has been reached. It is sometimes referred to as the tabulation method.

The method involves two steps:

1. Finding all prime implicants of the function.
2. Use those prime implicants in a *prime implicant chart* to find the essential prime implicants of the function, as well as other prime implicants that are necessary to cover the function.

Complexity

Although more practical than Karnaugh mapping when dealing with more than four variables, the Quine–McCluskey algorithm also has a limited range of use since the problem it solves is NP-hard: the runtime of the Quine–McCluskey algorithm grows exponentially with the number of variables. It can be shown that for a function of n variables the upper bound on the number of prime implicants is $3^n/n$. If $n = 32$ there may be over $6.5 * 10^{15}$ prime implicants. Functions with a large number of variables have to be minimized with potentially non-optimal heuristic methods, of which the Espresso heuristic logic minimizer is the de facto standard.^[1]

Example

Step 1: finding prime implicants

Minimizing an arbitrary function:

$$f(A, B, C, D) = \sum m(4, 8, 10, 11, 12, 15) + d(9, 14).$$

This expression says that the output function f will be 1 for the minterms 4,8,10,11,12 and 15 (denoted by the 'm' term). But it also says that we don't care about the output for 9 and 14 combinations. ('d' stands for don't care).

	A	B	C	D	f
m0	0	0	0	0	0
m1	0	0	0	1	0
m2	0	0	1	0	0
m3	0	0	1	1	0
m4	0	1	0	0	1
m5	0	1	0	1	0
m6	0	1	1	0	0
m7	0	1	1	1	0
m8	1	0	0	0	1
m9	1	0	0	1	x
m10	1	0	1	0	1
m11	1	0	1	1	1
m12	1	1	0	0	1

m13	1	1	0	1		0
m14	1	1	1	0	x	
m15	1	1	1	1		1

One can easily form the canonical sum of products expression from this table, simply by summing the minterms (leaving out don't-care terms) where the function evaluates to one:

$$f_{A,B,C,D} = A'BC'D' + AB'C'D' + AB'CD' + AB'CD + ABC'D' + ABCD,$$

which is not minimal. So to optimize, all minterms that evaluate to one are first placed in a minterm table. Don't-care terms are also added into this table, so they can be combined with minterms:

Number of 1s	Minterm	Binary Representation
1	m4	0100
	m8	1000
2	m9	1001
	m10	1010
	m12	1100
3	m11	1011
	m14	1110
4	m15	1111

At this point, one can start combining minterms with other minterms. If two terms vary by only a single digit changing, that digit can be replaced with a dash indicating that the digit doesn't matter. Terms that can't be combined any more are marked with a "*". When going from Size 2 to Size 4, treat '-' as a third bit value. Ex: -110 and -100 or -11- can be combined, but not -110 and 011-. (Trick: Match up the '-' first.)

Number of 1s	Minterm	0-Cube	Size 2 Implicants	Size 4 Implicants
1	m4	0100	m(4,12) -100*	m(8,9,10,11) 10--*
	m8	1000	m(8,9) 100-	m(8,10,12,14) 1--0*
--	--	--	m(8,10) 10-0	--
2	m9	1001	m(8,12) 1-00	m(10,11,14,15) 1-1-*
	m10	1010	--	--
	m12	1100	m(9,11) 10-1	--
--	--	--	m(10,11) 101-	--
3	m11	1011	m(10,14) 1-10	--
	m14	1110	m(12,14) 11-0	--
4	m15	1111	m(11,15) 1-11	--
			m(14,15) 111-	

Note: In this example, none of the terms in the size 4 implicants table can be combined any further. Be aware that this processing should be continued otherwise (size 8 etc.).

Step 2: prime implicant chart

None of the terms can be combined any further than this, so at this point we construct an essential prime implicant table. Along the side goes the prime implicants that have just been generated, and along the top go the minterms specified earlier. The don't care terms are not placed on top - they are omitted from this section because they are not necessary inputs.

	4	8	10	11	12	15		=>	A	B	C	D
m(4,12)*	X				X		-100	=>	-	1	0	0
m(8,9,10,11)		X	X	X			10--	=>	1	0	-	-
m(8,10,12,14)		X	X		X		1--0	=>	1	-	-	0
m(10,11,14,15)*			X	X		X	1-1-	=>	1	-	1	-

Here, each of the *essential* prime implicants has been starred - the second prime implicant can be 'covered' by the third and fourth, and the third prime implicant can be 'covered' by the second and first, and neither is thus essential. If a prime implicant is essential then, as would be expected, it is necessary to include it in the minimized boolean equation. In some cases, the essential prime implicants do not cover all minterms, in which case additional procedures for chart reduction can be employed. The simplest "additional procedure" is trial and error, but a more systematic way is Petrick's Method. In the current example, the essential prime implicants do not handle all of the minterms, so, in this case, one can combine the essential implicants with one of the two non-essential ones to yield one of these two equations:

$$f_{A,B,C,D} = BC'D' + AB' + AC$$

$$f_{A,B,C,D} = BC'D' + AD' + AC.$$

Both of those final equations are functionally equivalent to the original, verbose equation:

$$f_{A,B,C,D} = A'BC'D' + AB'C'D' + AB'C'D + AB'CD' + AB'CD + ABC'D' + ABCD' + ABCD.$$

References

[1] V.P. Nelson e.a., *Digital Circuit Analysis and Design*, Prentice Hall, 1995, pag. 234

External links

- All about Quine-McClusky (<http://www.embedded.com/columns/programmerstoolbox/29111968>), article by Jack Crenshaw comparing Quine-McClusky to Karnaugh maps
- Java-Applet (<http://user.cs.tu-berlin.de/~lordmaik/projects/quinemccluskey/quinemccluskey/quineapplet.htm>)Wikipedia:Link rot Applet to minimize a boolean function based on QuineMcCluskey Algorithm. (German page)
- Karma 3 (<http://www.inf.ufrgs.br/logics/>), A set of logic synthesis tools including Karnaugh maps, Quine-McCluskey minimization, BDDs, probabilities, teaching module and more. Logic Circuits Synthesis Labs (LogiCS) - UFRGS, Brazil.
- A. Costa BFnc (<http://www4.dei.isep.ipp.pt/acc/bfunc/>), QMC based boolean logic simplifiers supporting up to 64 inputs / 64 outputs (independently) or 32 outputs (simultaneously)
- Java applet (<http://www25.brinkster.com/denshade/QuineMcCluskey.html>)Wikipedia:Link rot to display all the generated primes.
- Python Implementation (<http://cheeseshop.python.org/pypi/qm/0.2>) by Robert Dick.
- Python Implementation (http://symlog.git.sourceforge.net/git/gitweb.cgi?p=symlog/symlog;a=blob_plain;f=symlog/logic.py;hb=HEAD) for symbolically reducing Boolean expressions.

- Quinessence (<http://sourceforge.net/projects/quinessence/>), an open source implementation written in Free Pascal by Marco Caminati.
- QCA (<http://cran.r-project.org/web/packages/QCA/index.html>) an open source, R based implementation used in the social sciences, by Adrian Duşa
- A series of two articles describing the algorithm(s) implemented in R: first article (<http://www.compasss.org/files/WPfiles/Dusa2007.pdf>) Wikipedia:Link rot and second article (<http://www.compasss.org/files/WPfiles/Dusa2007a.pdf>) Wikipedia:Link rot. The R implementation is exhaustive and it offers complete and exact solutions. It processes up to 20 input variables.
- minBool (http://www.p0p0v.com/science/#_minBool) an implementation by Andrey Popov.
- (http://www-ihs.theoinf.tu-ilmenau.de/~sane/projekte/qmc/embed_qmc.html), an applet for a step by step analyze of the QMC- algorithm by Christian Roth
- (<http://sourceforge.net/projects/qmcs>) SourceForge.net C++ program implementing the algorithm.
- Perl Module (<https://metacpan.org/module/Algorithm::QuineMcCluskey>) by Darren M. Kulp.
- (<http://sites.google.com/site/simpogical/download>) Tutorial on Quine-McCluskey and Petrick's method (pdf).
- (<http://code.google.com/p/quine-mccluskey-petrick/source/browse/>) C++ implementation (including Petrick) based on the tutorial above
- (<http://sourceforge.net/projects/mini-qmc>) Public Domain console based C program on SourceForge.net.
- Tomaszewski, S. P., Celik, I. U., Antoniou, G. E., "WWW-based Boolean function minimization" INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS AND COMPUTER SCIENCE, VOL 13; PART 4, pages 577-584, 2003. (<http://matwbn.icm.edu.pl/ksiazki/amc/amc13/amc13414.pdf>)
- For a fully worked out example visit: <http://www.cs.ualberta.ca/~amaral/courses/329/webslides/Topic5-QuineMcCluskey/sld024.htm>

Reed-Muller expansion

In Boolean logic, a **Reed-Muller** (or **Davio**) **expansion** is a decomposition of a boolean function.

For a boolean function $f(x_1, \dots, x_n)$ we set with respect to x_i :

$$f_{x_i}(x) = f(x_1, \dots, x_{i-1}, 1, x_{i+1}, \dots, x_n)$$

$$f_{\bar{x}_i}(x) = f(x_1, \dots, x_{i-1}, 0, x_{i+1}, \dots, x_n)$$

$$\frac{\partial f}{\partial x_i} = f_{x_i}(x) \oplus f_{\bar{x}_i}(x)$$

as the positive and negative cofactors of f , and the boolean derivation of f .

Then we have for the Reed-Muller or positive Davio expansion:

$$f = f_{\bar{x}_i} \oplus x_i \frac{\partial f}{\partial x_i}.$$

Similar to binary decision diagrams (BDDs), where nodes represent Shannon expansion with respect to the according variable, we can define a decision diagram based on the Reed-Muller expansion. These decision diagrams are called functional BDDs (FBDDs).

References

- Kebischull, U. and Rosenstiel, W., *Efficient graph-based computation and manipulation of functional decision diagrams*, Proceedings 4th European Conference on Design Automation, 1993, pp. 278–282

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

<i>p</i>	<i>p</i>
<i>Operand</i>	<i>Value</i>
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 $\neg p$, Np , Fpq , or $\sim p$) is as follows:

Logical Negation

p	$\neg p$
T	F
F	T

Binary operations

Truth table for all binary logical operators

Here is a truth table giving definitions of all 16 of the possible truth functions of 2 binary variables (P,Q are thus boolean variables; for details about the operators see below):

P	Q	F^0	NOR^1	Xq^2	$\neg p^3$	$\not\rightarrow^4$	$\neg q^5$	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

where T = true and F = false.

Key:

Operation name				
0	Opq	F	false	Contradiction
1	Xpq	NOR	\downarrow	Logical NOR
2	Mpq	Xq		Converse nonimplication
3	Fpq	Np	$\neg p$	Negation
4	Lpq	Xp	\square	Material nonimplication
5	Gpq	Nq	$\neg q$	Negation
6	Jpq	XOR	\oplus	Exclusive disjunction
7	Dpq	NAND	\uparrow	Logical NAND
8	Kpq	AND	\wedge	Logical conjunction
9	Epq	XNOR	If and only if	Logical biconditional
10	Hpq	q		Projection function
11	Cpq	XNp	if/then	Material implication
12	Ipq	p		Projection function
13	Bpq	XNq	then/if	Converse implication
14	Apq	OR	\vee	Logical disjunction
15	Vpq	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 $p \sqcap q$, Kpq , $p \& q$, or $p \cdot q$) is as follows:

Logical Conjunction

p	q	$p \sqcap q$
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 $p \sqcup q$, Apq , $p \sqcup q$, or $p + q$) is as follows:

Logical Disjunction

p	q	$p \sqcup q$
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 and the material conditional are both associated with an operation on two logical values, typically the values of two propositions, that produces a value of *false* just in the singular case the first operand is true and the second operand is false.

The truth table associated with the material conditional **if p then q** (symbolized as $p \rightarrow q$) and the logical implication **p implies q** (symbolized as $p \sqsubset q$, or Cpq) is as follows:

Logical Implication

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

It may also be useful to note that $p \rightarrow q$ is equivalent to $\neg p \sqcup q$.

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 $p \text{ XNOR } q$ (also written as $p \leftrightarrow q$, $\text{Eq}p$, $p = q$, or $p \sqcap q$) is as follows:

Logical Equality

p	q	$p \sqcap q$
T	T	T
T	F	F
F	T	F
F	F	T

So $p \text{ EQ } q$ 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 $p \text{ XOR } q$ (also written as $p \sqcup q$, Jpq , or $p \neq q$) is as follows:

Exclusive Disjunction

p	q	$p \sqcup q$
T	T	F
T	F	T
F	T	T
F	F	F

For two propositions, **XOR** can also be written as $(p = 1 \wedge q = 0) \vee (p = 0 \wedge q = 1)$.

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 $p \text{ NAND } q$ (also written as $p \uparrow q$, Dpq , or $p \mid q$) is as follows:

Logical NAND

p	q	$p \uparrow q$
T	T	F
T	F	T
F	T	T
F	F	T

It is frequently useful to express a logical operation as a compound operation, that is, as an operation that is built up or composed from other operations. Many such compositions are possible, depending on the operations that are taken as basic or "primitive" and the operations that are taken as composite or "derivative".

In the case of logical NAND, it is clearly expressible as a compound of NOT and AND.

The negation of a conjunction: $\neg(p \wedge q)$, and the disjunction of negations: $(\neg p) \vee (\neg q)$ can be tabulated as follows:

p	q	$p \wedge q$	$\neg(p \wedge q)$	$\neg p$	$\neg q$	$(\neg p) \vee (\neg q)$
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. \downarrow is also known as the Peirce arrow after its inventor, Charles Sanders Peirce, and is a Sole sufficient operator.

The truth table for $p \text{ NOR } q$ (also written as $p \downarrow q$, Xpq , or $p \sqcup q$) is as follows:

Logical NOR

p	q	$p \downarrow q$
T	T	F
T	F	F
F	T	F
F	F	T

The negation of a disjunction $\neg(p \vee q)$, and the conjunction of negations $(\neg p) \wedge (\neg q)$ can be tabulated as follows:

p	q	$p \sqcap q$	$\neg(p \sqcap q)$	$\neg p$	$\neg q$	$(\neg p) \sqcup (\neg q)$
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

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

This equivalence is one of De Morgan's laws.

Applications

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

$$\text{Logical Equivalence : } (p \rightarrow q) = (\neg p \sqcup q)$$

p	q	$\neg p$	$\neg p \sqcup q$	$p \rightarrow q$
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 $p \rightarrow q$ is logically equivalent to $\neg p \vee q$.

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	$P \wedge Q$	$P \vee Q$	$P \underline{\vee} Q$	$P \Delta Q$	$P \Rightarrow Q$	$P \Leftarrow Q$	$P \iff Q$
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)

Δ = XNOR (exclusive nor)

\rightarrow = conditional "if-then"

\leftarrow = conditional "(then)-if"

\iff biconditional or "if-and-only-if" is logically equivalent to Δ : 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:

\square	F	T	\square	F	T
F	F	F	F	F	T
T	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 $k = V_0 \cdot 2^0 + V_1 \cdot 2^1 + V_2 \cdot 2^2 + \dots + V_n \cdot 2^n$.

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 1's and 0's, 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. Pierce appears to be the earliest logicians (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

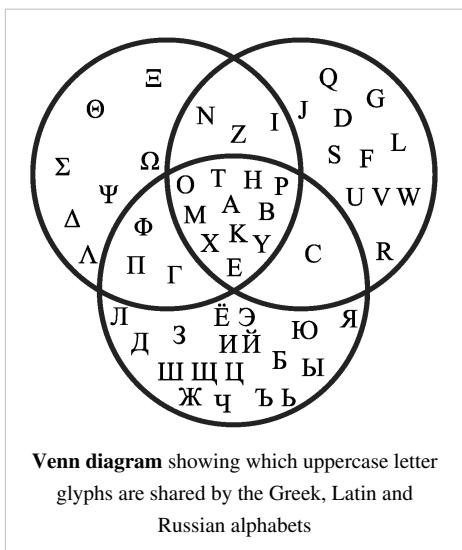
Further reading

- 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>)
- Online Truth Table Generator (<http://knowpapa.com/truth-table/>)
- 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>)

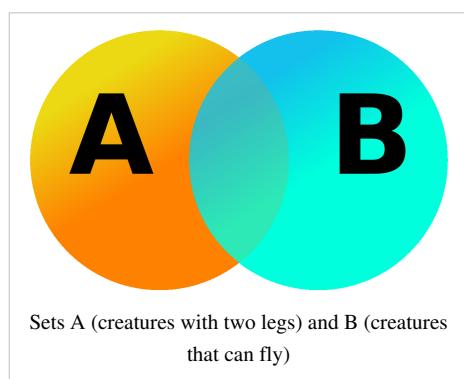
Venn diagram



Probability theory	
•	Probability axioms
•	Probability space
•	Sample space
•	Elementary event
•	Event
•	Random variable
•	Probability measure
•	Complementary event
•	Joint probability
•	Marginal probability
•	Conditional probability
•	Independence
•	Conditional independence
•	Law of total probability
•	Law of large numbers
•	Bayes' theorem
•	Boole's inequality
•	Venn diagram
•	Tree diagram

A **Venn diagram** or **set diagram** is a diagram that shows all possible logical relations between a finite collection of sets. Venn diagrams were conceived around 1880 by John Venn. They are used to teach elementary set theory, as well as illustrate simple set relationships in probability, logic, statistics, linguistics and computer science.

Example



This example involves two sets, A and B, represented here as coloured circles. The orange circle, set A, represents all living creatures that are two-legged. The blue circle, set B, represents the living creatures that can fly. Each separate type of creature can be imagined as a point somewhere in the diagram. Living creatures that both can fly *and* have two legs—for example, parrots—are then in both sets, so they correspond to points in the area where the blue and orange circles overlap. That area contains all such and only such living creatures.

Humans and penguins are bipedal, and so are then in the orange circle, but since they cannot fly they appear in the left part of the orange circle, where it does not overlap with the blue circle. Mosquitoes have six legs, and fly, so the point for mosquitoes is in the part of the blue circle that does not overlap with the orange one. Creatures that are not two-legged and cannot fly (for example, whales and spiders) would all be represented by points outside both circles.

The combined area of sets A and B is called the *union* of A and B, denoted by $A \cup B$. The union in this case contains all living creatures that are either two-legged or that can fly (or both).

The area in both A and B, where the two sets overlap, is called the *intersection* of A and B, denoted by $A \cap B$. For example, the intersection of the two sets is not empty, because there *are* points that represent creatures that are in *both* the orange and blue circles.

History

Venn diagrams were introduced in 1880 by John Venn (1834–1923) in a paper entitled *On the Diagrammatic and Mechanical Representation of Propositions and Reasonings* in the "Philosophical Magazine and Journal of Science", about the different ways to represent propositions by diagrams. The use of these types of diagrams in formal logic, according to Ruskey and M. Weston, is "not an easy history to trace, but it is certain that the diagrams that are popularly associated with Venn, in fact, originated much earlier. They are rightly associated with Venn, however, because he comprehensively surveyed and formalized their usage, and was the first to generalize them".

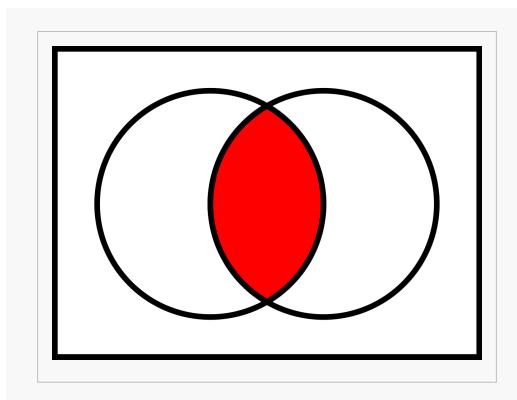
Venn himself did not use the term "Venn diagram" and referred to his invention as "Eulerian Circles." For example, in the opening sentence of his 1880 article Venn writes, "Schemes of diagrammatic representation have been so familiarly introduced into logical treatises during the last century or so, that many readers, even those who have made no professional study of logic, may be supposed to be acquainted with the general nature and object of such devices. Of these schemes one only, viz. that commonly called 'Eulerian circles,' has met with any general acceptance..." The first to use the term "Venn diagram" was Clarence Irving Lewis in 1918, in his book "A Survey of Symbolic Logic".

Venn diagrams are very similar to Euler diagrams, which were invented by Leonhard Euler (1708–1783) in the 18th century.^[1] M. E. Baron has noted that Leibniz (1646–1716) in the 17th century produced similar diagrams before Euler, but much of it was unpublished. She also observes even earlier Euler-like diagrams by Ramon Lull in the 13th Century.

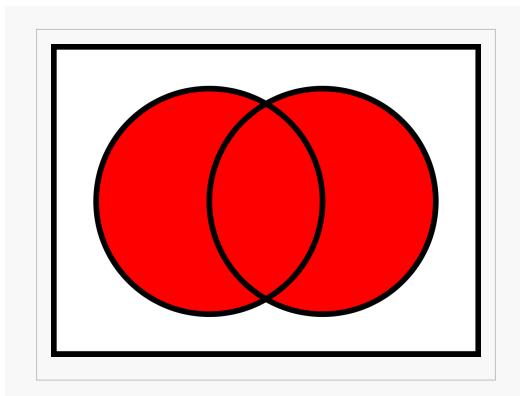
In the 20th century, Venn diagrams were further developed. D.W. Henderson showed in 1963 that the existence of an n -Venn diagram with n -fold rotational symmetry implied that n was a prime number. He also showed that such symmetric Venn diagrams exist when n is 5 or 7. In 2002 Peter Hamburger found symmetric Venn diagrams for $n = 11$ and in 2003, Griggs, Killian, and Savage showed that symmetric Venn diagrams exist for all other primes. Thus rotationally symmetric Venn diagrams exist if and only if n is a prime number.

Venn diagrams and Euler diagrams were incorporated as part of instruction in set theory as part of the new math movement in the 1960s. Since then, they have also been adopted by other curriculum fields such as reading.^[2]

Overview

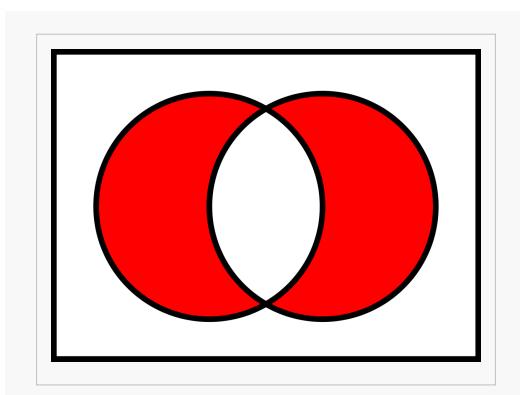
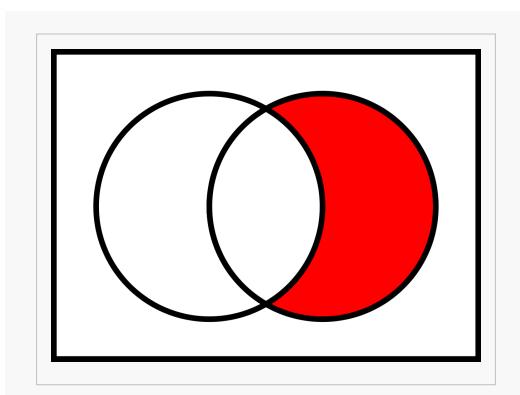
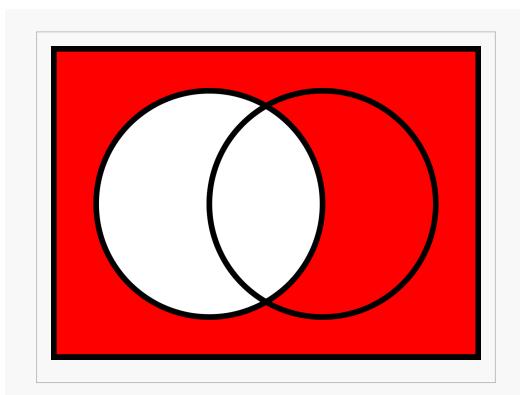


Intersection of two sets
 $A \cap B$



Union of two sets

$$A \cup B$$

Symmetric difference of two sets $A \Delta B$ Relative complement of A (left) in B (right)
 $A^c \cap B = B \setminus A$ 

$$\text{Absolute complement of } A \text{ in } U \\ A^c = U \setminus A$$

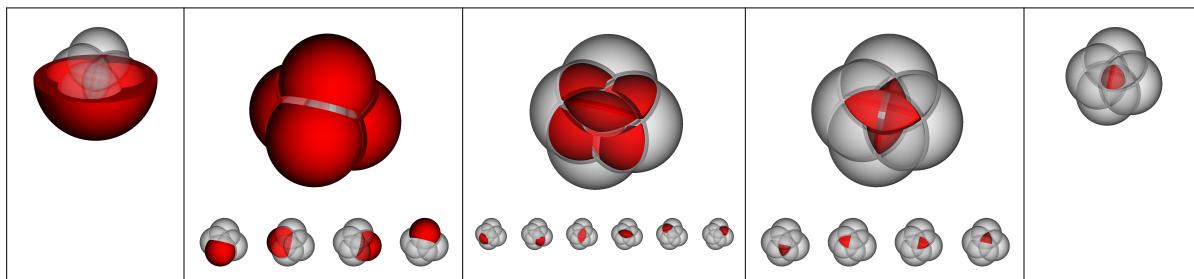
A Venn diagram is constructed with a collection of simple closed curves drawn in a plane. According to Lewis, the "principle of these diagrams is that classes [or *sets*] be represented by regions in such relation to one another that all the possible logical relations of these classes can be indicated in the same diagram. That is, the diagram initially leaves room for any possible relation of the classes, and the actual or given relation, can then be specified by indicating that some particular region is null or is not-null":¹⁵⁷

Venn diagrams normally comprise overlapping circles. The interior of the circle symbolically represents the elements of the set, while the exterior represents elements that are not members of the set. For instance, in a two-set Venn diagram, one circle may represent the group of all wooden objects, while another circle may represent the set of all tables. The overlapping area or *intersection* would then represent the set of all wooden tables. Shapes other than circles can be employed as shown below by Venn's own higher set diagrams. Venn diagrams do not generally contain information on the relative or absolute sizes (cardinality) of sets; i.e. they are schematic diagrams.

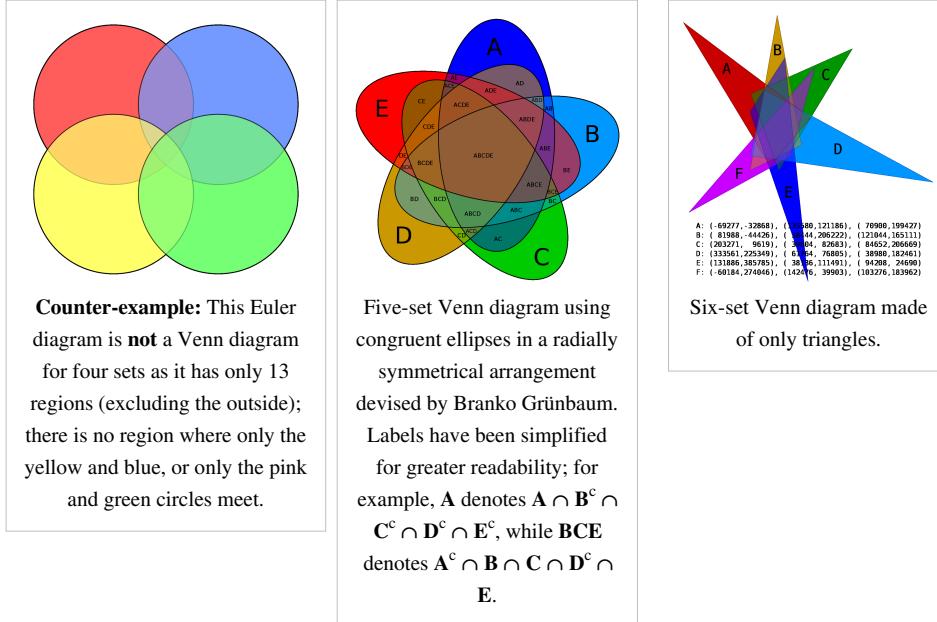
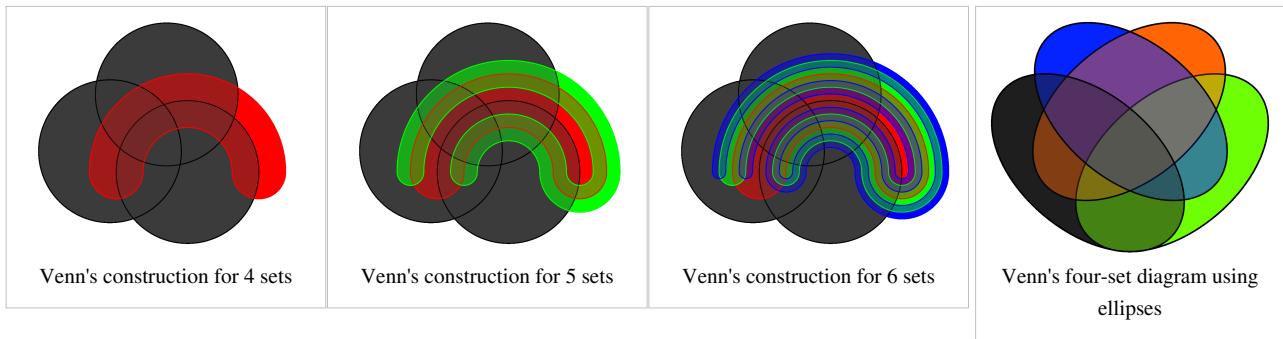
Venn diagrams are similar to Euler diagrams. However, a Venn diagram for n component sets must contain all 2^n hypothetically possible zones that correspond to some combination of inclusion or exclusion in each of the component sets. Euler diagrams contain only the actually possible zones in a given context. In Venn diagrams, a shaded zone may represent an empty zone, whereas in an Euler diagram the corresponding zone is missing from the diagram. For example, if one set represents *dairy products* and another *cheeses*, the Venn diagram contains a zone for cheeses that are not dairy products. Assuming that in the context *cheese* means some type of dairy product, the Euler diagram has the cheese zone entirely contained within the dairy-product zone—there is no zone for (non-existent) non-dairy cheese. This means that as the number of contours increases, Euler diagrams are typically less visually complex than the equivalent Venn diagram, particularly if the number of non-empty intersections is small.

Extensions to higher numbers of sets

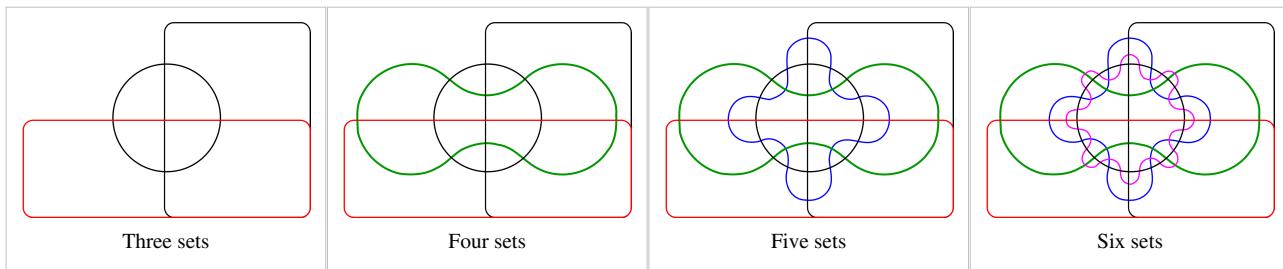
Venn diagrams typically represent two or three sets, but there are forms that allow for higher numbers. Shown below, four intersecting spheres form the highest order Venn diagram that is completely symmetric and can be visually represented. The 16 intersections correspond to the vertices of a tesseract (or the cells of a 16-cell respectively).



For higher numbers of sets, some loss of symmetry in the diagrams is unavoidable. Venn was keen to find "symmetrical figures...elegant in themselves," that represented higher numbers of sets, and he devised a four-set diagram using ellipses (see below). He also gave a construction for Venn diagrams for *any* number of sets, where each successive curve that delimits a set interleaves with previous curves, starting with the three-circle diagram.



Edwards' Venn diagrams



A. W. F. Edwards constructed a series of Venn diagrams for higher numbers of sets by segmenting the surface of a sphere. For example, three sets can be easily represented by taking three hemispheres of the sphere at right angles ($x = 0$, $y = 0$ and $z = 0$). A fourth set can be added to the representation by taking a curve similar to the seam on a tennis ball, which winds up and down around the equator, and so on. The resulting sets can then be projected back to a plane to give *cogwheel* diagrams with increasing numbers of teeth, as shown on the right. These diagrams were devised while designing a stained-glass window in memory of Venn.

Other diagrams

Edwards' Venn diagrams are topologically equivalent to diagrams devised by Branko Grünbaum, which were based around intersecting polygons with increasing numbers of sides. They are also 2-dimensional representations of hypercubes.

Smith^[citation needed] devised similar n -set diagrams using sine curves with the series of equations

$$y_i = \frac{\sin(2^i x)}{2^i} \text{ where } 0 \leq i \leq n-2 \text{ and } i \in \mathbb{N}.$$

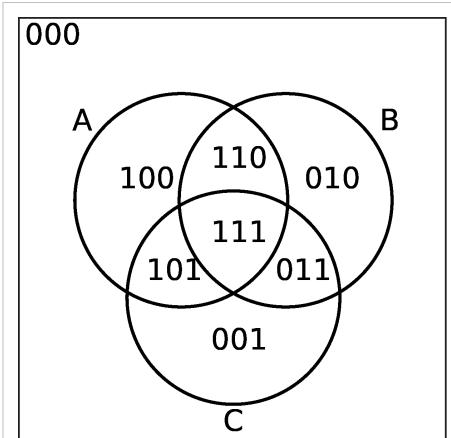
Charles Lutwidge Dodgson devised a five-set diagram.

Related concepts

Venn diagrams correspond to truth tables for the propositions $x \in A$, $x \in B$, etc., in the sense that each region of Venn diagram corresponds to one row of the truth table. Another way of representing sets is with R-Diagrams.

Notes

- [1] In Euler's *Letters to a German Princess*. In Venn's article, however, he suggests that the diagrammatic idea predates Euler, and is attributable to C. Weise or J. C. Lange.
- [2] Strategies for Reading Comprehension Venn Diagrams (<http://www.readingquest.org/strat/venn.html>)



Venn diagram as a truth table

References

Further reading

- A Survey of Venn Diagrams (<http://www.combinatorics.org/Surveys/ds5/VennEJC.html>) by F. Ruskey and M. Weston, is an extensive site with much recent research and many beautiful figures.
- Stewart, Ian (2004). "Ch. 4 Cogwheels of the Mind" (<http://books.google.com.au/books?id=u5GPE97-ZhsC&pg=PA51>). *Another Fine Math You've Got Me Into*. Dover Publications. pp. 51–64. ISBN 0-486-43181-9.
- Edwards, A.W.F. (2004). *Cogwheels of the mind: the story of Venn diagrams* (http://books.google.com/books?id=7_0Thy4V3JIC). JHU Press. ISBN 978-0-8018-7434-5.
- Venn, John (1880). "On the Diagrammatic and Mechanical Representation of Propositions and Reasonings". *Dublin Philosophical Magazine and Journal of Science* 9 (59): 1–18.
- Mamakani, Khalegh; Ruskey, Frank (27 July 2012), *A New Rose : The First Simple Symmetric 11-Venn Diagram* (<http://webhome.cs.uvic.ca/~ruskey/Publications/Venn11/Venn11.html>), arXiv: 1207.6452 (<http://arxiv.org/abs/1207.6452>)

External links

- Hazewinkel, Michiel, ed. (2001), "Venn diagram" (<http://www.encyclopediaofmath.org/index.php?title=p/v096550>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Weisstein, Eric W., " Venn Diagram (<http://mathworld.wolfram.com/VennDiagram.html>)", *MathWorld*.
- Free software for generating Venn and Euler diagrams using circles (<http://www.eulerdiagrams.com/inductivecircles.html>)
- Lewis Carroll's Logic Game – Venn vs. Euler (<http://www.cut-the-knot.org/LewisCarroll/dunham.shtml>) at cut-the-knot
- A Survey of Venn Diagrams (<http://www.combinatorics.org/Surveys/ds5/VennEJC.html>)
- Area proportional 3-way venn diagram applet (<http://www.cs.kent.ac.uk/people/staff/pjr/EulerVennCircles/EulerVennApplet.html>)
- Generating Venn Diagrams to explore Google Suggest results (<http://www.technomancy.org/google-suggest-venn/>)
- seven sets interactive Venn diagram displaying color combinations (<http://moebio.com/research/sevensets>)
- six sets Venn diagrams made from triangles (<http://www.combinatorics.org/Surveys/ds5/VennTriangleEJC.html>)
- Postscript for 9-set Venn (<http://qandr.org/quentin/software/venn>) and more
- VBVenn – A Visual Basic program for calculating and graphing quantitative two-circle Venn diagrams (<http://webdmamrl.er.usgs.gov/g1/FHWA/VBVenn/default.htm>)

Logical Connectives and functions

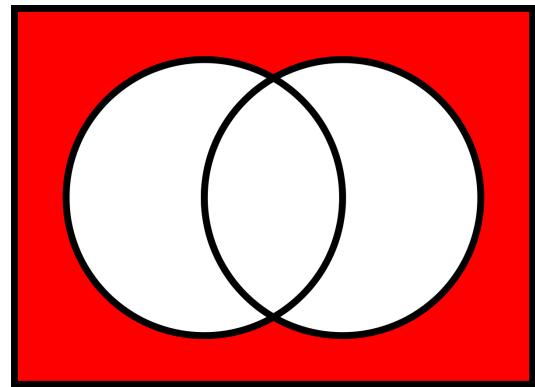
Ampheck

In boolean logic, **logical nor** or **joint denial** is a truth-functional operator which produces a result that is the negation of logical or. That is, a sentence of the form (p NOR q) is true precisely when neither p nor q is true—i.e. when both of p and q are *false*. In grammar, **nor** is a coordinating conjunction.

The NOR operator is also known as **Peirce's arrow** — Charles Sanders Peirce introduced the symbol \downarrow for it, and demonstrated that the logical NOR is completely expressible: by combining uses of the logical NOR it is possible to express any logical operation on two variables. Thus, as with its dual, the NAND operator (a.k.a. the Sheffer stroke — symbolized as either \mid or $/$), NOR can be used by itself, without any other logical operator, to constitute a logical formal system (making NOR functionally complete). It is also known as Quine's dagger (his symbol was \dagger), the **ampheck** (from Greek $\alpha\mu\phi\eta\kappa\eta\varsigma$, cutting both ways; compare *amphi-*) by Peirce,^[1] or "neither-nor".

One way of expressing p NOR q is $\overline{p \vee q}$, where the symbol \vee signifies OR and the bar signifies the negation of the expression under it: in essence, simply $\neg(p \vee q)$. Other ways of expressing p NOR q are Xpq , and $\overline{p + q}$.

The computer used in the spacecraft that first carried humans to the moon, the Apollo Guidance Computer, was constructed entirely using NOR gates with three inputs.^[citation needed]



Venn diagram of $A \downarrow B$
(nor part in red)

Definition

The **NOR operation** is a logical operation on two logical values, typically the values of two propositions, that produces a value of *true* if and only if both operands are *false*. In other words, it produces a value of *false* if and only if at least one operand is *true*.

Truth table

The truth table of **A NOR B** (also written as $A \downarrow B$) is as follows:

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

Properties

Logical NOR does not possess any of the five qualities (truth-preserving, false-preserving, linear, monotonic, self-dual) required to be absent from at least one member of a set of functionally complete operators. Thus, the set containing only NOR suffices as a complete set.

Introduction, elimination, and equivalencies

NOR has the interesting feature that all other logical operators can be expressed by interlaced NOR operations. The logical NAND operator also has this ability.

The logical NOR \downarrow is the negation of the disjunction:

$P \downarrow Q$	\Leftrightarrow	$\neg(P \vee Q)$
	\Leftrightarrow	

Expressed in terms of NOR \downarrow , the usual operators of propositional logic are:

$\neg P$	\Leftrightarrow	$P \downarrow P$
	\Leftrightarrow	
$P \rightarrow Q$	\Leftrightarrow	$((P \downarrow P) \downarrow Q)$
	\Leftrightarrow	
$P \vee Q$	\Leftrightarrow	$(P \downarrow Q)$
	\Leftrightarrow	
$P \wedge Q$	\Leftrightarrow	$(P \downarrow P) \downarrow (Q \downarrow Q)$
	\Leftrightarrow	

References

- [1] C.S. Peirce, CP 4.264

Balanced boolean function

In mathematics and computer science, a **balanced boolean function** is a boolean function whose output yields as many **0s** as **1s** over its input set. This means that for a uniformly random input string of bits, the probability of getting a **1** is 1/2.

An example of a balanced boolean function is the function that assigns a **1** to every even number and **0** to all odd numbers (likewise the other way around). The same applies for functions assigning **1** to all positive numbers and **0** otherwise.

A Boolean function of n bits is balanced if it takes the value 1 with probability 1/2.

Usage

Balanced boolean functions are primarily used in cryptography. If a function is not balanced, it will have a statistical bias, making it subject to cryptanalysis such as the correlation attack.

References

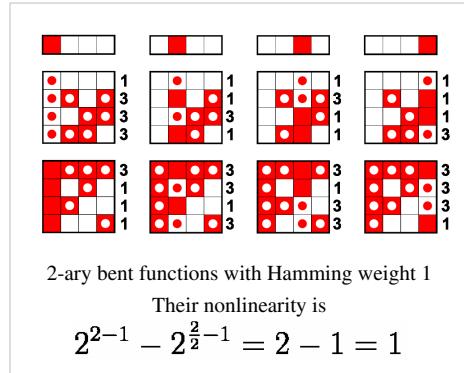
- Balanced boolean functions that can be evaluated so that every input bit is unlikely to be read^[1], Annual ACM Symposium on Theory of Computing

References

[1] <http://portal.acm.org/citation.cfm?id=1060627>

Bent function

In the mathematical field of combinatorics, a **bent function** is a special type of Boolean function. Defined and named in the 1960s by Oscar Rothaus in research not published until 1976, bent functions are so called because they are as different as possible from all linear and affine functions. They have been extensively studied for their applications in cryptography, but have also been applied to spread spectrum, coding theory, and combinatorial design. The definition can be extended in several ways, leading to different classes of generalized bent functions that share many of the useful properties of the original.



Walsh transform

Bent functions are defined in terms of the Walsh transform. The Walsh transform of a Boolean function $f: \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2$ is the function $\hat{f}: \mathbb{Z}_2^n \rightarrow \mathbb{Z}$ given by

$$\hat{f}(a) = \sum_{x \in \mathbb{Z}_2^n} (-1)^{f(x)+a \cdot x}$$

where $a \cdot x = a_1x_1 + a_2x_2 + \dots + a_nx_n \pmod{2}$ is the dot product in \mathbf{Z}_n

2. Alternatively, let $S_0(a) = \{ x \in \mathbf{Z}_n \mid f(x) = a \cdot x \}$

$2 : f(x) = a \cdot x \}$ and $S_1(a) = \{ x \in \mathbf{Z}_n \mid f(x) \neq a \cdot x \}$

$2 : f(x) \neq a \cdot x \}$. Then $|S_0(a)| + |S_1(a)| = 2^n$ and hence

$$\hat{f}(a) = |S_0(a)| - |S_1(a)| = 2|S_0(a)| - 2^n.$$

For any Boolean function f and $a \in \mathbf{Z}_n$

2 the transform lies in the range

$$-2^n \leq \hat{f}(a) \leq 2^n.$$

Moreover, the linear function $f_0(x) = a \cdot x$ and the affine function $f_1(x) = a \cdot x + 1$ correspond to the two extreme cases, since

$$\hat{f}_0(a) = 2^n, \quad \hat{f}_1(a) = -2^n.$$

Thus, for each $a \in \mathbf{Z}_n$

2 the value of $\hat{f}(a)$ characterizes where the function $f(x)$ lies in the range from $f_0(x)$ to $f_1(x)$.

Definition and properties

Rothaus defined a **bent function** as a Boolean function $f : \mathbf{Z}_n \rightarrow \mathbf{Z}_2$

2 → \mathbf{Z}_2 whose Walsh transform has constant absolute value. Bent functions are in a sense equidistant from all the affine functions, so they are equally hard to approximate with any affine function.

The simplest examples of bent functions, written in algebraic normal form, are $F(x_1, x_2) = x_1x_2$ and $G(x_1, x_2, x_3, x_4) = x_1x_2 + x_3x_4$. This pattern continues: $x_1x_2 + x_3x_4 + \dots + x_{n-1}x_n$ is a bent function \mathbf{Z}_n

2 → \mathbf{Z}_2 for every even n , but there is a wide variety of different types of bent functions as n increases. The sequence of values $(-1)^{f(x)}$, with $x \in \mathbf{Z}_n$

2 taken in lexicographical order, is called a **bent sequence**; bent functions and bent sequences have equivalent properties. In this ± 1 form, the Walsh transform is easily computed as

$$\hat{f}(a) = W(2^n)(-1)^{f(a)},$$

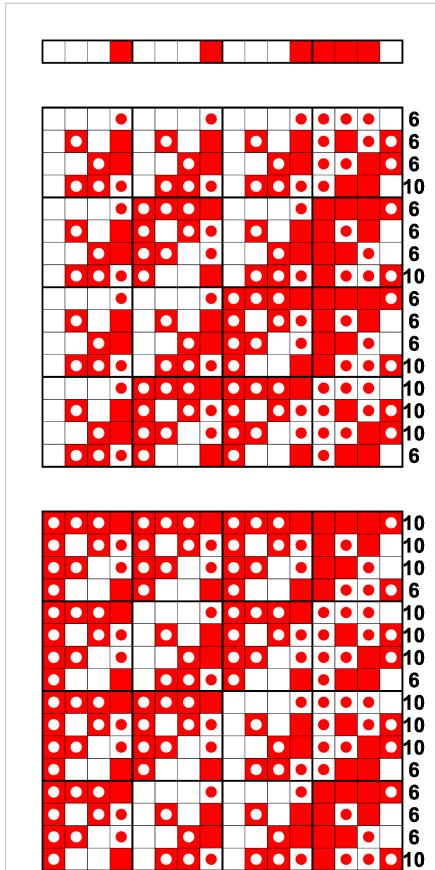
where $W(2^n)$ is the natural-ordered Walsh matrix and the sequence is treated as a column vector.

Rothaus proved that bent functions exist only for even n , and that for a bent function f , $|\hat{f}(a)| = 2^{n/2}$ for all $a \in \mathbf{Z}_n$

2. In fact, $\hat{f}(a) = 2^{n/2}(-1)^{g(a)}$, where g is also bent. In this case, $\hat{g}(a) = 2^{n/2}(-1)^{f(a)}$, so f and g are considered dual functions.

Every bent function has a Hamming weight (number of times it takes the value 1) of $2^{n-1} \pm 2^{n/2-1}$, and in fact agrees with any affine function at one of those two numbers of points. So the *nonlinearity* of f (minimum number of times it equals any affine function) is $2^{n-1} - 2^{n/2-1}$, the maximum possible. Conversely, any Boolean function with nonlinearity $2^{n-1} - 2^{n/2-1}$ is bent. The degree of f in algebraic normal form (called the *nonlinear order* of f) is at most $n/2$ (for $n > 2$).

Although bent functions are vanishingly rare among Boolean functions of many variables, they come in many different kinds. There has been detailed research into special classes of bent functions, such as the homogeneous ones or those arising from a monomial over a finite field, but so far the bent functions have defied all attempts at a complete enumeration or classification.



The nonlinearity of the bent function

$$x_1x_2 + x_3x_4 \text{ is } 2^{4-1} - 2^{\frac{4}{2}-1} = 8 - 2 = 6$$

+ stands for the exclusive or
(compare algebraic normal form)

Applications

As early as 1982 it was discovered that maximum length sequences based on bent functions have cross-correlation and autocorrelation properties rivalling those of the Gold codes and Kasami codes for use in CDMA. These sequences have several applications in spread spectrum techniques.

The properties of bent functions are naturally of interest in modern digital cryptography, which seeks to obscure relationships between input and output. By 1988 Forré recognized that the Walsh transform of a function can be used to show that it satisfies the Strict Avalanche Criterion (SAC) and higher-order generalizations, and recommended this tool to select candidates for good S-boxes achieving near-perfect diffusion. Indeed, the functions satisfying the SAC to the highest possible order are always bent. Furthermore, the bent functions are as far as possible from having what are called *linear structures*, nonzero vectors a such that $f(x+a) + f(x)$ is a constant. In the language of differential cryptanalysis (introduced after this property was discovered) the *derivative* of a bent function f at every nonzero point a (that is, $f_a(x) = f(x+a) + f(x)$) is a *balanced* Boolean function, taking on each value exactly half of the time. This property is called *perfect nonlinearity*.

Given such good diffusion properties, apparently perfect resistance to differential cryptanalysis, and resistance by definition to linear cryptanalysis, bent functions might at first seem the ideal choice for secure cryptographic functions such as S-boxes. Their fatal flaw is that they fail to be balanced. In particular, an invertible S-box cannot be constructed directly from bent functions, and a stream cipher using a bent combining function is vulnerable to a correlation attack. Instead, one might start with a bent function and randomly complement appropriate values until the result is balanced. The modified function still has high nonlinearity, and as such functions are very rare the process should be much faster than a brute-force search. But functions produced in this way may lose other desirable properties, even failing to satisfy the SAC—so careful testing is necessary. A number of cryptographers have worked on techniques for generating balanced functions that preserve as many of the good cryptographic qualities of bent functions as possible.

Some of this theoretical research has been incorporated into real cryptographic algorithms. The *CAST* design procedure, used by Carlisle Adams and Stafford Tavares to construct the S-boxes for the block ciphers CAST-128 and CAST-256, makes use of bent functions. The cryptographic hash function HAVAL uses Boolean functions built from representatives of all four of the equivalence classes of bent functions on six variables. The stream cipher Grain uses an NLFSR whose nonlinear feedback polynomial is, by design, the sum of a bent function and a linear function.

Generalizations

The most common class of *generalized bent functions* is the mod m type, $f : \mathbb{Z}_m^n \rightarrow \mathbb{Z}_m$ such that

$$\hat{f}(a) = \sum_{x \in \mathbb{Z}_m^n} e^{\frac{2\pi i}{m} (f(x) - a \cdot x)}$$

has constant absolute value $m^{n/2}$. Perfect nonlinear functions $f : \mathbb{Z}_m^n \rightarrow \mathbb{Z}_m$, those such that for all nonzero a , $f(x+a) - f(a)$ takes on each value m^{n-1} times, are generalized bent. If m is prime, the converse is true. In most cases only prime m are considered. For odd prime m , there are generalized bent functions for every positive n , even and odd. They have many of the same good cryptographic properties as the binary bent functions.

Semi-bent functions are an odd-order counterpart to bent functions. A semi-bent function is $f : \mathbb{Z}_m^n \rightarrow \mathbb{Z}_m$ with n odd, such that $|\hat{f}|$ takes only the values 0 and $m^{(n+1)/2}$. They also have good cryptographic characteristics, and some of them are balanced, taking on all possible values equally often.

The **partially bent functions** form a large class defined by a condition on the Walsh transform and autocorrelation functions. All affine and bent functions are partially bent. This is in turn a proper subclass of the *plateaued functions*.

The idea behind the **hyper-bent functions** is to maximize the minimum distance to *all* Boolean functions coming from bijective monomials on the finite field $GF(2^n)$, not just the affine functions. For these functions this distance is constant, which may make them resistant to an interpolation attack.

Other related names have been given to cryptographically important classes of functions \mathbb{Z}_n

$2 \rightarrow \mathbb{Z}_n$

2, such as **almost bent functions** and **crooked functions**. While not Boolean functions themselves, these are closely related to the bent functions and have good nonlinearity properties.

References

Further reading

- C. Carlet (May 1993). "Two New Classes of Bent Functions". Eurocrypt '93. pp. 77–101.
- J. Seberry; X. Zhang (March 1994). "Constructions of Bent Functions from Two Known Bent Functions". *Australasian Journal of Combinatorics* 9: 21–35. ISSN 1034-4942 (<http://www.worldcat.org/issn/1034-4942>). CiteSeerX: 10.1.1.55.531 (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.531>).
- T. Neumann; advisor: U. Dempwolff (May 2006). *Bent Functions*. CiteSeerX: 10.1.1.85.8731 (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.85.8731>).
- Colbourn, Charles J.; Dinitz, Jeffrey H. (2006). *Handbook of Combinatorial Designs* (2nd ed.). CRC Press. pp. 337–339. ISBN 978-1-58488-506-1.

Boolean algebras canonically defined

Boolean algebras have been formally defined variously as a kind of lattice and as a kind of ring. This article presents them, equally formally, as simply the models of the equational theory of two values, and observes the equivalence of both the lattice and ring definitions to this more elementary one.

Boolean algebra is a mathematically rich branch of abstract algebra. Just as group theory deals with groups, and linear algebra with vector spaces, Boolean algebras are models of the equational theory of the two values 0 and 1 (whose interpretation need not be numerical). Common to Boolean algebras, groups, and vector spaces is the notion of an algebraic structure, a set closed under zero or more operations satisfying certain equations.

Just as there are basic examples of groups, such as the group \mathbb{Z} of integers and the permutation group S_n of permutations of n objects, there are also basic examples of Boolean algebra such as the following.

- The algebra of binary digits or bits 0 and 1 under the logical operations including disjunction, conjunction, and negation. Applications include the propositional calculus and the theory of digital circuits.
- The algebra of sets under the set operations including union, intersection, and complement. Applications include any area of mathematics for which sets form a natural foundation.

Boolean algebra thus permits applying the methods of abstract algebra to mathematical logic, digital logic, and the set-theoretic foundations of mathematics.

Unlike groups of finite order, which exhibit complexity and diversity and whose first-order theory is decidable only in special cases, all finite Boolean algebras share the same theorems and have a decidable first-order theory. Instead the intricacies of Boolean algebra are divided between the structure of infinite algebras and the algorithmic complexity of their syntactic structure.

Definition

Boolean algebra treats the equational theory of the maximal two-element finitary algebra, called the **Boolean prototype**, and the models of that theory, called **Boolean algebras**. These terms are defined as follows.

An algebra is a family of operations on a set, called the underlying set of the algebra. We take the underlying set of the Boolean prototype to be $\{0,1\}$.

An algebra is **finitary** when each of its operations takes only finitely many arguments. For the prototype each argument of an operation is either 0 or 1, as is the result of the operation. The maximal such algebra consists of all finitary operations on $\{0,1\}$.

The number of arguments taken by each operation is called the arity of the operation. An operation on $\{0,1\}$ of arity n , or n -ary operation, can be applied to any of 2^n possible values for its n arguments. For each choice of arguments the operation may return 0 or 1, whence there are 2^2n n -ary operations.

The prototype therefore has two operations taking no arguments, called zeroary or **nullary** operations, namely zero and one. It has four unary operations, two of which are constant operations, another is the identity, and the most commonly used one, called *negation*, returns the opposite of its argument: 1 if 0, 0 if 1. It has sixteen binary operations; again two of these are constant, another returns its first argument, yet another returns its second, one is called *conjunction* and returns 1 if both arguments are 1 and otherwise 0, another is called *disjunction* and returns 0 if both arguments are 0 and otherwise 1, and so on. The number of $(n+1)$ -ary operations in the prototype is the square of the number of n -ary operations, so there are $16^2 = 256$ ternary operations, $256^2 = 65,536$ quaternary operations, and so on.

A family is indexed by an index set. In the case of a family of operations forming an algebra, the indices are called **operation symbols**, constituting the **language** of that algebra. The operation indexed by each symbol is called the denotation or **interpretation** of that symbol. Each operation symbol specifies the arity of its interpretation, whence all possible interpretations of a symbol have the same arity. In general it is possible for an algebra to interpret distinct symbols with the same operation, but this is not the case for the prototype, whose symbols are in one-one correspondence with its operations. The prototype therefore has 2^2n n -ary operation symbols, called the **Boolean operation symbols** and forming the language of Boolean algebra. Only a few operations have conventional symbols, such as \neg for negation, \wedge for conjunction, and \vee for disjunction. It is convenient to consider the i -th n -ary symbol to be nf_i as done below in the section on truth tables.

An equational theory in a given language consists of equations between terms built up from variables using symbols of that language. Typical equations in the language of Boolean algebra are $x \wedge y = y \wedge x$, $x \wedge x = x$, $x \wedge \neg x = y \wedge \neg y$, and $x \wedge y = x$.

An algebra **satisfies** an equation when the equation holds for all possible values of its variables in that algebra when the operation symbols are interpreted as specified by that algebra. The laws of Boolean algebra are the equations in the language of Boolean algebra satisfied by the prototype. The first three of the above examples are Boolean laws, but not the fourth since $1 \wedge 0 \neq 1$.

The equational theory of an algebra is the set of all equations satisfied by the algebra. The laws of Boolean algebra therefore constitute the equational theory of the Boolean prototype.

A model of a theory is an algebra interpreting the operation symbols in the language of the theory and satisfying the equations of the theory.

A Boolean algebra is any model of the laws of Boolean algebra.

That is, a Boolean algebra is a set and a family of operations thereon interpreting the Boolean operation symbols and satisfying the same laws as the Boolean prototype.

If we define a homologue of an algebra to be a model of the equational theory of that algebra, then a Boolean algebra can be defined as any homologue of the prototype.

Example 1. The Boolean prototype is a Boolean algebra, since trivially it satisfies its own laws. It is thus the prototypical Boolean algebra. We did not call it that initially in order to avoid any appearance of circularity in the definition.

Basis

The operations need not be all explicitly stated. A *basis* is any set from which the remaining operations can be obtained by composition. A "Boolean algebra" may be defined from any of several different bases. Three bases for Boolean algebra are in common use, the lattice basis, the ring basis, and the Sheffer stroke or NAND basis. These bases impart respectively a logical, an arithmetical, and a parsimonious character to the subject.

- The lattice basis originated in the 19th century with the work of Boole, Peirce, and others seeking an algebraic formalization of logical thought processes.
- The ring basis emerged in the 20th century with the work of Zhegalkin and Stone and became the basis of choice for algebraists coming to the subject from a background in abstract algebra. Most treatments of Boolean algebra assume the lattice basis, a notable exception being Halmos[1963] whose linear algebra background evidently endeared the ring basis to him.
- Since all finitary operations on {0,1} can be defined in terms of the Sheffer stroke NAND (or its dual NOR), the resulting economical basis has become the basis of choice for analyzing digital circuits, in particular gate arrays in digital electronics.

The common elements of the lattice and ring bases are the constants 0 and 1, and an associative commutative binary operation, called meet $x \wedge y$ in the lattice basis, and multiplication xy in the ring basis. The distinction is only terminological. The lattice basis has the further operations of join, $x \vee y$, and complement, $\neg x$. The ring basis has instead the arithmetic operation $x \oplus y$ of addition (the symbol \oplus is used in preference to $+$ because the latter is sometimes given the Boolean reading of join).

To be a basis is to yield all other operations by composition, whence any two bases must be intertranslatable. The lattice basis translates $x \vee y$ to the ring basis as $x \oplus y \oplus xy$, and $\neg x$ as $x \oplus 1$. Conversely the ring basis translates $x \oplus y$ to the lattice basis as $(x \vee y) \wedge \neg(x \wedge y)$.

Both of these bases allow Boolean algebras to be defined via a subset of the equational properties of the Boolean operations. For the lattice basis, it suffices to define a Boolean algebra as a distributive lattice satisfying $x \wedge \neg x = 0$ and $x \vee \neg x = 1$, called a complemented distributive lattice. The ring basis turns a Boolean algebra into a Boolean ring, namely a ring satisfying $x^2 = x$.

Emil Post gave a necessary and sufficient condition for a set of operations to be a basis for the nonzeroary Boolean operations. A *nontrivial* property is one shared by some but not all operations making up a basis. Post listed five nontrivial properties of operations, identifiable with the five Post's classes, each preserved by composition, and showed that a set of operations formed a basis if, for each property, the set contained an operation lacking that property. (The converse of Post's theorem, extending "if" to "if and only if," is the easy observation that a property from among these five holding of every operation in a candidate basis will also hold of every operation formed by composition from that candidate, whence by nontriviality of that property the candidate will fail to be a basis.) Post's five properties are:

- monotone, no 0-1 input transition can cause a 1-0 output transition;
- affine, representable with Zhegalkin polynomials that lack bilinear or higher terms, e.g. $x \oplus y \oplus 1$ but not xy ;
- self-dual, so that complementing all inputs complements the output, as with x , or the median operator $xy \oplus yz \oplus zx$, or their negations;
- strict (mapping the all-zeros input to zero);
- costrict (mapping all-ones to one).

The NAND (dually NOR) operation lacks all these, thus forming a basis by itself.

Truth tables

The finitary operations on $\{0,1\}$ may be exhibited as truth tables, thinking of 0 and 1 as the truth values **false** and **true**. They can be laid out in a uniform and application-independent way that allows us to name, or at least number, them individually. These names provide a convenient shorthand for the Boolean operations. The names of the n -ary operations are binary numbers of 2^n bits. There being 2^2n such operations, one cannot ask for a more succinct nomenclature! Note that each finitary operation can be called a switching function.

This layout and associated naming of operations is illustrated here in full for arities from 0 to 2.

Truth tables for the Boolean operations of arity up to 2

Constants

0f_0	0f_1
0	1

Unary Operations

x_0	1f_0	1f_1	1f_2	1f_3
0	0	1	0	1
1	0	0	1	1

Binary Operations

x_0	x_1	2f_0	2f_1	2f_2	2f_3	2f_4	2f_5	2f_6	2f_7	2f_8	2f_9	${}^2f_{10}$	${}^2f_{11}$	${}^2f_{12}$	${}^2f_{13}$	${}^2f_{14}$	${}^2f_{15}$
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

These tables continue at higher arities, with 2^n rows at arity n , each row giving a valuation or binding of the n variables x_0, \dots, x_{n-1} and each column headed nf_i giving the value ${}^nf_i(x_0, \dots, x_{n-1})$ of the i -th n -ary operation at that valuation. The operations include the variables, for example 1f_2 is x_0 while ${}^2f_{10}$ is x_0 (as two copies of its unary counterpart) and ${}^2f_{12}$ is x_1 (with no unary counterpart). Negation or complement $\neg x_0$ appears as 1f_1 and again as 2f_5 , along with ${}^2f_3(\neg x_1)$, which did not appear at arity 1), disjunction or union $x_0 \vee x_1$ as ${}^2f_{14}$, conjunction or intersection $x_0 \wedge x_1$ as 2f_8 , implication $x_0 \rightarrow x_1$ as ${}^2f_{13}$, exclusive-or symmetric difference $x_0 \oplus x_1$ as 2f_6 , set difference $x_0 - x_1$ as 2f_2 , and so on.

As a minor detail important more for its form than its content, the operations of an algebra are traditionally organized as a list. Although we are here indexing the operations of a Boolean algebra by the finitary operations on $\{0,1\}$, the truth-table presentation above serendipitously orders the operations first by arity and second by the layout of the tables for each arity. This permits organizing the set of all Boolean operations in the traditional list format. The list order for the operations of a given arity is determined by the following two rules.

- (i) The i -th row in the left half of the table is the binary representation of i with its least significant or 0-th bit on the left ("little-endian" order, originally proposed by Alan Turing, so it would not be unreasonable to call it Turing order).

(ii) The j -th column in the right half of the table is the binary representation of j , again in little-endian order. In effect the subscript of the operation *is* the truth table of that operation. By analogy with Gödel numbering of computable functions one might call this numbering of the Boolean operations the Boole numbering.

When programming in C or Java, bitwise disjunction is denoted $x|y$, conjunction $x\&y$, and negation $\sim x$. A program can therefore represent for example the operation $x\wedge(y\vee z)$ in these languages as $x\&(y|z)$, having previously set $x = 0xaa$, $y = 0xcc$, and $z = 0xf0$ (the "0x" indicates that the following constant is to be read in hexadecimal or base 16), either by assignment to variables or defined as macros. These one-byte (eight-bit) constants correspond to the columns for the input variables in the extension of the above tables to three variables. This technique is almost universally used in raster graphics hardware to provide a flexible variety of ways of combining and masking images, the typical operations being ternary and acting simultaneously on source, destination, and mask bits.

Examples

Bit vectors

Example 2. All bit vectors of a given length form a Boolean algebra "pointwise", meaning that any n -ary Boolean operation can be applied to n bit vectors one bit position at a time. For example the ternary OR of three bit vectors each of length 4 is the bit vector of length 4 formed by oring the three bits in each of the four bit positions, thus $0100\vee1000\vee1001 = 1101$. Another example is the truth tables above for the n -ary operations, whose columns are all the bit vectors of length 2^n and which therefore can be combined pointwise whence the n -ary operations form a Boolean algebra. This works equally well for bit vectors of finite and infinite length, the only rule being that the bit positions all be indexed by the same set in order that "corresponding position" be well defined.

The **atoms** of such an algebra are the bit vectors containing exactly one 1. In general the atoms of a Boolean algebra are those elements x such that $x\wedge y$ has only two possible values, x or 0.

Power set algebra

Example 3. The **power set algebra**, the set 2^W of all subsets of a given set W . This is just Example 2 in disguise, with W serving to index the bit positions. Any subset X of W can be viewed as the bit vector having 1's in just those bit positions indexed by elements of X . Thus the all-zero vector is the empty subset of W while the all-ones vector is W itself, these being the constants 0 and 1 respectively of the power set algebra. The counterpart of disjunction $x\vee y$ is union $X\cup Y$, while that of conjunction $x\wedge y$ is intersection $X\cap Y$. Negation $\neg x$ becomes $\sim X$, complement relative to W . There is also set difference $X\setminus Y = X\cap\sim Y$, symmetric difference $(X\setminus Y)\cup(Y\setminus X)$, ternary union $X\cup Y\cup Z$, and so on. The atoms here are the singletons, those subsets with exactly one element.

Examples 2 and 3 are special cases of a general construct of algebra called direct product, applicable not just to Boolean algebras but all kinds of algebra including groups, rings, etc. The direct product of any family B_i of Boolean algebras where i ranges over some index set I (not necessarily finite or even countable) is a Boolean algebra consisting of all I -tuples $(\dots x_i, \dots)$ whose i -th element is taken from B_i . The operations of a direct product are the corresponding operations of the constituent algebras acting within their respective coordinates; in particular operation ${}^n f_j$ of the product operates on n I -tuples by applying operation ${}^n f_j$ of B_i to the n elements in the i -th coordinate of the n tuples, for all i in I .

When all the algebras being multiplied together in this way are the same algebra A we call the direct product a *direct power* of A . The Boolean algebra of all 32-bit bit vectors is the two-element Boolean algebra raised to the 32nd power, or power set algebra of a 32-element set, denoted 2^{32} . The Boolean algebra of all sets of integers is $2^{\mathbb{Z}}$. All Boolean algebras we have exhibited thus far have been direct powers of the two-element Boolean algebra, justifying the name "power set algebra".

Representation theorems

It can be shown that every finite Boolean algebra is isomorphic to some power set algebra. Hence the cardinality (number of elements) of a finite Boolean algebra is a power of 2, namely one of $1, 2, 4, 8, \dots, 2^n, \dots$. This is called a **representation theorem** as it gives insight into the nature of finite Boolean algebras by giving a representation of them as power set algebras.

This representation theorem does not extend to infinite Boolean algebras: although every power set algebra is a Boolean algebra, not every Boolean algebra need be isomorphic to a power set algebra. In particular, whereas there can be no countably infinite power set algebras (the smallest infinite power set algebra is the power set algebra $2^{\mathbb{N}}$ of sets of natural numbers, shown by Cantor to be uncountable), there exist various countably infinite Boolean algebras.

To go beyond power set algebras we need another construct. A subalgebra of an algebra A is any subset of A closed under the operations of A . Every subalgebra of a Boolean algebra A must still satisfy the equations holding of A , since any violation would constitute a violation for A itself. Hence every subalgebra of a Boolean algebra is a Boolean algebra.

A subalgebra of a power set algebra is called a field of sets; equivalently a field of sets is a set of subsets of some set W including the empty set and W and closed under finite union and complement with respect to W (and hence also under finite intersection). Birkhoff's [1935] representation theorem for Boolean algebras states that every Boolean algebra is isomorphic to a field of sets. Now Birkhoff's HSP theorem for varieties can be stated as, every class of models of the equational theory of a class C of algebras is the Homomorphic image of a Subalgebra of a direct Product of algebras of C . Normally all three of H, S, and P are needed; what the first of these two Birkhoff theorems shows is that for the special case of the variety of Boolean algebras Homomorphism can be replaced by Isomorphism. Birkhoff's HSP theorem for varieties in general therefore becomes Birkhoff's ISP theorem for the variety of Boolean algebras.

Other examples

It is convenient when talking about a set X of natural numbers to view it as a sequence x_0, x_1, x_2, \dots of bits, with $x_i = 1$ if and only if $i \in X$. This viewpoint will make it easier to talk about subalgebras of the power set algebra $2^{\mathbb{N}}$, which this viewpoint makes the Boolean algebra of all sequences of bits. It also fits well with the columns of a truth table: when a column is read from top to bottom it constitutes a sequence of bits, but at the same time it can be viewed as the set of those valuations (assignments to variables in the left half of the table) at which the function represented by that column evaluates to 1.

Example 4. *Ultimately constant sequences.* Any Boolean combination of ultimately constant sequences is ultimately constant; hence these form a Boolean algebra. We can identify these with the integers by viewing the ultimately-zero sequences as nonnegative binary numerals (bit 0 of the sequence being the low-order bit) and the ultimately-one sequences as negative binary numerals (think two's complement arithmetic with the all-ones sequence being -1). This makes the integers a Boolean algebra, with union being bit-wise OR and complement being $\neg x - 1$. There are only countably many integers, so this infinite Boolean algebra is countable. The atoms are the powers of two, namely $1, 2, 4, \dots$. Another way of describing this algebra is as the set of all finite and cofinite sets of natural numbers, with the ultimately all-ones sequences corresponding to the cofinite sets, those sets omitting only finitely many natural numbers.

Example 5. *Periodic sequence.* A sequence is called *periodic* when there exists some number $n > 0$, called a witness to periodicity, such that $x_i = x_{i+n}$ for all $i \geq 0$. The period of a periodic sequence is its least witness. Negation leaves period unchanged, while the disjunction of two periodic sequences is periodic, with period at most the least common multiple of the periods of the two arguments (the period can be as small as 1, as happens with the union of any sequence and its complement). Hence the periodic sequences form a Boolean algebra.

Example 5 resembles Example 4 in being countable, but differs in being atomless. The latter is because the conjunction of any nonzero periodic sequence x with a sequence of greater period is neither 0 nor x . It can be shown that all countably infinite atomless Boolean algebras are isomorphic, that is, up to isomorphism there is only one such algebra.

Example 6. *Periodic sequence with period a power of two.* This is a proper subalgebra of Example 5 (a proper subalgebra equals the intersection of itself with its algebra). These can be understood as the finitary operations, with the first period of such a sequence giving the truth table of the operation it represents. For example the truth table of x_0 in the table of binary operations, namely ${}^2f_{10}$, has period 2 (and so can be recognized as using only the first variable) even though 12 of the binary operations have period 4. When the period is 2^n the operation only depends on the first n variables, the sense in which the operation is finitary. This example is also a countably infinite atomless Boolean algebra. Hence Example 5 is isomorphic to a proper subalgebra of itself! Example 6, and hence Example 5, constitutes the free Boolean algebra on countably many generators, meaning the Boolean algebra of all finitary operations on a countably infinite set of generators or variables.

Example 7. *Ultimately periodic sequences*, sequences that become periodic after an initial finite bout of lawlessness. They constitute a proper extension of Example 5 (meaning that Example 5 is a proper subalgebra of Example 7) and also of Example 4, since constant sequences are periodic with period one. Sequences may vary as to when they settle down, but any finite set of sequences will all eventually settle down no later than their slowest-to-settle member, whence ultimately periodic sequences are closed under all Boolean operations and so form a Boolean algebra. This example has the same atoms and coatoms as Example 4, whence it is not atomless and therefore not isomorphic to Example 5/6. However it contains an infinite atomless subalgebra, namely Example 5, and so is not isomorphic to Example 4, every subalgebra of which must be a Boolean algebra of finite sets and their complements and therefore atomic. This example is isomorphic to the direct product of Examples 4 and 5, furnishing another description of it.

Example 8. The direct product of a Periodic Sequence (Example 5) with any finite but nontrivial Boolean algebra. (The trivial one-element Boolean algebra is the unique finite atomless Boolean algebra.) This resembles Example 7 in having both atoms and an atomless subalgebra, but differs in having only finitely many atoms. Example 8 is in fact an infinite family of examples, one for each possible finite number of atoms.

These examples by no means exhaust the possible Boolean algebras, even the countable ones. Indeed there are uncountably many nonisomorphic countable Boolean algebras, which Jussi Ketonen [1978] classified completely in terms of invariants representable by certain hereditarily countable sets.

Boolean algebras of Boolean operations

The n -ary Boolean operations themselves constitute a power set algebra 2^W , namely when W is taken to be the set of 2^n valuations of the n inputs. In terms of the naming system of operations ${}^n f_i$ where i in binary is a column of a truth table, the columns can be combined with Boolean operations of any arity to produce other columns present in the table. That is, we can apply any Boolean operation of arity m to m Boolean operations of arity n to yield a Boolean operation of arity n , for any m and n .

The practical significance of this convention for both software and hardware is that n -ary Boolean operations can be represented as words of the appropriate length. For example each of the 256 ternary Boolean operations can be represented as an unsigned byte. The available logical operations such as AND and OR can then be used to form new operations. If we take x , y , and z (dispensing with subscripted variables for now) to be 10101010, 11001100, and 11110000 respectively (170, 204, and 240 in decimal, 0xaa, 0xcc, and 0xf0 in hexadecimal), their pairwise conjunctions are $x \wedge y = 10001000$, $y \wedge z = 11000000$, and $z \wedge x = 10100000$, while their pairwise disjunctions are $x \vee y = 11101110$, $y \vee z = 11111100$, and $z \vee x = 11111010$. The disjunction of the three conjunctions is 11101000, which also happens to be the conjunction of three disjunctions. We have thus calculated, with a dozen or so logical operations on bytes, that the two ternary operations

$$(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$$

and

$$(x \vee y) \wedge (y \vee z) \wedge (z \vee x)$$

are actually the same operation. That is, we have proved the equational identity

$$(x \wedge y) \vee (y \wedge z) \vee (z \wedge x) = (x \vee y) \wedge (y \vee z) \wedge (z \vee x),$$

for the two-element Boolean algebra. By the definition of "Boolean algebra" this identity must therefore hold in every Boolean algebra.

This ternary operation incidentally formed the basis for Grau's [1947] ternary Boolean algebras, which he axiomatized in terms of this operation and negation. The operation is symmetric, meaning that its value is independent of any of the $3! = 6$ permutations of its arguments. The two halves of its truth table 11101000 are the truth tables for \vee , 1110, and \wedge , 1000, so the operation can be phrased as **if** z **then** $x \vee y$ **else** $x \wedge y$. Since it is symmetric it can equally well be phrased as either of **if** x **then** $y \vee z$ **else** $y \wedge z$, or **if** y **then** $z \vee x$ **else** $z \wedge x$. Viewed as a labeling of the 8-vertex 3-cube, the upper half is labeled 1 and the lower half 0; for this reason it has been called the median operator, with the evident generalization to any odd number of variables (odd in order to avoid the tie when exactly half the variables are 0).

Axiomatizing Boolean algebras

The technique we just used to prove an identity of Boolean algebra can be generalized to all identities in a systematic way that can be taken as a sound and complete axiomatization of, or axiomatic system for, the equational laws of Boolean logic. The customary formulation of an axiom system consists of a set of axioms that "prime the pump" with some initial identities, along with a set of inference rules for inferring the remaining identities from the axioms and previously proved identities. In principle it is desirable to have finitely many axioms; however as a practical matter it is not necessary since it is just as effective to have a finite axiom schema having infinitely many instances each of which when used in a proof can readily be verified to be a legal instance, the approach we follow here.

Boolean identities are assertions of the form $s = t$ where s and t are n -ary terms, by which we shall mean here terms whose variables are limited to x_0 through x_{n-1} . An n -ary **term** is either an atom or an application. An application ${}^m f_i(t_0, \dots, t_{m-1})$ is a pair consisting of an m -ary operation ${}^m f_i$ and a list or m -tuple (t_0, \dots, t_{m-1}) of m n -ary terms called **operands**.

Associated with every term is a natural number called its **height**. Atoms are of zero height, while applications are of height one plus the height of their highest operand.

Now what is an atom? Conventionally an atom is either a constant (0 or 1) or a variable x_i where $0 \leq i < n$. For the proof technique here it is convenient to define atoms instead to be n -ary operations ${}^n f_i$, which although treated here as atoms nevertheless mean the same as ordinary terms of the exact form ${}^n f_i(x_0, \dots, x_{n-1})$ (exact in that the variables must listed in the order shown without repetition or omission). This is not a restriction because atoms of this form include all the ordinary atoms, namely the constants 0 and 1, which arise here as the n -ary operations ${}^n f_0$ and ${}^n f_{-1}$ for each n (abbreviating $2^n - 1$ to -1), and the variables x_0, \dots, x_{n-1} as can be seen from the truth tables where x_0 appears as both the unary operation ${}^1 f_2$ and the binary operation ${}^2 f_{10}$ while x_1 appears as ${}^2 f_{12}$.

The following axiom schema and three inference rules axiomatize the Boolean algebra of n -ary terms.

A1. ${}^m f_i({}^n f_j 0, \dots, {}^n f_j m-1) = {}^n f_{i \hat{j}}$ where $(i \hat{j})_v = i_v$, with \hat{j} being j transpose, defined by $(\hat{j}_v)_u = (j_u)_v$.

R1. With no premises infer $t = t$.

R2. From $s = u$ and $t = u$ infer $s = t$ where s , t , and u are n -ary terms.

R3. From $s_0 = t_0, \dots, s_{m-1} = t_{m-1}$ infer ${}^m f_i(s_0, \dots, s_{m-1}) = {}^m f_i(t_0, \dots, t_{m-1})$, where all terms s_i , t_i are n -ary.

The meaning of the side condition on **A1** is that $i \hat{j}$ is that 2^n -bit number whose v -th bit is the \hat{j}_v -th bit of i , where the ranges of each quantity are u : m , v : 2^n , j_u : 2^n , and \hat{j}_v : 2^m . (So j is an m -tuple of 2^n -bit numbers while \hat{j} as the transpose of j is a 2^n -tuple of m -bit numbers. Both j and \hat{j} therefore contain $m2^n$ bits.)

A1 is an axiom schema rather than an axiom by virtue of containing **metavariables**, namely m , i , n , and j_0 through j_{m-1} . The actual axioms of the axiomatization are obtained by setting the metavariables to specific values. For example if we take $m = n = i = j_0 = 1$, we can compute the two bits of $io\hat{f}$ from $i_1 = 0$ and $i_0 = 1$, so $io\hat{f} = 2$ (or 10 when written as a two-bit number). The resulting instance, namely ${}^1f_1({}^1f_1) = {}^1f_2$, expresses the familiar axiom $\neg\neg x = x$ of double negation. Rule **R3** then allows us to infer $\neg\neg\neg x = \neg x$ by taking s_0 to be ${}^1f_1({}^1f_1)$ or $\neg\neg x_0$, t_0 to be 1f_2 or x_0 , and ${}^m f_i$ to be 1f_1 or \neg .

For each m and n there are only finitely many axioms instantiating **A1**, namely $2^2m \times (2^2n)^m$. Each instance is specified by $2^m + m2^n$ bits.

We treat **R1** as an inference rule, even though it is like an axiom in having no premises, because it is a domain-independent rule along with **R2** and **R3** common to all equational axiomatizations, whether of groups, rings, or any other variety. The only entity specific to Boolean algebras is axiom schema **A1**. In this way when talking about different equational theories we can push the rules to one side as being independent of the particular theories, and confine attention to the axioms as the only part of the axiom system characterizing the particular equational theory at hand.

This axiomatization is complete, meaning that every Boolean law $s = t$ is provable in this system. One first shows by induction on the height of s that every Boolean law for which t is atomic is provable, using **R1** for the base case (since distinct atoms are never equal) and **A1** and **R3** for the induction step (s an application). This proof strategy amounts to a recursive procedure for evaluating s to yield an atom. Then to prove $s = t$ in the general case when t may be an application, use the fact that if $s = t$ is an identity then s and t must evaluate to the same atom, call it u . So first prove $s = u$ and $t = u$ as above, that is, evaluate s and t using **A1**, **R1**, and **R3**, and then invoke **R2** to infer $s = t$.

In **A1**, if we view the number n^m as the function type $m \rightarrow n$, and m_n as the application $m(n)$, we can reinterpret the numbers i , j , \hat{j} , and $io\hat{f}$ as functions of type $i: (m \rightarrow 2) \rightarrow 2$, $j: m \rightarrow ((n \rightarrow 2) \rightarrow 2)$, $\hat{j}: (n \rightarrow 2) \rightarrow (m \rightarrow 2)$, and $io\hat{f}: (n \rightarrow 2) \rightarrow 2$. The definition $(io\hat{f})_v = i_{\hat{j}(v)}$ in **A1** then translates to $(io\hat{f})(v) = i(\hat{j}(v))$, that is, $io\hat{f}$ is defined to be composition of i and \hat{j} understood as functions. So the content of **A1** amounts to defining term application to be essentially composition, modulo the need to transpose the m -tuple j to make the types match up suitably for composition. This composition is the one in Lawvere's previously mentioned category of power sets and their functions. In this way we have translated the commuting diagrams of that category, as the equational theory of Boolean algebras, into the equational consequences of **A1** as the logical representation of that particular composition law.

Underlying lattice structure

Underlying every Boolean algebra B is a partially ordered set or **poset** (B, \leq) . The **partial order** relation is defined by $x \leq y$ just when $x = x \wedge y$, or equivalently when $y = x \vee y$. Given a set X of elements of a Boolean algebra, an **upper bound** on X is an element y such that for every element x of X , $x \leq y$, while a lower bound on X is an element y such that for every element x of X , $y \leq x$.

A **sup** (supremum) of X is a least upper bound on X , namely an upper bound on X that is less or equal to every upper bound on X . Dually an **inf** (infimum) of X is a greatest lower bound on X . The sup of x and y always exists in the underlying poset of a Boolean algebra, being $x \vee y$, and likewise their inf exists, namely $x \wedge y$. The empty sup is 0 (the bottom element) and the empty inf is 1 (top). It follows that every finite set has both a sup and an inf. Infinite subsets of a Boolean algebra may or may not have a sup and/or an inf; in a power set algebra they always do.

Any poset (B, \leq) such that every pair x, y of elements has both a sup and an inf is called a **lattice**. We write $x \vee y$ for the sup and $x \wedge y$ for the inf. The underlying poset of a Boolean algebra always forms a lattice. The lattice is said to be **distributive** when $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$, or equivalently when $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$, since either law implies the other in a lattice. These are laws of Boolean algebra whence the underlying poset of a Boolean algebra forms a distributive lattice.

Given a lattice with a bottom element 0 and a top element 1, a pair x, y of elements is called **complementary** when $x \wedge y = 0$ and $x \vee y = 1$, and we then say that y is a complement of x and vice versa. Any element x of a distributive lattice with top and bottom can have at most one complement. When every element of a lattice has a complement the lattice is called complemented. It follows that in a complemented distributive lattice, the complement of an element always exists and is unique, making complement a unary operation. Furthermore every complemented distributive lattice forms a Boolean algebra, and conversely every Boolean algebra forms a complemented distributive lattice. This provides an alternative definition of a Boolean algebra, namely as any complemented distributive lattice. Each of these three properties can be axiomatized with finitely many equations, whence these equations taken together constitute a finite axiomatization of the equational theory of Boolean algebras.

In a class of algebras defined as all the models of a set of equations, it is usually the case that some algebras of the class satisfy more equations than just those needed to qualify them for the class. The class of Boolean algebras is unusual in that, with a single exception, every Boolean algebra satisfies exactly the Boolean identities and no more. The exception is the one-element Boolean algebra, which necessarily satisfies every equation, even $x = y$, and is therefore sometimes referred to as the inconsistent Boolean algebra.

Boolean homomorphisms

A Boolean homomorphism is a function $h: A \rightarrow B$ between Boolean algebras A, B such that for every Boolean operation ${}^m f_i$,

$$h({}^m f_i(x_0, \dots, x_{m-1})) = {}^m f_i(h(x_0), \dots, h(x_{m-1})).$$

The category **Bool** of Boolean algebras has as objects all Boolean algebras and as morphisms the Boolean homomorphisms between them.

There exists a unique homomorphism from the two-element Boolean algebra **2** to every Boolean algebra, since homomorphisms must preserve the two constants and those are the only elements of **2**. A Boolean algebra with this property is called an **initial** Boolean algebra. It can be shown that any two initial Boolean algebras are isomorphic, so up to isomorphism **2** is *the* initial Boolean algebra.

In the other direction, there may exist many homomorphisms from a Boolean algebra B to **2**. Any such homomorphism partitions B into those elements mapped to 1 and those to 0. The subset of B consisting of the former is called an ultrafilter of B . When B is finite its ultrafilters pair up with its atoms; one atom is mapped to 1 and the rest to 0. Each ultrafilter of B thus consists of an atom of B and all the elements above it; hence exactly half the elements of B are in the ultrafilter, and there as many ultrafilters as atoms.

For infinite Boolean algebras the notion of ultrafilter becomes considerably more delicate. The elements greater or equal than an atom always form an ultrafilter but so do many other sets; for example in the Boolean algebra of finite and cofinite sets of integers the cofinite sets form an ultrafilter even though none of them are atoms. Likewise the powerset of the integers has among its ultrafilters the set of all subsets containing a given integer; there are countably many of these "standard" ultrafilters, which may be identified with the integers themselves, but there are uncountably many more "nonstandard" ultrafilters. These form the basis for nonstandard analysis, providing representations for such classically inconsistent objects as infinitesimals and delta functions.

Infinitary extensions

Recall the definition of sup and inf from the section above on the underlying partial order of a Boolean algebra. A complete Boolean algebra is one every subset of which has both a sup and an inf, even the infinite subsets. Gaifman [1964] and Hales [1964] independently showed that infinite free complete Boolean algebras do not exist. This suggests that a logic with set-sized-infinitary operations may have class-many terms—just as a logic with finitary operations may have infinitely many terms.

There is however another approach to introducing infinitary Boolean operations: simply drop "finitary" from the definition of Boolean algebra. A model of the equational theory of the algebra of *all* operations on $\{0,1\}$ of arity up to the cardinality of the model is called a complete atomic Boolean algebra, or *CABA*. (In place of this awkward restriction on arity we could allow any arity, leading to a different awkwardness, that the signature would then be larger than any set, that is, a proper class. One benefit of the latter approach is that it simplifies the definition of homomorphism between CABAs of different cardinality.) Such an algebra can be defined equivalently as a complete Boolean algebra that is **atomic**, meaning that every element is a sup of some set of atoms. Free CABAs exist for all cardinalities of a set V of generators, namely the power set algebra 2^V , this being the obvious generalization of the finite free Boolean algebras. This neatly rescues infinitary Boolean logic from the fate the Gaifman–Hales result seemed to consign it to.

The nonexistence of free complete Boolean algebras can be traced to failure to extend the equations of Boolean logic suitably to all laws that should hold for infinitary conjunction and disjunction, in particular the neglect of distributivity in the definition of complete Boolean algebra. A complete Boolean algebra is called **completely distributive** when arbitrary conjunctions distribute over arbitrary disjunctions and vice versa. A Boolean algebra is a CABA if and only if it is complete and completely distributive, giving a third definition of CABA. A fourth definition is as any Boolean algebra isomorphic to a power set algebra.

A complete homomorphism is one that preserves all supers that exist, not just the finite supers, and likewise for infima. The category **CABA** of all CABAs and their complete homomorphisms is dual to the category of sets and their functions, meaning that it is equivalent to the opposite of that category (the category resulting from reversing all morphisms). Things are not so simple for the category **Bool** of Boolean algebras and their homomorphisms, which Marshall Stone showed in effect (though he lacked both the language and the conceptual framework to make the duality explicit) to be dual to the category of totally disconnected compact Hausdorff spaces, subsequently called Stone spaces.

Another infinitary class intermediate between Boolean algebras and complete Boolean algebras is the notion of a sigma-algebra. This is defined analogously to complete Boolean algebras, but with supers and infima limited to countable arity. That is, a sigma-algebra is a Boolean algebra with all countable supers and infima. Because the supers and infima are of bounded cardinality, unlike the situation with complete Boolean algebras, the Gaifman–Hales result does not apply and free sigma-algebras do exist. Unlike the situation with CABAs however, the free countably generated sigma algebra is not a power set algebra.

Other definitions of Boolean algebra

We have already encountered several definitions of Boolean algebra, as a model of the equational theory of the two-element algebra, as a complemented distributive lattice, as a Boolean ring, and as a product-preserving functor from a certain category (Lawvere). Two more definitions worth mentioning are:

Stone (1936)

A Boolean algebra is the set of all clopen sets of a topological space. It is no limitation to require the space to be a totally disconnected compact Hausdorff space, or Stone space, that is, every Boolean algebra arises in this way, up to isomorphism. Moreover if the two Boolean algebras formed as the clopen sets of two Stone spaces are isomorphic, so are the Stone spaces themselves, which is not the case for arbitrary topological spaces. This is just the reverse direction of the duality mentioned earlier from Boolean algebras to Stone spaces. This definition is fleshed out by the next definition.

Johnstone (1982)

A Boolean algebra is a filtered colimit of finite Boolean algebras.

(The circularity in this definition can be removed by replacing "finite Boolean algebra" by "finite power set" equipped with the Boolean operations standardly interpreted for power sets.)

To put this in perspective, infinite sets arise as filtered colimits of finite sets, infinite CABAs as filtered limits of finite power set algebras, and infinite Stone spaces as filtered limits of finite sets. Thus if one starts with the finite sets and asks how these generalize to infinite objects, there are two ways: "adding" them gives ordinary or inductive sets while "multiplying" them gives Stone spaces or profinite sets. The same choice exists for finite power set algebras as the duals of finite sets: addition yields Boolean algebras as inductive objects while multiplication yields CABAs or power set algebras as profinite objects.

A characteristic distinguishing feature is that the underlying topology of objects so constructed, when defined so as to be Hausdorff, is discrete for inductive objects and compact for profinite objects. The topology of finite Hausdorff spaces is always both discrete and compact, whereas for infinite spaces "discrete" and "compact" are mutually exclusive. Thus when generalizing finite algebras (of any kind, not just Boolean) to infinite ones, "discrete" and "compact" part company, and one must choose which one to retain. The general rule, for both finite and infinite algebras, is that finitary algebras are discrete, whereas their duals are compact and feature infinitary operations. Between these two extremes, there are many intermediate infinite Boolean algebras whose topology is neither discrete nor compact.

References

- Birkhoff, Garrett (1935). "On the structure of abstract algebras". *Proc. Camb. Phil. Soc.* **31**: 433–454. ISSN 0008-1981^[1].
- Boole, George (2003) [1854]. *An Investigation of the Laws of Thought*. Prometheus Books. ISBN 978-1-59102-089-9.
- Dwinger, Philip (1971). *Introduction to Boolean algebras*. Würzburg: Physica Verlag.
- Gaifman, Haim (1964). "Infinite Boolean Polynomials, I". *Fundamenta Mathematicae* **54**: 229–250. ISSN 0016-2736^[2].
- Givant, Steven; Halmos, Paul (2009). *Introduction to Boolean Algebras*. Undergraduate Texts in Mathematics, Springer. ISBN 978-0-387-40293-2.
- Grau, A.A. (1947). "Ternary Boolean algebra". *Bull: Am. Math. Soc.* **33** (6): 567–572. doi:10.1090/S0002-9904-1947-08834-0^[3].
- Hales, Alfred W. (1964). "On the Non-Existence of Free Complete Boolean Algebras". *Fundamenta Mathematicae* **54**: 45–66. ISSN 0016-2736^[2].
- Halmos, Paul (1963). *Lectures on Boolean Algebras*. van Nostrand. ISBN 0-387-90094-2.
- -----, and Givant, Steven (1998) *Logic as Algebra*. Dolciani Mathematical Exposition, No. 21. Mathematical Association of America.
- Johnstone, Peter T. (1982). *Stone Spaces*. Cambridge, UK: Cambridge University Press. ISBN 978-0-521-33779-3.
- Ketonen, Jussi (1978). "The structure of countable Boolean algebras". *Annals of Mathematics* **108** (1): 41–89. doi:10.2307/1970929^[4]. JSTOR 1970929^[5].
- Koppelberg, Sabine (1989) "General Theory of Boolean Algebras" in Monk, J. Donald, and Bonnet, Robert, eds., *Handbook of Boolean Algebras*, Vol. 1. North Holland. ISBN 978-0-444-70261-6.
- Peirce, C. S. (1989) *Writings of Charles S. Peirce: A Chronological Edition: 1879–1884*. Kloesel, C. J. W., ed. Indianapolis: Indiana University Press. ISBN 978-0-253-37204-8.
- Lawvere, F. William (1963). "Functorial semantics of algebraic theories"^[6]. *Proceedings of the National Academy of Sciences* **50** (5): 869–873. doi:10.1073/pnas.50.5.869^[7].
- Schröder, Ernst (1890–1910). *Vorlesungen über die Algebra der Logik (exakte Logik), I–III*. Leipzig: B.G. Teubner.
- Sikorski, Roman (1969). *Boolean Algebras* (3rd. ed.). Berlin: Springer-Verlag. ISBN 978-0-387-04469-9.
- Stone, M. H. (1936). "The Theory of Representation for Boolean Algebras". *Transactions of the American Mathematical Society* **40** (1): 37–111. doi:10.2307/1989664^[8]. ISSN 0002-9947^[9]. JSTOR 1989664^[10].

- Tarski, Alfred (1983). *Logic, Semantics, Metamathematics*, Corcoran, J., ed. Hackett. 1956 1st edition edited and translated by J. H. Woodger, Oxford Uni. Press. Includes English translations of the following two articles:
 - Tarski, Alfred (1929). "Sur les classes closes par rapport à certaines opérations élémentaires". *Fundamenta Mathematicae* **16**: 195–97. ISSN 0016-2736 ^[2].
 - Tarski, Alfred (1935). "Zur Grundlegung der Booleschen Algebra, I". *Fundamenta Mathematicae* **24**: 177–98. ISSN 0016-2736 ^[2].
- Vladimirov, D.A. (1969). *булевы алгебры (Boolean algebras, in Russian, German translation Boolesche Algebren 1974)*. Nauka (German translation Akademie-Verlag).

References

- [1] <http://www.worldcat.org/issn/0008-1981>
- [2] <http://www.worldcat.org/issn/0016-2736>
- [3] <http://dx.doi.org/10.1090%2FS0002-9904-1947-08834-0>
- [4] <http://dx.doi.org/10.2307%2F1970929>
- [5] <http://www.jstor.org/stable/1970929>
- [6] <http://www.tac.mta.ca/tac/reprints/articles/5/tr5abs.html>
- [7] <http://dx.doi.org/10.1073%2Fpnas.50.5.869>
- [8] <http://dx.doi.org/10.2307%2F1989664>
- [9] <http://www.worldcat.org/issn/0002-9947>
- [10] <http://www.jstor.org/stable/1989664>

Boolean function

In mathematics, a **(finitary) Boolean function** (or switching function) is a function of the form $f: \mathbf{B}^k \rightarrow \mathbf{B}$, where $\mathbf{B} = \{0, 1\}$ is a *Boolean domain* and k is a non-negative integer called the arity of the function. In the case where $k = 0$, the "function" is essentially a constant element of \mathbf{B} .

Every k -ary Boolean function can be expressed as a propositional formula in k variables x_1, \dots, x_k , and two propositional formulas are logically equivalent if and only if they express the same Boolean function. There are 2^{2^k} k -ary functions for every k .

Boolean functions in applications

A Boolean function describes how to determine a Boolean value output based on some logical calculation from Boolean inputs. Such functions play a basic role in questions of complexity theory as well as the design of circuits and chips for digital computers. The properties of Boolean functions play a critical role in cryptography, particularly in the design of symmetric key algorithms (see substitution box).

Boolean functions are often represented by sentences in propositional logic, and sometimes as multivariate polynomials over GF(2), but more efficient representations are binary decision diagrams (BDD), negation normal forms, and propositional directed acyclic graphs (PDAG).

In cooperative game theory, monotone Boolean functions are called **simple games** (voting games); this notion is applied to solve problems in social choice theory.

References

- Hazewinkel, Michiel, ed. (2001), "Boolean function" [1], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Digital Design, Mano. M. Morris
- Janković, Dragan; Stanković, Radomir S.; Moraga, Claudio (November 2003). "Arithmetic Expressions Optimisation Using Dual Polarity Property" [2] (PDF). *Serbian journal of electrical engineering* 1 (71 - 80). Retrieved 2010-04-06. More than one of | number= and | issue= specified (help)
- <http://www.answers.com/topic/switching-function?cat=technology>
- http://www-cse.uta.edu/~carroll/cse2341/summer99/powerpoint%20files/chapter_2.ppt

References

- [1] <http://www.encyclopediaofmath.org/index.php?title=p/b016940>
[2] http://www.journal.tfc.kg.ac.rs/Vol_1-1/05-Jankovic.pdf

Boolean-valued function

A **boolean-valued function**, in some usages is a predicate or a proposition, is a function of the type $f : X \rightarrow \mathbf{B}$, where X is an arbitrary set and where \mathbf{B} is a boolean domain.

A **boolean domain \mathbf{B}** is a generic 2-element set, say, $\mathbf{B} = \{0, 1\}$, whose elements are interpreted as logical values, for example, 0 = false and 1 = true.

In the formal sciences, mathematics, mathematical logic, statistics, and their applied disciplines, a boolean-valued function may also be referred to as a characteristic function, indicator function, predicate, or proposition. In all of these uses it is understood that the various terms refer to a mathematical object and not the corresponding semiotic sign or syntactic expression.

In formal semantic theories of truth, a **truth predicate** is a predicate on the sentences of a formal language, interpreted for logic, that formalizes the intuitive concept that is normally expressed by saying that a sentence is true. A truth predicate may have additional domains beyond the formal language domain, if that is what is required to determine a final truth value.

References

- Brown, Frank Markham (2003), *Boolean Reasoning: The Logic of Boolean Equations*, 1st edition, Kluwer Academic Publishers, Norwell, MA. 2nd edition, Dover Publications, Mineola, NY, 2003.
- Kohavi, Zvi (1978), *Switching and Finite Automata Theory*, 1st edition, McGraw-Hill, 1970. 2nd edition, McGraw-Hill, 1978.
- Korfhage, Robert R. (1974), *Discrete Computational Structures*, Academic Press, New York, NY.
- Mathematical Society of Japan, *Encyclopedic Dictionary of Mathematics*, 2nd edition, 2 vols., Kiyosi Itô (ed.), MIT Press, Cambridge, MA, 1993. Cited as EDM.
- Minsky, Marvin L., and Papert, Seymour, A. (1988), *Perceptrons, An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, 1969. Revised, 1972. Expanded edition, 1988.

Conditioned disjunction

In logic, **conditioned disjunction** (sometimes called **conditional disjunction**) is a ternary logical connective introduced by Church. Given operands p , q , and r , which represent truth-valued propositions, the meaning of the conditioned disjunction $[p, q, r]$ is given by:

$$[p, q, r] \leftrightarrow (q \rightarrow p) \wedge (\neg q \rightarrow r)$$

In words, $[p, q, r]$ is equivalent to: "if q then p , else r ", or " p or r , according as q or not q ". This may also be stated as " q implies p and, not q implies r ". So, for any values of p , q , and r , the value of $[p, q, r]$ is the value of p when q is true, and is the value of r otherwise.

The conditioned disjunction is also equivalent to:

$$(q \wedge p) \vee (\neg q \wedge r)$$

and has the same truth table as the "ternary" (?) operator in many programming languages.

In conjunction with truth constants denoting each truth-value, conditioned disjunction is truth-functionally complete for classical logic.^[1] Its truth table is the following:

Conditioned disjunction

p	q	r	[p,q,r]
T	T	T	T
T	T	F	T
T	F	T	T
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	T
F	F	F	F

There are other truth-functionally complete ternary connectives.

References

[1] Wesselkamper, T., "A sole sufficient operator", *Notre Dame Journal of Formal Logic*, Vol. XVI, No. 1 (1975), pp. 86-88.

Converse implication

Converse implication is the converse of implication. That is to say; that for any two propositions P and Q, if Q implies P, then P is the converse implication of Q.

It may take the following forms:

$p \sqsupseteq q$, Bpq , or $p \leftarrow q$

Definition

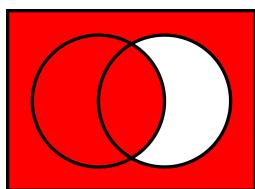
Truth table

The truth table of ACB

a	b	\sqsupseteq
T	T	T
T	F	T
F	T	F
F	F	T

Venn diagram

The Venn diagram of "If B then A" (the white area shows where the statement is false)



Properties

truth-preserving: The interpretation under which all variables are assigned a truth value of 'true' produces a truth value of 'true' as a result of converse implication.

Natural language

"Not q without p."

"p if q."

Boolean Algebra

$(A + B')$

Converse nonimplication

In logic, **converse nonimplication**^[1] is a logical connective which is the negation of the converse of implication.

Definition

$p \not\subset q$ which is the same as $\sim(p \subset q)$

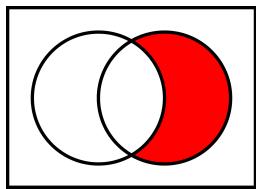
Truth table

The truth table of $p \not\subset q$.

p	q	$\not\subset$
T	T	F
T	F	F
F	T	T
F	F	F

Venn diagram

The Venn Diagram of "It is not the case that B implies A" (the red area is true)



Properties

falsehood-preserving: The interpretation under which all variables are assigned a truth value of 'false' produces a truth value of 'false' as a result of converse nonimplication

Symbol

Alternatives for $p \not\subset q$ are

- $\tilde{p} \leftarrow q$: \leftarrow combines Converse implication's left arrow(\leftarrow) with Negation's tilde(\sim).
- Mpq : uses prefixed capital letter.
- $p \not\leftarrow q$: $\not\leftarrow$ combines *Converse implication's* left arrow(\leftarrow) denied by means of a stroke(/).

Natural language

Rhetorical

"not A but B"

Boolean algebra

Converse Nonimplication in a general Boolean algebra is defined as $q \leftrightarrow p = q'p$.

Example of a 2-element Boolean algebra: the 2 elements $\{0,1\}$ with 0 as zero and 1 as unity element, operators \sim as complement operator, \vee as join operator and \wedge as meet operator, build the Boolean algebra of propositional logic.

$\begin{array}{ c c c } \hline \sim x & 1 & 0 \\ \hline x & 0 & 1 \\ \hline \end{array}$	and	$\begin{array}{ c c c c } \hline y & & & \\ \hline 1 & 1 & 1 & \\ \hline 0 & 0 & 1 & \\ \hline y \vee x & 0 & 1 & x \\ \hline \end{array}$	and	$\begin{array}{ c c c c } \hline y & & & \\ \hline 1 & 0 & 1 & \\ \hline 0 & 0 & 0 & \\ \hline y \wedge x & 0 & 1 & x \\ \hline \end{array}$	then $y \leftrightarrow x$ means	$\begin{array}{ c c c c } \hline y & & & \\ \hline 1 & 0 & 0 & \\ \hline 0 & 0 & 1 & \\ \hline y \leftrightarrow x & 0 & 1 & x \\ \hline \end{array}$
(Negation)		(Inclusive Or)		(And)		(Converse Nonimplication)

[4] Example of a 4-element Boolean algebra: the 4 divisors $\{1,2,3,6\}$ of 6 with 1 as zero and 6 as unity element, operators c (codivisor of 6) as complement operator, \vee (least common multiple) as join operator and \wedge (greatest common divisor) as meet operator, build a Boolean algebra.

$\begin{array}{ c c c c c } \hline x^c & 6 & 3 & 2 & 1 \\ \hline x & 1 & 2 & 3 & 6 \\ \hline \end{array}$	and	$\begin{array}{ c c c c c c } \hline y & & & & & \\ \hline 6 & 6 & 6 & 6 & 6 & \\ \hline 3 & 3 & 6 & 3 & 6 & \\ \hline 2 & 2 & 2 & 6 & 6 & \\ \hline 1 & 1 & 2 & 3 & 6 & \\ \hline y \vee x & 1 & 2 & 3 & 6 & x \\ \hline \end{array}$	and	$\begin{array}{ c c c c c c } \hline y & & & & & \\ \hline 6 & 1 & 2 & 3 & 6 & \\ \hline 3 & 1 & 1 & 3 & 3 & \\ \hline 2 & 1 & 2 & 1 & 2 & \\ \hline 1 & 1 & 1 & 1 & 1 & \\ \hline y \wedge x & 1 & 2 & 3 & 6 & x \\ \hline \end{array}$	then $y \leftrightarrow x$ means	$\begin{array}{ c c c c c c } \hline y & & & & & \\ \hline 6 & 1 & 1 & 1 & 1 & \\ \hline 3 & 1 & 2 & 1 & 2 & \\ \hline 2 & 1 & 1 & 3 & 3 & \\ \hline 1 & 1 & 2 & 3 & 6 & \\ \hline y \leftrightarrow x & 1 & 2 & 3 & 6 & x \\ \hline \end{array}$
(Codivisor 6)		(Least Common Multiple)		(Greatest Common Divisor)		(x's greatest Divisor coprime with y)

Properties

Non-associative

$r \leftrightarrow (q \leftrightarrow p) = (r \leftrightarrow q) \leftrightarrow p$ iff $rp=0$ [5] (In a two-element Boolean algebra the latter condition is reduced to $r=0$ or $p=0$). Hence in a nontrivial Boolean algebra Converse Nonimplication is **nonassociative**.

Clearly, it is associative iff $rp=0$.

Non-commutative

- $q \leftrightarrow p = p \leftrightarrow q$ iff $q=p$ [6]. Hence Converse Nonimplication is **noncommutative**.

Neutral and absorbing elements

- 0 is a left neutral element ($0 \leftrightarrow p = p$) and a right absorbing element ($p \leftrightarrow 0 = 0$).
- $1 \leftrightarrow p = 0$, $p \leftrightarrow 1 = p'$, and $p \leftrightarrow p = 0$.
- Implication $q \rightarrow p$ is the dual of Converse Nonimplication $q \leftrightarrow p$ [7].

[6]

Converse Nonimplication is noncommutative				
Step	Make use of	Resulting in		
s.1	Definition	$q \tilde{\rightarrow} p = q' p$		
s.2	Definition	$p \tilde{\rightarrow} q = p' q$		
s.3	s.1 s.2	$q \tilde{\rightarrow} p = p \tilde{\rightarrow} q \Leftrightarrow q' p = q p'$		
s.4		q	=	$q \cdot 1$
s.5	s.4.right - expand Unit element		=	$q \cdot (p + p')$
s.6	s.5.right - evaluate expression		=	$qp + qp'$
s.7	s.4.left=s.6.right	$q = qp + qp'$		
s.8		$q' p = qp'$	\Rightarrow	$qp + qp' = qp + q' p$
s.9	s.8-regroup common factors		\Rightarrow	$q \cdot (p + p') = (q + q') \cdot p$
s.10	s.9- join of complements equals unity		\Rightarrow	$q \cdot 1 = 1 \cdot p$
s.11	s.10.right - evaluate expression		\Rightarrow	$q = p$
s.12	s.8 s.11	$q' p = qp' \Rightarrow q = p$		
s.13		$q = p \Rightarrow q' p = qp'$		
s.14	s.12 s.13	$q = p \Leftrightarrow q' p = qp'$		
s.15	s.3 s.14	$q \tilde{\rightarrow} p = p \tilde{\rightarrow} q \Leftrightarrow q = p$		

[7]

Implication is the dual of Converse Nonimplication				
Step	Make use of	Resulting in		
s.1	Definition	$dual(q \tilde{\rightarrow} p)$	=	$dual(q' p)$
s.2	s.1.right - .'s dual is +		=	$q' + p$
s.3	s.2.right - Involution complement		=	$(q' + p)''$
s.4	s.3.right - De Morgan's laws applied once		=	$(qp')'$
s.5	s.4.right - Commutative law		=	$(p' q)'$
s.6	s.5.right		=	$(p \tilde{\rightarrow} q)'$
s.7	s.6.right		=	$p \tilde{\rightarrow} q$
s.8	s.7.right		=	$q \rightarrow p$
s.9	s.1.left=s.8.right	$dual(q \tilde{\rightarrow} p) = q \rightarrow p$		

Computer science

An example for converse nonimplication in computer science can be found when performing a right outer join on a set of tables from a database, if records not matching the join-condition from the "left" table are being excluded.^[2]

Notes

[1] Lehtonen, Eero, and Poikonen, J.H.

[2] <http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html>

References

- Knuth, Donald E. (2011). *The Art of Computer Programming, Volume 4A: Combinatorial Algorithms, Part 1* (1st ed.). Addison-Wesley Professional. ISBN 0-201-03804-8.

Evasive Boolean function

In mathematics, an **evasive Boolean function** f (of n variables) is a Boolean function for which every decision tree algorithm has running time of exactly n . Consequently every decision tree algorithm that represents the function has, at worst case, a running time of n .

Examples

An example for a non-evasive boolean function

The following is a Boolean function on the three variables x, y, z :

$f(x, y, z)$	=	$(x \wedge y)$	\vee	$(\neg x \wedge z)$
	=		\vee	

(where \wedge is the bitwise "and", \vee is the bitwise "or", and \neg is the bitwise "not").

This function is not evasive, because there is a decision tree that solves it by checking exactly two variables: The algorithm first checks the value of x . If x is true, the algorithm checks the value of y and returns it.

$$(\neg x = \text{false}) \Rightarrow ((\neg x \wedge z) = \text{false})$$

If x is false, the algorithm checks the value of z and returns it.

A simple example for an evasive boolean function

Consider this simple "and" function on three variables:

$f(x, y, z)$	=	$(x \wedge y \wedge z)$
		

A worst-case input (for every algorithm) is 1, 1, 1. In every order we choose to check the variables, we have to check all of them. (Note that in general there could be a different worst-case input for every decision tree algorithm.) Hence the functions: "and", "or" (on n variables) are evasive.

Binary zero-sum games

For the case of binary zero-sum games, every evaluation function is evasive.

In every zero-sum game, the value of the game is achieved by the minimax algorithm (player 1 tries to maximize the profit, and player 2 tries to minimize the cost).

In the binary case, the max function equals the bitwise "or", and the min function equals the bitwise "and".

A decision tree for this game will be of this form:

- every leaf will have value in {0, 1}.
- every node is connected to one of {"and", "or"}

For every such tree with n leaves, the running time in the worst case is n (meaning that the algorithm must check all the leaves):

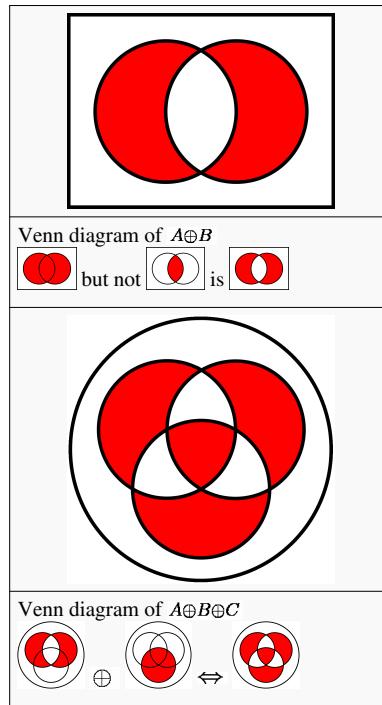
We will exhibit an adversary that produces a worst-case input – for every leaf that the algorithm checks, the adversary will answer 0 if the leaf's parent is an Or node, and 1 if the parent is an And node.

This input (0 for all Or nodes' children, and 1 for all And nodes' children) forces the algorithm to check all nodes:

As in the second example

- in order to calculate the Or result, if all children are 0 we must check them all.
- In order to calculate the and result, if all children are 1 we must check them all.

Exclusive or



Exclusive disjunction or **exclusive or** (/ɛksHelp:IPA for English#Key'ɔr/) is a logical operation that outputs true whenever both inputs differ (one is true, the other is false). It is symbolized by the prefix operator **J** and by the infix operators **XOR**, **EOR**, **EXOR**, \oplus , $\overline{\oplus}$, and $\overline{\overline{\oplus}}$. In mathematical logic, there is no symbol for xor. The opposite of XOR is logical biconditional, which outputs true whenever both inputs are the same.

It gains the name "exclusive or" because the meaning of "or" is ambiguous when both operands are true; exclusive or *excludes* that case. This is sometimes thought of as "one or the other but not both".

More generally, XOR is true whenever an odd number of inputs is true. A chain of XORs— a XOR b XOR c XOR d (and so on)—is true whenever an odd number of the inputs are true and is false whenever an even number of inputs are true.

Truth table

The truth table of A XOR B shows that it outputs true whenever the inputs differ:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
()	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(A)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
(B)	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
(AB)	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
(C)	0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
(AC)	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(BC)	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(ABC)	0	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
(D)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(AD)	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(BD)	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(ABD)	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(CD)	0	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
(ACD)	0	1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(BCD)	0	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
(ABCD)	0	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0

Arguments on the left combined by XOR
This is a binary Walsh matrix
(compare: Hadamard code)

XOR Truth Table

Input	Output	
	A	B
0 0	0	
0 1		1
1 0		1
1 1	0	

Equivalencies, elimination, and introduction

Exclusive disjunction essentially means 'either one, but not both'. In other words, if and only if one is true, the other cannot be true. For example, one of the two horses will win the race, but not both of them. The exclusive disjunction : $p \oplus q$, or Jpq , can be expressed in terms of the logical conjunction (\wedge), the disjunction (\vee), and the negation (\neg) as follows:

$$p \oplus q = (p \vee q) \wedge \neg(p \wedge q)$$

The exclusive disjunction $p \oplus q$ can also be expressed in the following way:

$$p \oplus q = (p \wedge \neg q) \vee (\neg p \wedge q)$$

This representation of XOR may be found useful when constructing a circuit or network, because it has only one \neg operation and small number of \wedge and \vee operations. The proof of this identity is given below:

$$\begin{aligned}
 p \oplus q &= (p \wedge \neg q) \vee (\neg p \wedge q) \\
 &= ((p \wedge \neg q) \vee \neg p) \wedge ((p \wedge \neg q) \vee q) \\
 &= ((p \vee \neg p) \wedge (\neg q \vee \neg p)) \wedge ((p \vee q) \wedge (\neg q \vee q)) \\
 &= (\neg p \vee \neg q) \wedge (p \vee q) \\
 &= \neg(p \wedge q) \wedge (p \vee q)
 \end{aligned}$$

It is sometimes useful to write $p \oplus q$ in the following way:

$$p \oplus q = \neg((p \wedge q) \vee (\neg p \wedge \neg q))$$

This equivalence can be established by applying De Morgan's laws twice to the fourth line of the above proof.

The exclusive or is also equivalent to the negation of a logical biconditional, by the rules of material implication (a material conditional is equivalent to the disjunction of the negation of its antecedent and its consequence) and material equivalence.

In summary, we have, in mathematical and in engineering notation:

$$\begin{aligned}
 p \oplus q &= (p \wedge \neg q) \vee (\neg p \wedge q) = p\bar{q} + \bar{p}q \\
 &= (p \vee q) \wedge (\neg p \vee \neg q) = (p + q)(\bar{p} + \bar{q}) \\
 &= (p \vee q) \wedge \neg(p \wedge q) = (p + q)(\bar{p}\bar{q})
 \end{aligned}$$

Relation to modern algebra

Although the operators \wedge (conjunction) and \vee (disjunction) are very useful in logic systems, they fail a more generalizable structure in the following way:

The systems $(\{T, F\}, \wedge)$ and $(\{T, F\}, \vee)$ are monoids. This unfortunately prevents the combination of these two systems into larger structures, such as a mathematical ring.

However, the system using exclusive or $(\{T, F\}, \oplus)$ is an abelian group. The combination of operators \wedge and \oplus over elements $\{T, F\}$ produce the well-known field F_2 . This field can represent any logic obtainable with the system (\wedge, \vee) and has the added benefit of the arsenal of algebraic analysis tools for fields.

More specifically, if one associates F with 0 and T with 1, one can interpret the logical "AND" operation as multiplication on F_2 and the "XOR" operation as addition on F_2 :

$$r = p \wedge q \Leftrightarrow r = p \cdot q \pmod{2}$$

$$r = p \oplus q \Leftrightarrow r = p + q \pmod{2}$$

Using this basis to describe a boolean system is referred to as algebraic normal form

Exclusive "or" in English

The Oxford English Dictionary explains "either ... or" as follows:

"The primary function of *either*, etc., is to emphasize the perfect indifference of the two (or more) things or courses ... ; but a secondary function is to emphasize the mutual exclusiveness, = either of the two, but not both."^[1]

The exclusive-or explicitly states "one or the other, but not neither nor both." However, the mapping correspondence between formal Boolean operators and natural language conjunctions is far from simple or one-to-one, and has been studied for decades in linguistics and analytic philosophy^[citation needed].

Following this kind of common-sense intuition about "or", it is sometimes argued that in many natural languages, English included, the word "or" has an "exclusive" sense.^[citation needed] The **exclusive disjunction** of a pair of propositions, (p, q) , is supposed to mean that p is true or q is true, but not both. For example, it might be argued that

the normal intention of a statement like "You may have coffee, or you may have tea" is to stipulate that exactly one of the conditions can be true. Certainly under some circumstances a sentence like this example should be taken as forbidding the possibility of one's accepting both options. Even so, there is good reason to suppose that this sort of sentence is not disjunctive at all. If all we know about some disjunction is that it is true overall, we cannot be sure that either of its disjuncts is true.^[citation needed] Wikipedia:Disputed statement For example, if a woman has been told that her friend is either at the snack bar or on the tennis court, she cannot validly infer that he is on the tennis court. But if her waiter tells her that she may have coffee or she may have tea, she can validly infer that she may have tea. Nothing classically thought of as a disjunction has this property. This is so even given that she might reasonably take her waiter as having denied her the possibility of having both coffee and tea.

(Note: If the waiter intends that choosing neither tea nor coffee is an option i.e. ordering nothing, the appropriate operator is NAND: p NAND q.)^[citation needed] Wikipedia:Disputed statement

In English, the construct "either ... or" is usually used to indicate exclusive or and "or" generally used for inclusive.^[citation needed] Wikipedia:Disputed statement But in Spanish, the word "o" (or) can be used in the form $p \text{ o } q$ (exclusive) or the form $\text{o } p \text{ o } q$ (inclusive). Some may contend that any binary or other n-ary exclusive "or" is true if and only if it has an odd number of true inputs (this is not, however, the only reasonable definition; for example, digital xor gates with multiple inputs typically do not use that definition), and that there is no conjunction in English that has this general property. For example, Barrett and Stenner contend in the 1971 article "The Myth of the Exclusive 'Or'" (Mind, 80 (317), 116–121) that no author has produced an example of an English or-sentence that appears to be false because both of its inputs are true, and brush off or-sentences such as "The light bulb is either on or off" as reflecting particular facts about the world rather than the nature of the word "or". However, the "barber paradox"—Everybody in town shaves himself or is shaved by the barber, who shaves the barber? -- would not be paradoxical if "or" could not be exclusive (although a purist could say that "either" is required in the statement of the paradox).

Whether these examples can be considered "natural language" is another question.^[citation needed] Wikipedia:Disputed statement. Certainly when one sees a menu stating "Lunch special: sandwich and soup or salad" (parsed as "sandwich and (soup or salad)" according to common usage in the restaurant trade), one would not expect to be permitted to order both soup and salad. Nor would one expect to order neither soup nor salad, because that belies the nature of the "special", that ordering the two items together is cheaper than ordering them a la carte. Similarly, a lunch special consisting of one meat, French fries or mashed potatoes and vegetable would consist of three items, only one of which would be a form of potato. If one wanted to have meat and both kinds of potatoes, one would ask if it were possible to substitute a second order of potatoes for the vegetable. And, one would not expect to be permitted to have both types of potato and vegetable, because the result would be a vegetable plate rather than a meat plate.

Alternative symbols

The symbol used for exclusive disjunction varies from one field of application to the next, and even depends on the properties being emphasized in a given context of discussion. In addition to the abbreviation "XOR", any of the following symbols may also be seen:

- A plus sign (+). This makes sense mathematically because exclusive disjunction corresponds to addition modulo 2, which has the following addition table, clearly isomorphic to the one above:

Addition Modulo 2

p	q	$p + q$
0	0	0
0	1	1
1	0	1
1	1	0

- The use of the plus sign has the added advantage that all of the ordinary algebraic properties of mathematical rings and fields can be used without further ado. However, the plus sign is also used for Inclusive disjunction in some notation systems.
- A plus sign that is modified in some way, such as being encircled (\oplus). This usage faces the objection that this same symbol is already used in mathematics for the *direct sum* of algebraic structures.
- A prefixed J, as in Jpq .
- An inclusive disjunction symbol (\vee) that is modified in some way, such as being underlined ($\underline{\vee}$) or with dot above ($\dot{\vee}$).
- In several programming languages, such as C, C++, C#, Java, Perl, MATLAB, and Python, a caret (^) is used to denote the bitwise XOR operator. This is not used outside of programming contexts because it is too easily confused with other uses of the caret.
- The symbol \bowtie , sometimes written as $><$ or as $>-<$.
- In IEC symbology, an exclusive or is marked " $=1$ ".

Properties

Commutativity: yes

$A \oplus B$	\Leftrightarrow	$B \oplus A$
	\Leftrightarrow	

Associativity: yes

A	\oplus	$(B \oplus C)$	\Leftrightarrow			$(A \oplus B)$	\oplus	C
	\oplus		\Leftrightarrow		\Leftrightarrow		\oplus	

Distributivity: The exclusive or doesn't distribute over any binary function (not even itself), but logical conjunction (see there) distributes over exclusive or.

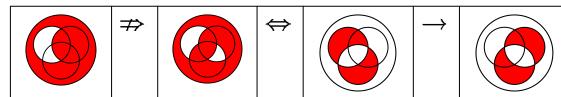
(Conjunction and exclusive or form the multiplication and addition operations of a field GF(2), and as in any field they obey the distributive law.)

Idempotency: no

A	\oplus	A	\Leftrightarrow	0	\Rightarrow	A
	\oplus		\Leftrightarrow		\Rightarrow	

Monotonicity: no

$A \rightarrow B$	\Rightarrow			$(A \oplus C)$	\rightarrow	$(B \oplus C)$
-------------------	---------------	--	--	----------------	---------------	----------------



Truth-preserving: no

When all inputs are true, the output is not true.

$A \wedge B$	\Rightarrow	$A \oplus B$
	\Rightarrow	

Falsehood-preserving: yes

When all inputs are false, the output is false.

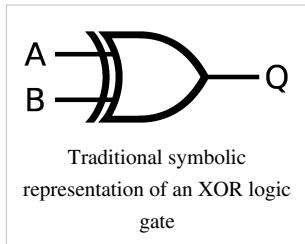
$A \oplus B$	\Rightarrow	$A \vee B$
	\Rightarrow	

Walsh spectrum: (2,0,0,-2)

Non-linearity: 0 (the function is linear)

If using binary values for true (1) and false (0), then *exclusive or* works exactly like addition modulo 2.

Computer science



Bitwise operation

Exclusive disjunction is often used for bitwise operations. Examples:

- $1 \text{ xor } 1 = 0$
- $1 \text{ xor } 0 = 1$
- $0 \text{ xor } 1 = 1$
- $0 \text{ xor } 0 = 0$
- $1110 \text{ xor } 1001 = 0111$ (this is equivalent to addition without carry)

As noted above, since exclusive disjunction is identical to addition modulo 2, the bitwise exclusive disjunction of two n -bit strings is identical to the standard vector of addition in the vector space $(\mathbb{Z}/2\mathbb{Z})^n$.

In computer science, exclusive disjunction has several uses:

- It tells whether two bits are unequal.
- It is an optional bit-flipper (the deciding input chooses whether to invert the data input).
- It tells whether there is an odd number of 1 bits ($A \oplus B \oplus C \oplus D \oplus E$ is true iff an odd number of the variables are true).

In logical circuits, a simple adder can be made with an XOR gate to add the numbers, and a series of AND, OR and NOT gates to create the carry output.

On some computer architectures, it is more efficient to store a zero in a register by xor-ing the register with itself (bits xor-ed with themselves are always zero) instead of loading and storing the value zero.

In simple threshold activated neural networks, modeling the 'xor' function requires a second layer because 'xor' is not a linearly separable function.

Exclusive-or is sometimes used as a simple mixing function in cryptography, for example, with one-time pad or Feistel network systems.

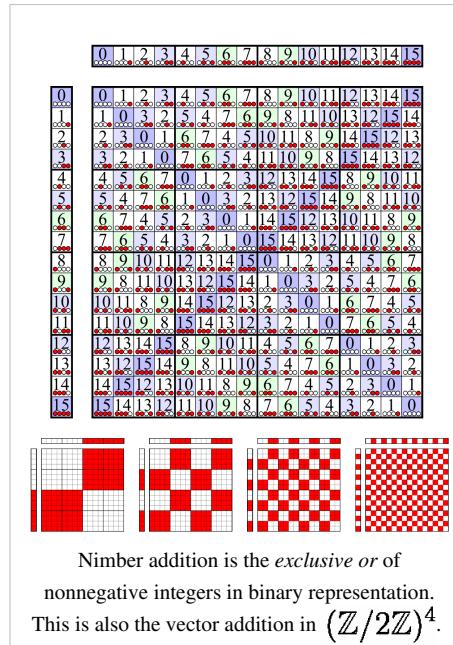
Similarly, XOR can be used in generating entropy pools for hardware random number generators. The XOR operation preserves randomness, meaning that a random bit XORed with a non-random bit will result in a random bit. Multiple sources of potentially random data can be combined using XOR, and the unpredictability of the output is guaranteed to be at least as good as the best individual source.^[2]

XOR is used in RAID 3–6 for creating parity information. For example, RAID can "back up" bytes 10011100 and 01101100 from two (or more) hard drives by XORing the just mentioned bytes, resulting in (11110000) and writing it to another drive. Under this method, if any one of the three hard drives are lost, the lost byte can be re-created by XORing bytes from the remaining drives. For instance, if the drive containing 01101100 is lost, 10011100 and 11110000 can be XORed to recover the lost byte.

XOR is also used to detect an overflow in the result of a signed binary arithmetic operation. If the leftmost retained bit of the result is not the same as the infinite number of digits to the left, then that means overflow occurred. XORing those two bits will give a "1" if there is an overflow.

XOR can be used to swap two numeric variables in computers, using the XOR swap algorithm; however this is regarded as more of a curiosity and not encouraged in practice.

In computer graphics, XOR-based drawing methods are often used to manage such items as bounding boxes and cursors on systems without alpha channels or overlay planes.



Encodings

Apart from the obvious ASCII codes, the operator is encoded at U+22BB \boxplus xor (HTML: ⊻) and U+2295 \oplus circled plus (HTML: ⊕ ⊕), both in block Mathematical Operators.

Notes

[1] or, conj.2 (adv.3) 2a *Oxford English Dictionary*, second edition (1989). OED Online.

[2] <http://www.digipedia.org/usenet/thread/11834/2075/>

External links

- An example of XOR being used in cryptography (<http://www.codeplex.com/rexor>)

False (logic)

In logic, **false** is a truth value or a nullary logical connective. In a truth-functional system of propositional logic it is one of two postulated truth values, along with its negation, truth.^[1] Usual notations of the false are 0 (especially in Boolean logic and computer science) and the up tack symbol \perp .^[2]

Another approach is used for several formal theories (for example, intuitionistic propositional calculus) where the false is a propositional constant (i.e. a nullary connective) \perp , the truth value of this constant being always false in the sense above.^{[3][4][5]}

In classical logic and Boolean logic

Boolean logic defines the false in both senses mentioned above: "0" is a propositional constant, whose value by definition is 0. In a classical propositional calculus, depending on the chosen set of fundamental connectives, the false may or may not have a dedicated symbol. Such formulas as $p \wedge \neg p$ and $\neg(p \rightarrow p)$ may be used instead.

In both systems the negation of the truth gives false. The negation of false is equivalent to the truth not only in classical logic and Boolean logic, but also in most other logical systems, as explained below.

False, negation and contradiction

In most logical systems, negation, material conditional and false are related as:

$$\neg p \Leftrightarrow (p \rightarrow \perp)$$

This is the definition of negation in some systems,^[6] such as intuitionistic logic, and can be proven in propositional calculi where negation is a fundamental connective. Because $p \rightarrow p$ is usually a theorem or axiom, a consequence is that the negation of false ($\neg \perp$) is true.

The contradiction is a statement which entails the false, i.e. $\varphi \vdash \perp$. Using the equivalence above, the fact that φ is a contradiction may be derived, for example, from $\vdash \neg\varphi$. Contradiction and the false are sometimes not distinguished, especially due to Latin term *falsum* denoting both. Contradiction means a statement is proven to be false, but the false itself is a proposition which is defined to be opposite to the truth.

Logical systems may or may not contain the principle of explosion (in Latin, *ex falso quodlibet*), $\perp \vdash \varphi$.

Consistency

A formal theory using " \perp " connective is defined to be consistent if and only if the false is not its theorem. In the absence of propositional constants, some substitutes such as mentioned above may be used instead to define consistency.

References

- [1] Jennifer Fisher, *On the Philosophy of Logic*, Thomson Wadsworth, 2007, ISBN 0-495-00888-5, p. 17. (http://books.google.com/books?id=k8L_YW-IEEQC&pg=PT27)
- [2] Willard Van Orman Quine, *Methods of Logic*, 4th ed, Harvard University Press, 1982, ISBN 0-674-57176-2, p. 34. (<http://books.google.com/books?id=liHivlUYWcUC&pg=PA34>)
- [3] George Edward Hughes and D.E. Londey, *The Elements of Formal Logic*, Methuen, 1965, p. 151. (<http://books.google.com/books?id=JbwOAAAAQAAJ&pg=PA151>)
- [4] Leon Horsten and Richard Pettigrew, *Continuum Companion to Philosophical Logic*, Continuum International Publishing Group, 2011, ISBN 1-4411-5423-X, p. 199. (http://books.google.com/books?id=w_abLTXIFkcC&pg=PA199)
- [5] Graham Priest, *An Introduction to Non-Classical Logic: From If to Is*, 2nd ed, Cambridge University Press, 2008, ISBN 0-521-85433-4, p. 105. (<http://books.google.com/books?id=rMXVbmAw3YwC&pg=PA105>)
- [6] Dov M. Gabbay and Franz Guenther (eds), *Handbook of Philosophical Logic, Volume 6*, 2nd ed, Springer, 2002, ISBN 1-4020-0583-0, p. 12. (<http://books.google.com.au/books?id=JyewdfGhNAsC&pg=PA12>)

Functional completeness

In logic, a **functionally complete** set of logical connectives or Boolean operators is one which can be used to express all possible truth tables by combining members of the set into a Boolean expression.^{[1][2]} A well-known complete set of connectives is { AND, NOT }, consisting of binary conjunction and negation. The singleton sets { NAND } and { NOR } are also functionally complete.

In a context of propositional logic, functionally complete sets of connectives are also called **(expressively) adequate**.^[3]

From the point of view of digital electronics, functional completeness means that every possible logic gate can be realized as a network of gates of the types prescribed by the set. In particular, all logic gates can be assembled from either only binary NAND gates, or only binary NOR gates.

Formal definition

Given the Boolean domain $\mathbf{B} = \{0,1\}$, a set F of Boolean functions $f_i: \mathbf{B}^n \rightarrow \mathbf{B}$ is **functionally complete** if the clone on \mathbf{B} generated by the basic functions f_i contains all functions $f: \mathbf{B}^n \rightarrow \mathbf{B}$, for all *strictly positive* integers $n \geq 1$. In other words, the set is functionally complete if every Boolean function that takes at least one variable can be expressed in terms of the functions f_i . Since every Boolean function of at least one variable can be expressed in terms of binary Boolean functions, F is functionally complete if and only if every binary Boolean function can be expressed in terms of the functions in F .

A more natural condition would be that the clone generated by F consist of all functions $f: \mathbf{B}^n \rightarrow \mathbf{B}$, for all integers $n \geq 0$. However, the examples given above are not functionally complete in this stronger sense because it is not possible to write a nullary function, i.e. a constant expression, in terms of F if F itself does not contain at least one nullary function. With this stronger definition, the smallest functionally complete sets would have 2 elements.

Another natural condition would be that the clone generated by F together with the two nullary constant functions be functionally complete or, equivalently, functionally complete in the strong sense of the previous paragraph. The example of the Boolean function given by $S(x, y, z) = z$ if $x = y$ and $S(x, y, z) = x$ otherwise shows that this condition is strictly weaker than functional completeness.

Informal definition

Modern texts on logic typically take as primitive some subset of the connectives: conjunction (\wedge), or Kpq ; disjunction (\vee), or Apq ; negation (\neg), Np ; or material conditional (\rightarrow), or Cpq ; and possibly the biconditional (\leftrightarrow), or Epq . These connectives are functionally complete. However, they do not form a minimal functionally complete set, as the conditional and biconditional may be defined as:

$$A \rightarrow B := \neg A \vee B$$

$$A \leftrightarrow B := (A \rightarrow B) \wedge (B \rightarrow A).$$

So $\{\neg, \wedge, \vee\}$ is also functionally complete. But then, \vee can be defined as

$$A \vee B := \neg(\neg A \wedge \neg B).$$

\wedge can also be defined in terms of \vee in a similar manner.

It is also the case that \vee can be defined in terms of \rightarrow as follows:

$$A \vee B := (A \rightarrow B) \rightarrow B.$$

No further simplifications are possible. Hence \neg and one of $\{\wedge, \vee, \rightarrow\}$ are each minimal functionally complete subsets of $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

Characterization of functional completeness

Emil Post proved that a set of logical connectives is functionally complete if and only if it is not a subset of any of the following sets of connectives:

- The monotonic connectives; changing the truth value of any connected variables from **F** to **T** without changing any from **T** to **F** never makes these connectives change their return value from **T** to **F**, e.g. $\vee, \wedge, \top, \perp$.
- The affine connectives, such that each connected variable either always or never affects the truth value these connectives return, e.g. $\neg, \top, \perp, \leftrightarrow, \not\leftrightarrow$.
- The **self-dual** connectives, which are equal to their own de Morgan dual; if the truth values of all variables are reversed, so is the truth value these connectives return, e.g. $\neg, MAJ(p,q,r)$.
- The **truth-preserving** connectives; they return the truth value **T** under any interpretation which assigns **T** to all variables, e.g. $\vee, \wedge, \top, \rightarrow, \leftrightarrow$.
- The **falsity-preserving** connectives; they return the truth value **F** under any interpretation which assigns **F** to all variables, e.g. $\vee, \wedge, \perp, \not\rightarrow, \not\leftrightarrow$.

In fact, Post gave a complete description of the lattice of all clones (sets of operations closed under composition and containing all projections) on the two-element set $\{\mathbf{T}, \mathbf{F}\}$, nowadays called Post's lattice, which implies the above result as a simple corollary: the five mentioned sets of connectives are exactly the maximal clones.

Minimal functionally complete operator sets

When a single logical connective or Boolean operator is functionally complete by itself, it is called a **Sheffer function**^[4] or sometimes a **sole sufficient operator**. There are no unary operators with this property, and the only binary Sheffer functions — NAND and NOR are dual. These were discovered but not published by Charles Sanders Peirce around 1880, and rediscovered independently and published by Henry M. Sheffer in 1913. In digital electronics terminology, the binary NAND gate and the binary NOR gate are the only binary universal logic gates.

The following are the minimal functionally complete sets of logical connectives with arity ≤ 2 :^[5]

One element

$$\{\text{NAND}\}, \{\text{NOR}\}.$$

Two elements

$\{\vee, \neg\}, \{\wedge, \neg\}, \{\rightarrow, \neg\}, \{\leftarrow, \neg\}, \{\rightarrow, \perp\}, \{\leftarrow, \perp\}, \{\rightarrow, \not\leftrightarrow\}, \{\leftarrow, \not\leftrightarrow\}, \{\rightarrow, \not\rightarrow\}, \{\leftarrow, \not\rightarrow\}, \{\leftarrow, \not\leftrightarrow\}, \{\not\rightarrow, \neg\}, \{\not\leftrightarrow, \neg\}, \{\not\rightarrow, \top\}, \{\not\leftrightarrow, \top\}, \{\not\rightarrow, \leftrightarrow\}, \{\not\leftrightarrow, \leftrightarrow\}$.

Three elements

$\{\vee, \leftrightarrow, \perp\}, \{\vee, \leftrightarrow, \not\leftrightarrow\}, \{\vee, \not\leftrightarrow, \top\}, \{\wedge, \leftrightarrow, \perp\}, \{\wedge, \leftrightarrow, \not\leftrightarrow\}, \{\wedge, \not\leftrightarrow, \top\}$.

There are no minimal functionally complete sets of more than three at most binary logical connectives. Constant unary or binary connectives and binary connectives that depend only on one of the arguments have been suppressed to keep the list readable. E.g. the set consisting of binary \vee and the binary connective given by negation of the first argument (ignoring the second) is another minimal functionally complete set.

Examples

- Examples of using the NAND completeness. As illustrated by,^[6]
 - $\neg A = A \text{ NAND } A$
 - $A \wedge B = \neg(A \text{ NAND } B) = (A \text{ NAND } B) \text{ NAND } (A \text{ NAND } B)$
 - $A \vee B = (A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B)$
- Examples of using the NOR completeness. As illustrated by,^[7]
 - $\neg A = A \text{ NOR } A$
 - $A \wedge B = (A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$
 - $A \vee B = (A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B)$

Note that, an electronic circuit or a software function is optimized by the reuse, that reduce the number of gates. For instance, the " $A \wedge B$ " operation, when expressed by NAND gates, is implemented with the reuse of " $A \text{ NAND } B$ ",

$$X = (A \text{ NAND } B); A \wedge B = X \text{ NAND } X$$

In other domains

Apart from logical connectives (Boolean operators), functional completeness can be introduced in other domains. For example, a set of reversible gates is called functionally complete, if it can express every reversible operator.

The 3-input Fredkin gate is functionally complete reversible gate by itself – a sole sufficient operator. There are many other three-input universal logic gates, such as the Toffoli gate.

Set theory

There is an isomorphism between the Algebra of sets and the Boolean algebra, that is, they have the same structure. Then, if we map boolean operators into set operators, the "translated" above text are valid also for sets: there are many "minimal complete set of set-theory operators" that can generate any other set relations. The more popular "Minimal complete operator sets" are $\{\neg, \cap\}$ and $\{\neg, \cup\}$.

References

- [1] . ("Complete set of logical connectives").
- [2] . ("[F]unctional completeness of [a] set of logical operators").
- [3] . (Defines "expressively adequate", shortened to "adequate set of connectives" in a section heading.)
- [4] The term was originally restricted to *binary* operations, but since the end of the 20th century it is used more generally. .
- [5] Wernick, William (1942) "Complete Sets of Logical Functions," *Transactions of the American Mathematical Society* 51: 117–32. In his list on the last page of the article, Wernick does not distinguish between \leftarrow and \rightarrow , or between UNIQ-math-0-a68bc4e06c66b481-QINU and UNIQ-math-1-a68bc4e06c66b481-QINU .
- [6] "NAND Gate Operations" at <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nand.html>
- [7] "NOR Gate Operations" at <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nor.html>
- Wernick, William (1942) "Complete Sets of Logical Functions," *Transactions of the American Mathematical Society* 51: 117–32.

If and only if

$\leftrightarrow \Leftrightarrow \equiv$

Logical symbols

representing *iff*

In logic and related fields such as mathematics and philosophy, **if and only if** (shortened **iff**) is a biconditional logical connective between statements.

In that it is biconditional, the connective can be likened to the standard material conditional ("only if", equal to "if ... then") combined with its reverse ("if"); hence the name. The result is that the truth of either one of the connected statements requires the truth of the other, i.e., either both statements are true, or both are false. It is controversial whether the connective thus defined is properly rendered by the English "if and only if", with its pre-existing meaning. There is nothing to stop one from *stipulating* that we may read this connective as "only if and if", although this may lead to confusion.

In writing, phrases commonly used, with debatable propriety, as alternatives to P "if and only if" Q include *Q is necessary and sufficient for P*, *P is equivalent (or materially equivalent) to Q* (compare material implication), *P precisely if Q*, *P precisely (or exactly) when Q*, *P exactly in case Q*, and *P just in case Q*. Many authors regard "iff" as unsuitable in formal writing; others use it freely.^[citation needed]

In *logic formulae*, logical symbols are used instead of these phrases; see the discussion of notation.

Definition

The truth table of $p \leftrightarrow q$ is as follows:^[1]

Iff

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Note that it is equivalent to that produced by the XNOR gate, and opposite to that produced by the XOR gate.

Usage

Notation

The corresponding logical symbols are " \leftrightarrow ", " \Leftrightarrow " and " \equiv ", and sometimes "iff". These are usually treated as equivalent. However, some texts of mathematical logic (particularly those on first-order logic, rather than propositional logic) make a distinction between these, in which the first, \leftrightarrow , is used as a symbol in logic formulas, while \Leftrightarrow is used in reasoning about those logic formulas (e.g., in metalogic). In Łukasiewicz's notation, it is the prefix symbol 'E'.

Another term for this logical connective is exclusive nor.

Proofs

In most logical systems, one proves a statement of the form "P iff Q" by proving "if P, then Q" and "if Q, then P". Proving this pair of statements sometimes leads to a more natural proof, since there are not obvious conditions in which one would infer a biconditional directly. An alternative is to prove the disjunction "(P and Q) or (not-P and not-Q)", which itself can be inferred directly from either of its disjuncts—that is, because "iff" is truth-functional, "P iff Q" follows if P and Q have both been shown true, or both false.

Origin of iff

Usage of the abbreviation "iff" first appeared in print in John L. Kelley's 1955 book *General Topology*.^[2] Its invention is often credited to Paul Halmos, who wrote "I invented 'iff,' for 'if and only if'—but I could never believe I was really its first inventor."

Distinction from "if" and "only if"

1. "**If the fruit is an apple, then Madison will eat it.**" or "**Madison will eat the fruit if it is an apple.**"
(equivalent to "**Only if Madison will eat the fruit, is it an apple;**" or "**Madison will eat the fruit \leftarrow fruit is an apple**")

This states only that Madison will eat fruits that are apples. It does not, however, preclude the possibility that Madison might also have occasion to eat bananas. Maybe she will, maybe she will not—the sentence does not tell us. All that is known for certain is that she will eat any and all apples that she happens upon. That the fruit is an apple is a *sufficient* condition for Madison to eat the fruit.

2. "**Only if the fruit is an apple, will Madison eat it.**" or "**Madison will eat the fruit only if it is an apple.**"
(equivalent to "**If Madison will eat the fruit, then it is an apple**" or "**Madison will eat the fruit \rightarrow fruit is an apple**")

This states that the only fruit Madison will eat is an apple. It does not, however, preclude the possibility that Madison will refuse an apple if it is made available, in contrast with (1), which requires Madison to eat any available apple. In this case, that a given fruit is an apple is a *necessary* condition for Madison to be eating it. It is not a sufficient condition since Madison might not eat any and all apples she is given.

3. "**If and only if the fruit is an apple will Madison eat it**" or "**Madison will eat the fruit if and only if it is an apple**" or "**Madison will eat the fruit \leftrightarrow fruit is an apple**".

This, however, makes it quite clear that Madison will eat all and only those fruits that are apples. She will not leave any such fruit uneaten, and she will not eat any other type of fruit. That a given fruit is an apple is both a necessary and a sufficient condition for Madison to eat the fruit.

Sufficiency is the inverse of necessity. That is to say, given $P \rightarrow Q$ (i.e. if P then Q), P would be a sufficient condition for Q , and Q would be a necessary condition for P . Also, given $P \rightarrow Q$, it is true that $\neg Q \rightarrow \neg P$ (where \neg is the negation operator, i.e. "not"). This means that the relationship between P and Q , established by $P \rightarrow Q$, can be

expressed in the following, all equivalent, ways:

P is sufficient for Q

Q is necessary for P

$\neg Q$ is sufficient for $\neg P$

$\neg P$ is necessary for $\neg Q$

As an example, take (1), above, which states $P \rightarrow Q$, where P is "the fruit in question is an apple" and Q is "Madison will eat the fruit in question". The following are four equivalent ways of expressing this very relationship:

If the fruit in question is an apple, then Madison will eat it.

Only if Madison will eat the fruit in question, is it an apple.

If Madison will not eat the fruit in question, then it is not an apple.

Only if the fruit in question is not an apple, will Madison not eat it.

So we see that (2), above, can be restated in the form of *if...then* as "If Madison will eat the fruit in question, then it is an apple"; taking this in conjunction with (1), we find that (3) can be stated as "If the fruit in question is an apple, then Madison will eat it; AND if Madison will eat the fruit, then it is an apple".

Advanced considerations

Philosophical interpretation

A sentence that is composed of two other sentences joined by "iff" is called a *biconditional*. "Iff" joins two sentences to form a new sentence. It should not be confused with logical equivalence which is a description of a relation between two sentences. The biconditional " A iff B " uses the sentences A and B , describing a relation between the states of affairs which A and B describe. By contrast " A is logically equivalent to B " mentions both sentences: it describes a logical relation between those two sentences, and not a factual relation between whatever matters they describe. See use–mention distinction for more on the difference between *using* a sentence and *mentioning* it.

The distinction is a very confusing one, and has led many a philosopher Wikipedia:Avoid weasel words astray. Certainly it is the case that when A is logically equivalent to B , " A iff B " is true. But the converse does not hold. Reconsidering the sentence:

If and only if the fruit is an apple will Madison eat it.

There is clearly no logical equivalence between the two halves of this particular biconditional. For more on the distinction, see W. V. Quine's *Mathematical Logic*, Section 5.

One way of looking at " A if and only if B " is that it means " A if B " (B implies A) and " A only when B " (not B implies not A). "Not B implies not A " means A implies B , so then there is two way implication.

Definitions

In philosophy and logic, "iff" is used to indicate definitions, since definitions are supposed to be universally quantified biconditionals. In mathematics and elsewhere, however, the word "if" is normally used in definitions, rather than "iff". This is due to the observation that "if" in the English language has a definitional meaning, separate from its meaning as a propositional connective. This separate meaning can be explained by noting that a definition (for instance: A group is "abelian" if it satisfies the commutative law; or: A grape is a "raisin" if it is well dried) is not an equivalence to be proved, but a rule for interpreting the term defined.

Examples

Here are some examples of true statements that use "iff" - true biconditionals (the first is an example of a definition, so it would normally have been written with "if"):

- A person is a bachelor *iff* that person is a marriageable man who has never married.
- "Snow is white" in English is true *iff* "Schnee ist weiß" in German is true.
- For any p , q , and r : $(p \& q) \& r$ iff $p \& (q \& r)$. (Since this is written using variables and "&", the statement would usually be written using " \leftrightarrow ", or one of the other symbols used to write biconditionals, in place of "iff").
- For any real numbers x and y , $x=y+1$ iff $y=x-1$.
- A subset containing n elements of an n -dimensional vector space is linearly independent iff it spans the vector space.
- The triangular number $\frac{n(n+1)}{2}$ is an even perfect number iff $n = 2^p-1$ is a Mersenne prime, with p being a prime number. As of Feb 2013, only 48 such even perfect numbers and Mersenne primes have been discovered.
- $y(x)$ is a solution to the differential equation $y=f(x,y)$ if and only if the curve associated with $y(x)$ is an integral curve of the direction field associated with $y=f(x,y)$.

Analogs

Other words are also sometimes emphasized in the same way by repeating the last letter; for example *orr* for "Or and only Or" (the exclusive disjunction).

The statement "(A iff B)" is equivalent to the statement "(not A or B) and (not B or A)," and is also equivalent to the statement "(not A and not B) or (A and B)".

It is also equivalent to: not[(A or B) and (not A or not B)],

or more simply:

$$\neg [(\neg A \vee \neg B) \wedge (A \vee B)]$$

which converts into

$$[(\neg A \wedge \neg B) \vee (A \wedge B)]$$

and

$$[(\neg A \vee B) \wedge (A \vee \neg B)]$$

which were given in verbal interpretations above.

More general usage

Iff is used outside the field of logic, wherever logic is applied, especially in mathematical discussions. It has the same meaning as above: it is an abbreviation for *if and only if*, indicating that one statement is both necessary and sufficient for the other. This is an example of mathematical jargon. (However, as noted above, *if*, rather than *iff*, is more often used in statements of definition.)

The elements of X are *all and only* the elements of Y is used to mean: "for any z in the domain of discourse, z is in X if and only if z is in Y ."

Notes

- [1] $p \Leftrightarrow q$ (<http://www.wolframalpha.com/input/?i=p+<=>+q>). WolframAlpha
- [2] *General Topology*, reissue ISBN 978-0-387-90125-1

Inclusion (Boolean algebra)

In Boolean algebra (structure), the **inclusion relation** $a \leq b$ is defined as $ab' = 0$ and is the Boolean analogue to the subset relation in set theory. Inclusion is a partial order.

The inclusion relation $a < b$ can be expressed in many ways:

- $a < b$
- $ab' = 0$
- $a' + b = 1$
- $b' < a'$
- $a + b = b$
- $ab = a$

The inclusion relation has a natural interpretation in various Boolean algebras: in the subset algebra, the subset relation; in arithmetic Boolean algebra, divisibility; in the algebra of propositions, material implication; in the two-element algebra, the set { (0,0), (0,1), (1,1) }.

Some useful properties of the inclusion relation are:

- $a \leq a + b$
- $ab \leq a$

The inclusion relation may be used to define **Boolean intervals** such that $a \leq x \leq b$. A Boolean algebra whose carrier set is restricted to the elements in an interval is itself a Boolean algebra.

References

- Frank Markham Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd edition, 2003, p. 52

Indicative conditional

In natural languages, an **indicative conditional**^{[1][2]} is the logical operation given by statements of the form "If A then B". Unlike the material conditional, an indicative conditional does not have a stipulated definition. The philosophical literature on this operation is broad, and no clear consensus has been reached.

Distinctions between the material conditional and the indicative conditional

The material conditional does not always function in accordance with everyday if-then reasoning. Therefore there are drawbacks with using the material conditional to represent if-then statements.

One problem is that the material conditional allows implications to be true even when the antecedent is irrelevant to the consequent. For example, it's commonly accepted that the sun is made of gas, on one hand, and that 3 is a prime number, on the other. The standard definition of implication allows us to conclude that, since the sun is made of gas, 3 is a prime number. This is arguably synonymous to the following: the sun's being made of gas makes 3 be a prime number. Many people intuitively think that this is false, because the sun and the number three simply have nothing to do with one another. Logicians have tried to address this concern by developing alternative logics, e.g., relevant logic.

For a related problem, see vacuous truth.

Another issue is that the material conditional is not designed to deal with counterfactuals and other cases that people often find in if-then reasoning. This has inspired people to develop modal logic.

A further problem is that the material conditional is such that P AND $\neg P \rightarrow Q$, regardless of what Q is taken to mean. That is, a contradiction implies that absolutely everything is true. Logicians concerned with this have tried to develop paraconsistent logics.

Psychology and indicative conditionals

Most behavioral experiments on conditionals in the psychology of reasoning have been carried out with indicative conditionals, causal conditionals, and counterfactual conditionals. People readily make the modus ponens inference, that is, given *if A then B*, and given A, they conclude B, but only about half of participants in experiments make the modus tollens inference, that is, given *if A then B*, and given *not-B*, only about half of participants conclude *not-A*, the remainder say that nothing follows (Evans *et al.*, 1993). When participants are given counterfactual conditionals, they make both the modus ponens and the modus tollens inferences (Byrne, 2005).

References

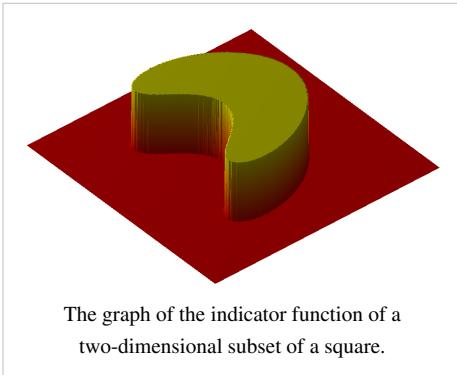
- [1] Stalnaker, R, *Philosophia* (1975)
- [2] Ellis, B, *Australasian Journal of Philosophy* (1984)

Further reading

- Byrne, R.M.J. (2005). *The Rational Imagination: How People Create Counterfactual Alternatives to Reality*. Cambridge, MA: MIT Press.
- Edgington, Dorothy. (2006). "Conditionals". *The Stanford Encyclopedia of Philosophy*, Edward Zalta (ed.). <http://plato.stanford.edu/entries/conditionals/>.
- Evans, J. St. B. T., Newstead, S. and Byrne, R. M. J. (1993). *Human Reasoning: The Psychology of Deduction*. Hove, Psychology Press.

Indicator function

In mathematics, an **indicator function** or a **characteristic function** is a function defined on a set X that indicates membership of an element in a subset A of X , having the value 1 for all elements of A and the value 0 for all elements of X not in A .



Definition

The indicator function of a subset A of a set X is a function

$$\mathbf{1}_A : X \rightarrow \{0, 1\}$$

defined as

$$\mathbf{1}_A(x) := \begin{cases} 1 & \text{if } x \in A, \\ 0 & \text{if } x \notin A. \end{cases}$$

The Iverson bracket allows the equivalent notation, $[x \in A]$, to be used instead of $\mathbf{1}_A(x)$.

The function $\mathbf{1}_A$ is sometimes denoted $\mathbf{1}_{A \in A}$, χ_A or \mathbf{I}_A or even just A . (The Greek letter χ appears because it is the initial letter of the Greek word *characteristic*.)

Remark on notation and terminology

- The notation $\mathbf{1}_A$ may signify the identity function.[Wikipedia:Please clarify](#)
- The notation χ_A may signify the characteristic function in convex analysis.[Wikipedia:Please clarify](#)

A related concept in statistics is that of a dummy variable (this must not be confused with "dummy variables" as that term is usually used in mathematics, also called a bound variable).

The term "characteristic function" has an unrelated meaning in probability theory. For this reason, probabilists use the term **indicator function** for the function defined here almost exclusively, while mathematicians in other fields are more likely to use the term **characteristic function** to describe the function which indicates membership in a set.

Basic properties

The *indicator* or *characteristic* function of a subset A of some set X , maps elements of X to the range $\{0,1\}$.

This mapping is surjective only when A is a non-empty proper subset of X . If $A \equiv X$, then $\mathbf{1}_A = 1$. By a similar argument, if $A \equiv \emptyset$ then $\mathbf{1}_A = 0$.

In the following, the dot represents multiplication, $1 \cdot 1 = 1$, $1 \cdot 0 = 0$ etc. "+" and "-" represent addition and subtraction. " \cap " and " \cup " is intersection and union, respectively.

If A and B are two subsets of X , then

$$\mathbf{1}_{A \cap B} = \min\{\mathbf{1}_A, \mathbf{1}_B\} = \mathbf{1}_A \cdot \mathbf{1}_B,$$

$$\mathbf{1}_{A \cup B} = \max\{\mathbf{1}_A, \mathbf{1}_B\} = \mathbf{1}_A + \mathbf{1}_B - \mathbf{1}_A \cdot \mathbf{1}_B,$$

and the indicator function of the complement of A i.e. A^C is:

$$\mathbf{1}_{A^C} = 1 - \mathbf{1}_A.$$

More generally, suppose A_1, \dots, A_n is a collection of subsets of X . For any $x \in X$:

$$\prod_{k \in I} (1 - \mathbf{1}_{A_k}(x))$$

is clearly a product of 0s and 1s. This product has the value 1 at precisely those $x \in X$ which belong to none of the sets A_k and is 0 otherwise. That is

$$\prod_{k \in I} (1 - \mathbf{1}_{A_k}) = \mathbf{1}_{X - \bigcup_k A_k} = 1 - \mathbf{1}_{\bigcup_k A_k}.$$

Expanding the product on the left hand side,

$$\mathbf{1}_{\bigcup_k A_k} = 1 - \sum_{F \subseteq \{1, 2, \dots, n\}} (-1)^{|F|} \mathbf{1}_{\bigcap_F A_k} = \sum_{\emptyset \neq F \subseteq \{1, 2, \dots, n\}} (-1)^{|F|+1} \mathbf{1}_{\bigcap_F A_k}$$

where $|F|$ is the cardinality of F . This is one form of the principle of inclusion-exclusion.

As suggested by the previous example, the indicator function is a useful notational device in combinatorics. The notation is used in other places as well, for instance in probability theory: if X is a probability space with probability measure \mathbb{P} and A is a measurable set, then $\mathbf{1}_A$ becomes a random variable whose expected value is equal to the probability of A :

$$E(\mathbf{1}_A) = \int_X \mathbf{1}_A(x) d\mathbb{P} = \int_A d\mathbb{P} = P(A).$$

This identity is used in a simple proof of Markov's inequality.

In many cases, such as order theory, the inverse of the indicator function may be defined. This is commonly called the generalized Möbius function, as a generalization of the inverse of the indicator function in elementary number theory, the Möbius function. (See paragraph below about the use of the inverse in classical recursion theory.)

Mean, variance and covariance

Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with $A \in \mathcal{F}$, the indicator random variable $\mathbf{1}_A : \Omega \rightarrow \mathbb{R}$ is defined by $\mathbf{1}_A(\omega) = 1$ if $\omega \in A$, otherwise $\mathbf{1}_A(\omega) = 0$.

Mean: $E(\mathbf{1}_A(\omega)) = P(A)$

Variance: $\text{Var}(\mathbf{1}_A(\omega)) = P(A)(1 - P(A))$

Covariance: $\text{Cov}(\mathbf{1}_A(\omega), \mathbf{1}_B(\omega)) = P(A \cap B) - P(A)P(B)$

Characteristic function in recursion theory, Gödel's and Kleene's *representing function*

Kurt Gödel described the *representing function* in his 1934 paper "On Undecidable Propositions of Formal Mathematical Systems". (The paper appears on pp. 41–74 in Martin Davis ed. *The Undecidable*):

"There shall correspond to each class or relation R a representing function $\varphi(x_1, \dots, x_n) = 0$ if $R(x_1, \dots, x_n)$ and $\varphi(x_1, \dots, x_n) = 1$ if $\sim R(x_1, \dots, x_n)$." (p. 42; the " \sim " indicates logical inversion i.e. "NOT")

Stephen Kleene (1952) (p. 227) offers up the same definition in the context of the primitive recursive functions as a function φ of a predicate P , takes on values 0 if the predicate is true and 1 if the predicate is false.

For example, because the product of characteristic functions $\varphi_1 * \varphi_2 * \dots * \varphi_n = 0$ whenever any one of the functions equals 0, it plays the role of logical OR: IF $\varphi_1 = 0$ OR $\varphi_2 = 0$ OR \dots OR $\varphi_n = 0$ THEN their product is 0. What appears to the modern reader as the representing function's logical-inversion, i.e. the representing function is 0 when the function R is "true" or satisfied", plays a useful role in Kleene's definition of the logical functions OR, AND, and IMPLY (p. 228), the bounded- (p. 228) and unbounded- (p. 279ff) mu operators (Kleene (1952)) and the CASE function (p. 229).

Characteristic function in fuzzy set theory

In classical mathematics, characteristic functions of sets only take values 1 (members) or 0 (non-members). In fuzzy set theory, characteristic functions are generalized to take value in the real unit interval [0, 1], or more generally, in some algebra or structure (usually required to be at least a poset or lattice). Such generalized characteristic functions are more usually called membership functions, and the corresponding "sets" are called *fuzzy* sets. Fuzzy sets model the gradual change in the membership degree seen in many real-world predicates like "tall", "warm", etc.

Derivatives of the indicator function

A particular indicator function, which is very well known, is the Heaviside step function. The Heaviside step function is the indicator function of the one-dimensional positive half-line, i.e. the domain $[0, \infty)$. It is well-known that the distributional derivative of the Heaviside step function, indicated by $H(x)$, is equal to the Dirac delta function, i.e.

$$\delta(x) = \frac{dH(x)}{dx},$$

with the following property:

$$\int_{-\infty}^{\infty} f(x) \delta(x) dx = f(0).$$

The derivative of the Heaviside step function can be seen as the 'inward normal derivative' at the 'boundary' of the domain given by the positive half-line. In higher dimensions, the derivative naturally generalises to the inward normal derivative, while the Heaviside step function naturally generalises to the indicator function of some domain D . The surface of D will be denoted by S . Proceeding, it can be derived that the inward normal derivative of the indicator gives rise to a 'surface delta function', which can be indicated by $\delta_S(\mathbf{x})$:

$$\delta_S(\mathbf{x}) = -\mathbf{n}_x \cdot \nabla_x \mathbf{1}_{\mathbf{x} \in D}$$

where n is the outward normal of the surface S . This 'surface delta function' has the following property:

$$-\int_{\mathbf{R}^n} f(\mathbf{x}) \mathbf{n}_x \cdot \nabla_x \mathbf{1}_{\mathbf{x} \in D} d^n \mathbf{x} = \oint_S f(\beta) d^{n-1} \beta.$$

By setting the function f equal to one, it follows that the inward normal derivative of the indicator integrates to the numerical value of the surface area S .

Notes

References

- Folland, G.B. (1999). *Real Analysis: Modern Techniques and Their Applications* (Second ed.). John Wiley & Sons, Inc.
- Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001). "Section 5.2: Indicator random variables". *Introduction to Algorithms* (Second Edition ed.). MIT Press and McGraw-Hill. pp. 94–99. ISBN 0-262-03293-7.
- Davis, Martin, ed. (1965). *The Undecidable*. New York: Raven Press Books, Ltd.
- Kleene, Stephen (1971) [1952]. *Introduction to Metamathematics* (Sixth Reprint with corrections). Netherlands: Wolters-Noordhoff Publishing and North Holland Publishing Company.
- Boolos, George; Burgess, John P.; Jeffrey, Richard C. (2002). *Computability and Logic*. Cambridge UK: Cambridge University Press. ISBN 0-521-00758-5.
- Zadeh, Lotfi A. (June 1965). "Fuzzy sets" (<http://www-bisc.cs.berkeley.edu/zadeh/papers/Fuzzy Sets-1965.pdf>) (PDF). *Information and Control* 8 (3): 338–353. doi: 10.1016/S0019-9958(65)90241-X ([http://dx.doi.org/10.1016/S0019-9958\(65\)90241-X](http://dx.doi.org/10.1016/S0019-9958(65)90241-X)).

- Goguen, Joseph (1967). "L-fuzzy sets". *Journal of Mathematical Analysis and Applications* **18** (1): 145–174. doi: 10.1016/0022-247X(67)90189-8 ([http://dx.doi.org/10.1016/0022-247X\(67\)90189-8](http://dx.doi.org/10.1016/0022-247X(67)90189-8)).

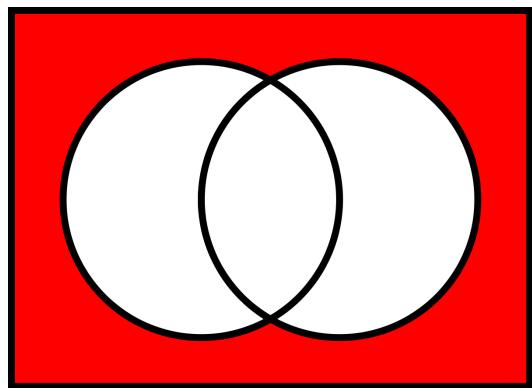
Logical NOR

In boolean logic, **logical nor** or **joint denial** is a truth-functional operator which produces a result that is the negation of logical or. That is, a sentence of the form $(p \text{ NOR } q)$ is true precisely when neither p nor q is true—i.e. when both of p and q are *false*. In grammar, **nor** is a coordinating conjunction.

The NOR operator is also known as **Peirce's arrow** — Charles Sanders Peirce introduced the symbol \downarrow for it, and demonstrated that the logical NOR is completely expressible: by combining uses of the logical NOR it is possible to express any logical operation on two variables. Thus, as with its dual, the NAND operator (a.k.a. the Sheffer stroke — symbolized as either \mid or $/$), NOR can be used by itself, without any other logical operator, to constitute a logical formal system (making NOR functionally complete). It is also known as Quine's dagger (his symbol was \dagger), the **ampheck** (from Greek αμφηκης, cutting both ways; compare *amphi-*) by Peirce,^[1] or "neither-nor".

One way of expressing $p \text{ NOR } q$ is $\overline{p \vee q}$, where the symbol \vee signifies OR and the bar signifies the negation of the expression under it: in essence, simply $\neg(p \vee q)$. Other ways of expressing $p \text{ NOR } q$ are Xpq , and $\overline{p + q}$.

The computer used in the spacecraft that first carried humans to the moon, the Apollo Guidance Computer, was constructed entirely using NOR gates with three inputs. [citation needed]



Venn diagram of $A \downarrow B$

(nor part in red)

Definition

The **NOR operation** is a logical operation on two logical values, typically the values of two propositions, that produces a value of *true* if and only if both operands are *false*. In other words, it produces a value of *false* if and only if at least one operand is *true*.

Truth table

The truth table of **A NOR B** (also written as $A \downarrow B$) is as follows:

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

Properties

Logical NOR does not possess any of the five qualities (truth-preserving, false-preserving, linear, monotonic, self-dual) required to be absent from at least one member of a set of functionally complete operators. Thus, the set containing only NOR suffices as a complete set.

Introduction, elimination, and equivalencies

NOR has the interesting feature that all other logical operators can be expressed by interlaced NOR operations. The logical NAND operator also has this ability.

The logical NOR \downarrow is the negation of the disjunction:

$P \downarrow Q$	\Leftrightarrow	$\neg(P \vee Q)$
	\Leftrightarrow	

Expressed in terms of NOR \downarrow , the usual operators of propositional logic are:

$\neg P$	\Leftrightarrow	$P \downarrow P$
	\Leftrightarrow	
$P \rightarrow Q$	\Leftrightarrow	$((P \downarrow P) \downarrow Q)$
	\Leftrightarrow	
$P \vee Q$	\Leftrightarrow	$(P \downarrow Q)$
	\Leftrightarrow	
$P \wedge Q$	\Leftrightarrow	$(P \downarrow P) \downarrow (Q \downarrow Q)$
	\Leftrightarrow	
$P \vee Q$	\Leftrightarrow	$(P \downarrow Q) \downarrow (P \downarrow Q)$
	\Leftrightarrow	

References

- [1] C.S. Peirce, CP 4.264

Logical biconditional

In logic and mathematics, the **logical biconditional** (sometimes known as the **material biconditional**) is the logical connective of two statements asserting " p if and only if q ", where q is a *hypothesis* (or *antecedent*) and p is a *conclusion* (or *consequent*).^[1] This is often abbreviated p iff q . The operator is denoted using a doubleheaded arrow (\leftrightarrow), a prefixed E (Epq), an equality sign ($=$), an equivalence sign (\equiv), or EQV . It is logically equivalent to $(p \rightarrow q) \wedge (q \rightarrow p)$, or the XNOR (exclusive nor) boolean operator. It is equivalent to "(not p or q) and (not q or p)". It is also logically equivalent to "(p and q) or (not p and not q)", meaning "both or neither".

The only difference from material conditional is the case when the hypothesis is false but the conclusion is true. In that case, in the conditional, the result is true, yet in the biconditional the result is false.

In the conceptual interpretation, $a = b$ means "All a 's are b 's and all b 's are a 's"; in other words, the sets a and b coincide: they are identical. This does not mean that the concepts have the same meaning. Examples: "triangle" and "trilateral", "equiangular triangle" and "equilateral triangle". The antecedent is the *subject* and the consequent is the *predicate* of a universal affirmative proposition.

In the propositional interpretation, $a \Leftrightarrow b$ means that a implies b and b implies a ; in other words, that the propositions are equivalent, that is to say, either true or false at the same time. This does not mean that they have the same meaning. Example: "The triangle ABC has two equal sides", and "The triangle ABC has two equal angles". The antecedent is the *premise* or the *cause* and the consequent is the *consequence*. When an implication is translated by a *hypothetical* (or *conditional*) judgment the antecedent is called the *hypothesis* (or the *condition*) and the consequent is called the *thesis*.

A common way of demonstrating a biconditional is to use its equivalence to the conjunction of two converse conditionals, demonstrating these separately.

When both members of the biconditional are propositions, it can be separated into two conditionals, of which one is called a *theorem* and the other its *reciprocal*.^[citation needed] Thus whenever a theorem and its reciprocal are true we have a biconditional. A simple theorem gives rise to an implication whose antecedent is the *hypothesis* and whose consequent is the *thesis* of the theorem.

It is often said that the hypothesis is the *sufficient condition* of the thesis, and the thesis the *necessary condition* of the hypothesis; that is to say, it is sufficient that the hypothesis be true for the thesis to be true; while it is necessary that the thesis be true for the hypothesis to be true also. When a theorem and its reciprocal are true we say that its hypothesis is the necessary and sufficient condition of the thesis; that is to say, that it is at the same time both cause and consequence.

Definition

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 and only if both operands are false or both operands are true.

Truth table

The truth table for $A \leftrightarrow B$ (also written as $A \sqsubseteq B$, $A = B$, or $A EQ B$) is as follows:

INPUT		OUTPUT
A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

More than two statements combined by \leftrightarrow are ambiguous:

$x_1 \leftrightarrow x_2 \leftrightarrow x_3 \leftrightarrow \dots \leftrightarrow x_n$ may be meant as $((x_1 \leftrightarrow x_2) \leftrightarrow x_3) \leftrightarrow \dots \leftrightarrow x_n$,

or may be used to say that all x_i are *together true or together false*: $(x_1 \wedge \dots \wedge x_n) \vee (\neg x_1 \wedge \dots \wedge \neg x_n)$

Only for zero or two arguments this is the same.

The following truth tables show the same bit pattern only in the line with no argument and in the lines with two arguments:

()	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15	x16
(A)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(B)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
(AB)	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
(C)	0	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0
(A C)	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1
(BC)	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
(ABC)	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
(D)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
(A D)	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
(B D)	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0
(AB D)	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0
(CD)	0	0	0	0	1	1	0	0	0	0	1	1	0	0	0	0
(A CD)	0	1	0	0	1	0	0	1	0	1	0	0	1	0	0	1
(BCD)	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
(ABCD)	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0

$x_1 \leftrightarrow \dots \leftrightarrow x_n$
meant as equivalent to
 $\neg(x_1 \oplus \dots \oplus \neg x_n)$

The central Venn diagram below,
and line (ABC) in this matrix
represent the same operation.

The left Venn diagram below, and the lines (AB) in these matrices represent the same operation.

Venn diagrams

Red areas stand for true (as in  for *and*).

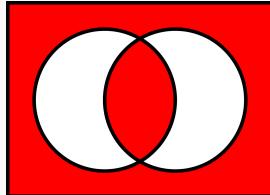
$$x_1 \leftrightarrow \dots \leftrightarrow x_n$$

meant as shorthand for

$$(x_1 \wedge \dots \wedge x_n)$$

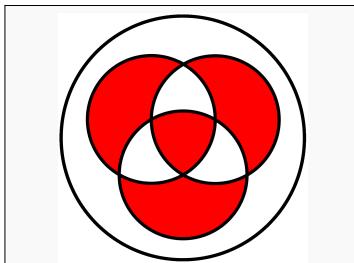
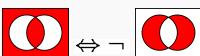
$$\vee (\neg x_1 \wedge \dots \wedge \neg x_n)$$

The Venn diagram directly below, and line (ABC) in this matrix represent the same operation.



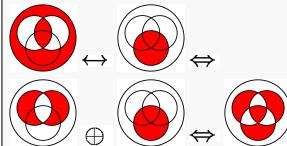
The biconditional of two statements
is the negation of the exclusive or:

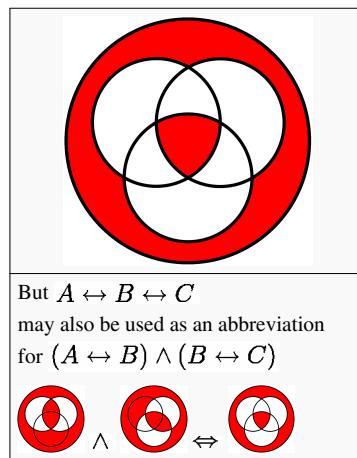
$$A \leftrightarrow B \Leftrightarrow \neg(A \oplus B)$$



The biconditional and the exclusive or of three statements give the same result:

$$A \leftrightarrow B \leftrightarrow C \Leftrightarrow A \oplus B \oplus C$$





Properties

commutativity: yes

$A \leftrightarrow B$	\Leftrightarrow	$B \leftrightarrow A$
	\Leftrightarrow	

associativity: yes

A	\leftrightarrow	$(B \leftrightarrow C)$	\leftrightarrow			$(A \leftrightarrow B)$	\leftrightarrow	C
	\leftrightarrow		\leftrightarrow		\leftrightarrow		\leftrightarrow	

distributivity: Biconditional doesn't distribute over any binary function (not even itself), but logical disjunction (see there) distributes over biconditional.

idempotency: no

A	\leftrightarrow	A	\leftrightarrow	1	\Rightarrow	A
	\leftrightarrow		\leftrightarrow		\Rightarrow	

monotonicity: no

$A \rightarrow B$	\Rightarrow			$(A \leftrightarrow C)$	\rightarrow	$(B \leftrightarrow C)$
	\Rightarrow		\Leftrightarrow		\rightarrow	

truth-preserving: yes

When all inputs are true, the output is true.

$A \wedge B$	\Rightarrow	$A \leftrightarrow B$
	\Rightarrow	

falsehood-preserving: no

When all inputs are false, the output is not false.

$A \leftrightarrow B$	$\not\Rightarrow$	$A \vee B$
	$\not\Rightarrow$	

Walsh spectrum: (2,0,0,2)

Nonlinearity: 0 (the function is linear)

Rules of inference

Like all connectives in first-order logic, the biconditional has rules of inference that govern its use in formal proofs.

Biconditional introduction

Biconditional introduction allows you to infer that, if B follows from A, and A follows from B, then A if and only if B.

For example, from the statements "if I'm breathing, then I'm alive" and "if I'm alive, then I'm breathing", it can be inferred that "I'm breathing if and only if I'm alive" or, equally inferrable, "I'm alive if and only if I'm breathing."

$$\begin{array}{l} B \rightarrow A \\ A \rightarrow B \\ \hline \therefore A \leftrightarrow B \end{array}$$

$$\begin{array}{l} B \rightarrow A \\ A \rightarrow B \\ \hline \therefore B \leftrightarrow A \end{array}$$

Biconditional elimination

Biconditional elimination allows one to infer a conditional from a biconditional: if $(A \leftrightarrow B)$ is true, then one may infer one direction of the biconditional, $(A \rightarrow B)$ and $(B \rightarrow A)$.

For example, if it's true that I'm breathing if and only if I'm alive, then it's true that if I'm breathing, I'm alive; likewise, it's true that if I'm alive, I'm breathing.

Formally:

$$\begin{array}{l} (A \leftrightarrow B) \\ \hline \therefore (A \rightarrow B) \end{array}$$

also

$$\begin{array}{l} (A \leftrightarrow B) \\ \hline \therefore (B \rightarrow A) \end{array}$$

Colloquial usage

One unambiguous way of stating a biconditional in plain English is of the form " b if a and a if b ". Another is " a if and only if b ". Slightly more formally, one could say " b implies a and a implies b ". The plain English "if" may sometimes be used as a biconditional. One must weigh context heavily.

For example, "I'll buy you a new wallet if you need one" may be meant as a biconditional, since the speaker doesn't intend a valid outcome to be buying the wallet whether or not the wallet is needed (as in a conditional). However, "it is cloudy if it is raining" is not meant as a biconditional, since it can be cloudy while not raining.

Notes

[1] Handbook of Logic, page 81

References

- Brennan, Joseph G. Handbook of Logic, 2nd Edition (<http://www.archive.org/stream/handbookoflogics012674mbp#page/n90/mode/1up>). Harper & Row. 1961

This article incorporates material from Biconditional on PlanetMath, which is licensed under the Creative Commons Attribution/Share-Alike License.

Logical conjunction

In logic and mathematics, a two-place logical operator **and**, also known as **logical conjunction**,^[1] results in *true* if both of its operands are true, otherwise the value of *false*.

The analogue of conjunction for a (possibly infinite) family of statements is universal quantification, which is part of predicate logic.

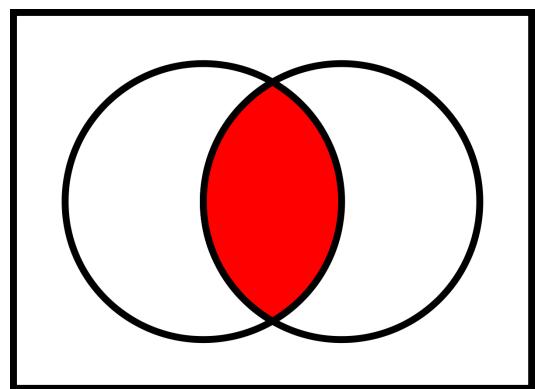
Notation

And is usually expressed with the prefix operator **K**, or an infix operator. In mathematics and logic, the infix operator is usually \sqcap ; in electronics \cdot ; and in programming languages, **&** or **and**. Some programming languages have a related control structure, the short-circuit and, written **&&**, **and then**, etc.

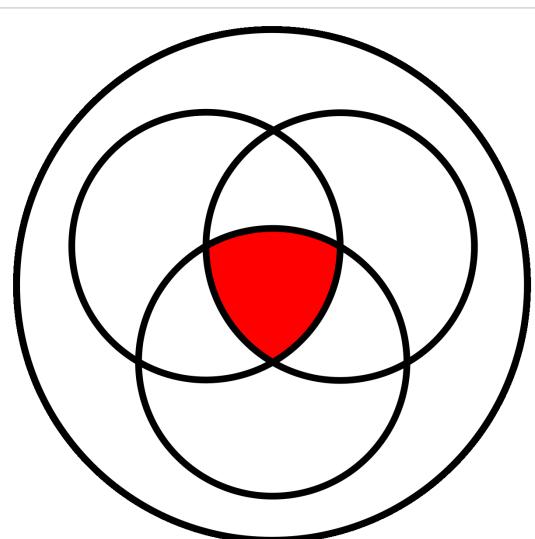
Definition

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

The conjunctive identity is 1, which is to say that AND-ing an expression with 1 will never change the value of the expression. In keeping with the concept of vacuous truth, when conjunction is defined as an operator or function of arbitrary arity, the empty conjunction (AND-ing over an empty set of operands) is often defined as having the result 1.



Venn diagram of $A \wedge B$



Venn diagram of $A \wedge B \wedge C$

Truth table

The truth table of $A \wedge B$:

$($	$)$	A	B	C	D	$(A \wedge B)$	$(B \wedge C)$	$(C \wedge D)$	$(A \wedge C)$	$(B \wedge D)$	$(A \wedge D)$	$(A \wedge B \wedge C)$	$(A \wedge B \wedge D)$	$(A \wedge C \wedge D)$	$(B \wedge C \wedge D)$	$(A \wedge B \wedge C \wedge D)$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Conjunctions of the arguments on the left
The true bits form a Sierpinski triangle

INPUT		OUTPUT
A	B	$A \wedge B$
T	T	T
T	F	F
F	T	F
F	F	F

Introduction and elimination rules

As a rule of inference, conjunction introduction is a classically valid, simple argument form. The argument form has two premises, A and B . Intuitively, it permits the inference of their conjunction.

$A,$

$B.$

Therefore, A and B .

or in logical operator notation:

$A,$

B

$\vdash A \wedge B$

Here is an example of an argument that fits the form *conjunction introduction*:

Bob likes apples.

Bob likes oranges.

Therefore, Bob likes apples and oranges.

Conjunction elimination is another classically valid, simple argument form. Intuitively, it permits the inference from any conjunction of either element of that conjunction.

A and $B.$

Therefore, $A.$

...or alternately,

A and $B.$

Therefore, B .

In logical operator notation:

$$A \wedge B$$

$$\vdash A$$

...or alternately,

$$A \wedge B$$

$$\vdash B$$

Properties

commutativity: yes

$A \wedge B$	\Leftrightarrow	$B \wedge A$
	\Leftrightarrow	

associativity: yes

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

distributivity: with various operations, especially with *or*

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

others

with exclusive or:

A	\wedge	$(B \oplus C)$	\Leftrightarrow			$(A \wedge B)$	\oplus	$(A \wedge C)$
	\wedge		\Leftrightarrow		\Leftrightarrow		\oplus	

with material nonimplication:

A	\wedge	$(B \rightarrow C)$	\Leftrightarrow			$(A \wedge B)$	\rightarrow	$(A \wedge C)$
	\wedge		\Leftrightarrow		\Leftrightarrow		\rightarrow	

with itself:

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

idempotency: yes

A	\wedge	A	\Leftrightarrow	A
	\wedge		\Leftrightarrow	

monotonicity: yes

$A \rightarrow B$	\Rightarrow			$(A \wedge C) \rightarrow (B \wedge C)$	
	\Rightarrow		\Leftrightarrow		\rightarrow

truth-preserving: yes

When all inputs are true, the output is true.

$A \wedge B$	\Rightarrow	$A \wedge B$
	\Rightarrow	
		(to be tested)

falsehood-preserving: yes

When all inputs are false, the output is false.

$A \wedge B$	\Rightarrow	$A \vee B$
	\Rightarrow	
		(to be tested)

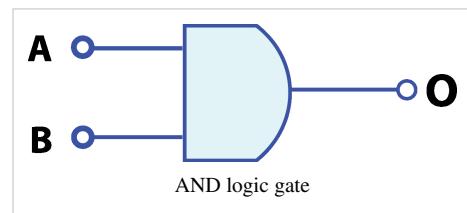
Walsh spectrum: (1,-1,-1,1)

Nonlinearity: 1 (the function is bent)

If using binary values for true (1) and false (0), then *logical conjunction* works exactly like normal arithmetic multiplication.

Applications in computer engineering

In high-level computer programming and digital electronics, logical conjunction is commonly represented by an infix operator, usually as a keyword such as "AND", an algebraic multiplication, or the ampersand symbol "&". Many languages also provide short-circuit control structures corresponding to logical conjunction.



Logical conjunction is often used for bitwise operations, where 0 corresponds to false and 1 to true:

- 0 AND 0 = 0,
- 0 AND 1 = 0,
- 1 AND 0 = 0,
- 1 AND 1 = 1.

The operation can also be applied to two binary words viewed as bitstrings of equal length, by taking the bitwise AND of each pair of bits at corresponding positions. For example:

- 11000110 AND 10100011 = 10000010.

This can be used to select part of a bitstring using a bit mask. For example, `10011101 AND 00001000 = 00001000` extracts the fifth bit of an 8-bit bitstring.

In computer networking, bit masks are used to derive the network address of a subnet within an existing network from a given IP address, by ANDing the IP address and the subnet mask.

Logical conjunction "AND" is also used in SQL operations to form database queries.

The Curry-Howard correspondence relates logical conjunction to product types.

Set-theoretic correspondence

The membership of an element of an intersection set in set theory is defined in terms of a logical conjunction: $x \in A \cap B$ if and only if $(x \in A) \wedge (x \in B)$. Through this correspondence, set-theoretic intersection shares several properties with logical conjunction, such as associativity, commutativity, and idempotence.

Natural language

The logical conjunction *and* in logic is related to, but not the same as, the grammatical conjunction *and* in natural languages.

English "and" has properties not captured by logical conjunction. For example, "and" sometimes implies order. For example, "They got married and had a child" in common discourse means that the marriage came before the child. The word "and" can also imply a partition of a thing into parts, as "The American flag is red, white, and blue." Here it is not meant that the flag is *at once* red, white, and blue, but rather that it has a part of each color.

External links

- Hazewinkel, Michiel, ed. (2001), "Conjunction" ^[2], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Wolfram MathWorld: Conjunction ^[3]

References

- [1] Moore and Parker, *Critical Thinking*
- [2] <http://www.encyclopediaofmath.org/index.php?title=p/c025080>
- [3] <http://mathworld.wolfram.com/Conjunction.html>

Logical disjunction

In logic and mathematics, **or** is a truth-functional operator also known as **(inclusive) disjunction** and **alternation**. The logical connective that represents this operator is also known as "or", and typically written as \vee or $+$. The "or" operator produces a result of true whenever *one or more* of its operands are true. For example, in this context, " A or B " is true if A is true, or if B is true, or if both A and B are true. In grammar, **or** is a coordinating conjunction. An operand of a disjunction is called a **disjunct**.

The "or" operator differs from the exclusive or in that the latter returns false when both of its inputs are true, while "or" returns true. In ordinary language, outside of contexts such as formal logic, mathematics and programming, "or" sometimes has the meaning of "exclusive or". For example, "Please ring me or send an email" likely means "do one or the other, but not both". On the other hand, "Her grades are so good that she's either very bright or studies hard" allows for the possibility that the person is both bright and works hard. In other words, in ordinary language "or" can mean inclusive or exclusive or. Usually the intended meaning is clear from the context.

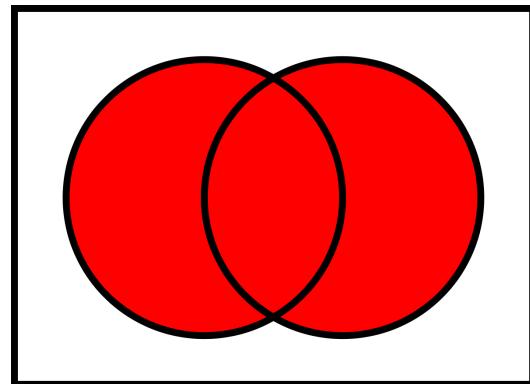
Notation

Or is usually expressed with the prefix operator **A**, or with an infix operator. In mathematics and logic, the infix operator is usually \sqcup ; in electronics, $+$; and in programming languages, **I** or **or**. Some programming languages have a related control structure, the short-circuit or, written **||**, **or else**, etc.

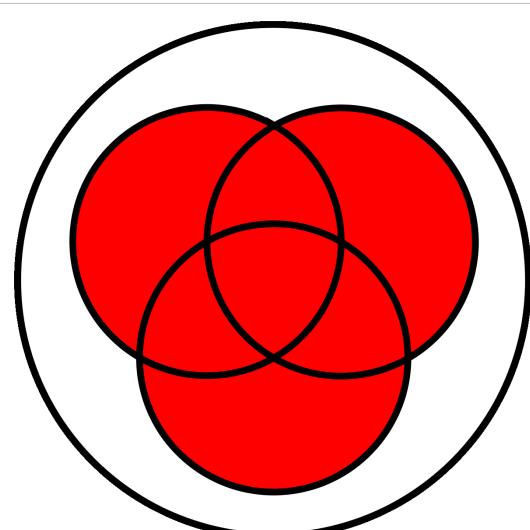
Definition

Logical disjunction is an operation on two logical values, typically the values of two propositions, that produces a value of *false* if and only if both of its operands are *false*. More generally a disjunction is a logical formula that can have one or more literals separated only by ORs. A single literal is often considered to be a degenerate disjunction.

The disjunctive identity is 0, which is to say that OR-ing an expression with 0 will never change the value of the expression. In keeping with the concept of vacuous truth, when disjunction is defined as an operator or function of arbitrary arity, the empty disjunction (OR-ing over an empty set of operands) is often defined as having the result 0.



Venn diagram of $A \vee B$



Venn diagram of $A \vee B \vee C$

Truth table

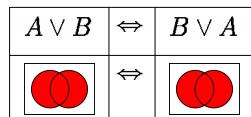
The truth table of $A \vee B$:

Disjunctions of the arguments on the left
The false bits form a Sierpinski triangle

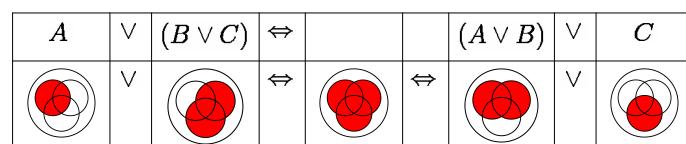
INPUT	OUTPUT	
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

Properties

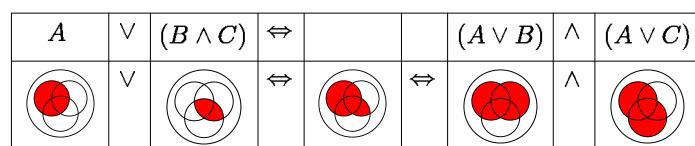
- **Commutativity**



- **Associativity**

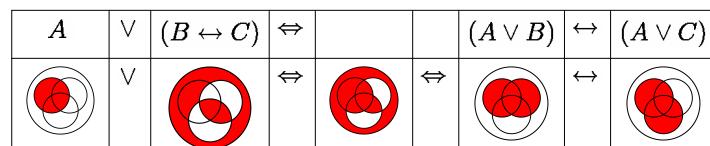


- **Distributivity** with various operations, especially with *and*



others

with biconditional:



with material implication:

A	\vee	$(B \rightarrow C)$	\Leftrightarrow			$(A \vee B)$	\rightarrow	$(A \vee C)$
	\vee		\Leftrightarrow		\Leftrightarrow		\rightarrow	

with itself:

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

- **Idempotency**

A	\vee	A	\Leftrightarrow	A
	\vee		\Leftrightarrow	

- **Monotonicity**

$A \rightarrow B$	\Rightarrow			$(A \vee C) \rightarrow (B \vee C)$
	\Rightarrow		\Leftrightarrow	

- **Truth-preserving validity**

When all inputs are true, the output is true.

$A \wedge B$	\Rightarrow	$A \vee B$
	\Rightarrow	
(to be tested)		

- **False-preserving validity**

When all inputs are false, the output is false.

$A \vee B$	\Rightarrow	$A \vee B$
	\Rightarrow	
(to be tested)		

- **Walsh spectrum: (3,-1,-1,-1)**

- **Nonlinearity: 1** (the function is bent)

If using binary values for true (1) and false (0), then *logical disjunction* works almost like binary addition. The only difference is that $1 \vee 1 = 1$, while $1 + 1 = 10$.

Symbol

The mathematical symbol for logical disjunction varies in the literature. In addition to the word "or", and the formula " Apq ", the symbol " \vee ", deriving from the Latin word *vel* ("either", "or") is commonly used for disjunction. For example: " $A \vee B$ " is read as "A or B". Such a disjunction is false if both A and B are false. In all other cases it is true.

All of the following are disjunctions:

$$\begin{aligned} & A \vee B \\ & \neg A \vee B \\ & A \vee \neg B \vee \neg C \vee D \vee \neg E. \end{aligned}$$

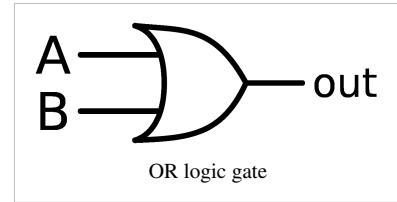
The corresponding operation in set theory is the set-theoretic union.

Applications in computer science

Operators corresponding to logical disjunction exist in most programming languages.

Bitwise operation

Disjunction is often used for bitwise operations. Examples:



- 0 or 0 = 0
- 0 or 1 = 1
- 1 or 0 = 1
- 1 or 1 = 1
- 1010 or 1100 = 1110

The `or` operator can be used to set bits in a bit field to 1, by `or-ing` the field with a constant field with the relevant bits set to 1. For example, `x = x | 0b00000001` will force the final bit to 1 while leaving other bits unchanged.

Logical operation

Many languages distinguish between bitwise and logical disjunction by providing two distinct operators; in languages following C, bitwise disjunction is performed with the single pipe (`|`) and logical disjunction with the double pipe (`||`) operators.

Logical disjunction is usually short-circuited; that is, if the first (left) operand evaluates to `true` then the second (right) operand is not evaluated. The logical disjunction operator thus usually constitutes a sequence point.

In a parallel (concurrent) language, it is possible to short-circuit both sides: they are evaluated in parallel, and if one terminates with value `true`, the other is interrupted. This operator is thus called the **parallel or**.

Although in most languages the type of a logical disjunction expression is boolean and thus can only have the value `true` or `false`, in some (such as Python and JavaScript) the logical disjunction operator returns one of its operands: the first operand if it evaluates to a `true` value, and the second operand otherwise.

Constructive disjunction

The Curry–Howard correspondence relates a constructivist form of disjunction to tagged union types.

Union

The membership of an element of an union set in set theory is defined in terms of a logical disjunction: $x \in A \cup B$ if and only if $(x \in A) \vee (x \in B)$. Because of this, logical disjunction satisfies many of the same identities as set-theoretic union, such as associativity, commutativity, distributivity, and de Morgan's laws.

Notes

- George Boole, closely following analogy with ordinary mathematics, premised, as a necessary condition to the definition of " $x + y$ ", that x and y were mutually exclusive. Jevons, and practically all mathematical logicians after him, advocated, on various grounds, the definition of "logical addition" in a form which does not necessitate mutual exclusiveness.

External links

- Hazewinkel, Michiel, ed. (2001), "Disjunction" ^[1], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Stanford Encyclopedia of Philosophy entry ^[2]
- Eric W. Weisstein. "Disjunction." From MathWorld--A Wolfram Web Resource ^[3]

References

[1] <http://www.encyclopediaofmath.org/index.php?title=p/d033260>

[2] <http://plato.stanford.edu/entries/disjunction/>

[3] <http://mathworld.wolfram.com/Disjunction.html>

Logical equality

Logical equality is a logical operator that corresponds to equality in Boolean algebra and to the logical biconditional in propositional calculus. It gives the functional value *true* if both functional arguments have the same logical value, and *false* if they are different.

It is customary practice in various applications, if not always technically precise, to indicate the operation of **logical equality** on the logical operands x and y by any of the following forms:

$$\begin{array}{lll} x \leftrightarrow y & x \Leftrightarrow y & Exy \\ x \text{ EQ } y & x = y & \end{array}$$

Some logicians, however, draw a firm distinction between a *functional form*, like those in the lefthand column, which they interpret as an application of a function to a pair of arguments — and thus a mere indication that the value of the compound expression depends on the values of the component expressions — and an *equational form*, like those in the righthand column, which they interpret as an assertion that the arguments have equal values, in other words, that the functional value of the compound expression is *true*.

In mathematics, the plus sign "+" almost invariably indicates an operation that satisfies the axioms assigned to addition in the type of algebraic structure that is known as a *field*. For boolean algebra, this means that the logical operation signified by "+" is not the same as the inclusive disjunction signified by " \vee " but is actually equivalent to the logical inequality operator signified by " \neq ", or what amounts to the same thing, the exclusive disjunction signified by "XOR". Naturally, these variations in usage have caused some failures to communicate between mathematicians and switching engineers over the years. At any rate, one has the following array of corresponding forms for the symbols associated with logical inequality:

$$\begin{array}{lll} x + y & x \neq y & Jxy \\ x \text{ XOR } y & x \neq y & \end{array}$$

This explains why "EQ" is often called "XNOR" in the combinational logic of circuit engineers, since it is the *Negation* of the *XOR* operation; **NXOR** is a less commonly used alternative.^[citation needed] Another rationalization of the admittedly circuitous name "XNOR" is that one begins with the "both false" operator NOR and then adds the eXception, "or both true".

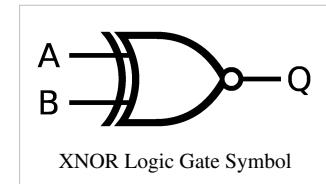
Definition

Logical equality is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if and only if both operands are false or both operands are true.

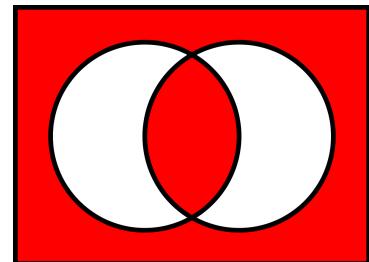
The truth table of $p \text{ EQ } q$ (also written as $p = q$, $p \leftrightarrow q$, or $p \sqsubseteq q$, or $p == q$) is as follows:

Logical Equality

p	q	$p = q$
0	0	1
0	1	0
1	0	0
1	1	1



XNOR Logic Gate Symbol



The Venn diagram of A EQ B (red part is true)

Alternative descriptions

The form $(x = y)$ is equivalent to the form $(x \wedge y) \vee (\neg x \wedge \neg y)$.

$$(x = y) = \neg(x \oplus y) = \neg x \oplus y = x \oplus \neg y = (x \wedge y) \vee (\neg x \wedge \neg y) = (\neg x \vee y) \wedge (x \vee \neg y)$$

For the operands x and y , the truth table of the logical equality operator is as follows:

$x \leftrightarrow y$		y	
		T	F
x		T	F
		F	F

External links

- Mathworld, XNOR [1]

References

[1] <http://mathworld.wolfram.com/XNOR.html>

Logical implication

Logical consequence (also **entailment**) is one of the most fundamental concepts in logic. It is the relationship between statements that holds true when one logically "follows from" one or more others. Valid logical arguments are ones in which the conclusions follow from its premises, and its conclusions are consequences of its premises. The philosophical analysis of logical consequence involves asking, 'in what sense does a conclusion follow from its premises?' and 'what does it mean for a conclusion to be a consequence of premises?'^[1] All of philosophical logic can be thought of as providing accounts of the nature of logical consequence, as well as logical truth.^[2]

Logical consequence is taken to be both necessary and formal with examples explicated using models and proofs. A sentence is said to be a logical consequence of a set of sentences, for a given language, if and only if, using logic alone (i.e. without regard to any interpretations of the sentences) the sentence must be true if every sentence in the set were to be true.^[3]

Logicians make precise accounts of logical consequence with respect to a given language \mathcal{L} by constructing a deductive system for \mathcal{L} , or by formalizing the intended semantics for \mathcal{L} . Alfred Tarski highlighted three salient features for which any adequate characterization of logical consequence needs to account: 1) that the logical consequence relation relies on the logical form of the sentences involved, 2) that the relation is a priori, i.e. it can be determined whether or not it holds without regard to sense experience, and 3) that the relation has a modal component.

Formal accounts of logical consequence

The most widely prevailing view on how to best account for logical consequence is to appeal to formality. This is to say that whether statements follow from one another logically depends on the structure or logical form of the statements without regard to the contents of that form.

Syntactic accounts of logical consequence rely on schemes using inference rules. For instance, we can express the logical form of a valid argument as "All A are B . All C are A . Therefore, All C are B ." This argument is formally valid, because every instance of arguments constructed using this scheme are valid.

This is in contrast to an argument like "Fred is Mike's brother's son. Therefore Fred is Mike's nephew." Since this argument depends on the meanings of the words "brother", "son", and "nephew", the statement "Fred is Mike's nephew" is a so-called material consequence of "Fred is Mike's brother's son," not a formal consequence. A formal consequence must be true *in all cases*, however this is an incomplete definition of formal consequence, since even the argument " P is Q 's brother's son, therefore P is Q 's nephew" is valid in all cases, but is not a *formal* argument.

A priori property of logical consequence

If you know that Q follows logically from P no information about the possible interpretations of P or Q will affect that knowledge. Our knowledge that Q is a logical consequence of P cannot be influenced by empirical knowledge. Deductively valid arguments can be known to be so without recourse to experience, so they must be knowable a priori. However, formality alone does not guarantee that logical consequence is not influenced by empirical knowledge. So the a priori property of logical consequence is considered to be independent of formality.

Proofs and models

The two prevailing techniques for providing accounts of logical consequence involve expressing the concept in terms of *proofs* and via *models*. The study of the syntactic consequence (of a logic) is called (its) proof theory whereas the study of (its) semantic consequence is called (its) model theory.

Syntactic consequence

A formula A is a **syntactic consequence**^{[4][5][6][7]} within some formal system \mathcal{FS} of a set Γ of formulas if there is a formal proof in \mathcal{FS} of A from the set Γ .

$$\Gamma \vdash_{\mathcal{FS}} A$$

Syntactic consequence does not depend on any interpretation of the formal system.^[8]

Semantic consequence

A formula A is a **semantic consequence** within some formal system \mathcal{FS} of a set of statements Γ

$$\Gamma \models_{\mathcal{FS}} A,$$

if and only if there is no model \mathcal{I} in which all members of Γ are true and A is false.^[9] Or, in other words, the set of the interpretations that make all members of Γ true is a subset of the set of the interpretations that make A true.

Modal accounts

Modal accounts of logical consequence are variations on the following basic idea:

$\Gamma \vdash A$ is true if and only if it is *necessary* that if all of the elements of Γ are true, then A is true.

Alternatively (and, most would say, equivalently):

$\Gamma \vdash A$ is true if and only if it is *impossible* for all of the elements of Γ to be true and A false.

Such accounts are called "modal" because they appeal to the modal notions of logical necessity and logical possibility. 'It is necessary that' is often expressed as a universal quantifier over possible worlds, so that the accounts above translate as:

$\Gamma \vdash A$ is true if and only if there is no possible world at which all of the elements of Γ are true and A is false (untrue).

Consider the modal account in terms of the argument given as an example above:

All frogs are green.

Kermit is a frog.

Therefore, Kermit is green.

The conclusion is a logical consequence of the premises because we can't imagine a possible world where (a) all frogs are green; (b) Kermit is a frog; and (c) Kermit is not green.

Modal-formal accounts

Modal-formal accounts of logical consequence combine the modal and formal accounts above, yielding variations on the following basic idea:

$\Gamma \vdash A$ if and only if it is impossible for an argument with the same logical form as Γ / A to have true premises and a false conclusion.

Warrant-based accounts

The accounts considered above are all "truth-preservational," in that they all assume that the characteristic feature of a good inference is that it never allows one to move from true premises to an untrue conclusion. As an alternative, some have proposed "warrant-preservational" accounts, according to which the characteristic feature of a good inference is that it never allows one to move from justifiably assertible premises to a conclusion that is not justifiably assertible. This is (roughly) the account favored by intuitionists such as Michael Dummett.

Non-monotonic logical consequence

The accounts discussed above all yield monotonic consequence relations, i.e. ones such that if A is a consequence of Γ , then A is a consequence of any superset of Γ . It is also possible to specify non-monotonic consequence relations to capture the idea that, e.g., 'Tweety can fly' is a logical consequence of

{Birds can typically fly, Tweety is a bird}

but not of

{Birds can typically fly, Tweety is a bird, Tweety is a penguin}.

For more on this, see Belief revision#Non-monotonic inference relation.

References

- [1] Beall, JC and Restall, Greg, *Logical Consequence* (<http://plato.stanford.edu/archives/fall2009/entries/logical-consequence/>) The Stanford Encyclopedia of Philosophy (Fall 2009 Edition), Edward N. Zalta (ed.).
- [2] Quine, Willard Van Orman, *Philosophy of logic*
- [3] McKeon, Matthew, *Logical Consequence* (<http://www.iep.utm.edu/logcon/>) Internet Encyclopedia of Philosophy.
- [4] Dummett, Michael (1993) philosophy of language (http://books.google.com/books?id=EYP7uCZIRQYC&pg=PA82&lpg=PA82&dq=syntactic+consequence&source=bl&ots=Ms58438B6w&sig=FE38FCaZpRpAr18gtG7INX4wieM&hl=en&ei=qOy7SoLIEI7KsQPgnYG7BA&sa=X&oi=book_result&ct=result&resnum=6#v=onepage&q=syntactic%20consequence&f=false) Harvard University Press, p.82ff
- [5] Lear, Jonathan (1986) and Logical Theory (http://books.google.com/books?id=lXI7AAAAIAAJ&pg=PA1&lpg=PA1&dq=syntactic+consequence&source=bl&ots=8IYWyFYTN-&sig=wrOg75cFxQwn1Uq-8LShBNXf9w0&hl=en&ei=I-y7SpHtLZLotgOsnLHcBQ&sa=X&oi=book_result&ct=result&resnum=10#v=onepage&q=syntactic%20consequence&f=false) Cambridge University Press, 136p.
- [6] Creath, Richard, and Friedman, Michael (2007) Cambridge companion to Carnap (http://books.google.com/books?id=87BcFLgJmxMC&pg=PA189&lpg=PA189&dq=syntactic+consequence&source=bl&ots=Fn2zomcMZP&sig=8hnJWsJFysNhmWLskICo4IQDYAc&hl=en&ei=I-y7SpHtLZLotgOsnLHcBQ&sa=X&oi=book_result&ct=result&resnum=6#v=onepage&q=syntactic%20consequence&f=false) Cambridge University Press, 371p.
- [7] FOLDOC: "syntactic consequence" (<http://www.swif.uniba.it/lei/foldop/foldoc.cgi?syntactic+consequence>)
- [8] Hunter, Geoffrey, *Metalogic: An Introduction to the Metatheory of Standard First-Order Logic*, University of California Pres, 1971, p. 75.
- [9] Etchemendy, John, *Logical consequence*, The Cambridge Dictionary of Philosophy

Logical negation

In logic, **negation**, also called **logical complement**, is an operation that essentially takes a proposition p to another proposition "not p ", written $\neg p$, which is interpreted intuitively as being true when p is false and false when p is true. Negation is thus a unary (single-argument) logical connective. It may be applied as an operation on propositions, truth values, or semantic values more generally. In classical logic, negation is normally identified with the truth function that takes *truth* to *falsity* and vice versa. In intuitionistic logic, according to the Brouwer–Heyting–Kolmogorov interpretation, the negation of a proposition p is the proposition whose proofs are the refutations of p .

Definition

Classical negation is an operation on one logical value, typically the value of a proposition, that produces a value of *true* when its operand is false and a value of *false* when its operand is true. So, if statement A is true, then $\neg A$ (pronounced "not A ") would therefore be false; and conversely, if $\neg A$ is true, then A would be false.

The truth table of $\neg p$ is as follows:

Truth table of $\neg p$

p	$\neg p$
True	False
False	True

Classical negation can be defined in terms of other logical operations. For example, $\neg p$ can be defined as $p \rightarrow F$, where " \rightarrow " is logical consequence and F is absolute falsehood. Conversely, one can define F as $p \& \neg p$ for any proposition p , where " $\&$ " is logical conjunction. The idea here is that any contradiction is false. While these ideas work in both classical and intuitionistic logic, they do not work in Brazilian logic, where contradictions are not necessarily false. But in classical logic, we get a further identity: $p \rightarrow q$ can be defined as $\neg p \vee q$, where " \vee " is logical disjunction: "not p , or q ".

Algebraically, classical negation corresponds to complementation in a Boolean algebra, and intuitionistic negation to pseudocomplementation in a Heyting algebra. These algebras provide a semantics for classical and intuitionistic logic respectively.

Notation

The negation of a proposition p is notated in different ways in various contexts of discussion and fields of application. Among these variants are the following:

Notation	Vocalization
$\neg p$	not p
\bar{p}	not p
$\sim p$	not p
Np	en p
p'	p prime, p complement
\bar{p}	p bar, bar p
$!p$	bang p not p

In Set Theory \ is also used to indicate 'not member of': $U \setminus A$ is the set of all members of U that are not members of A .

No matter how it is notated or symbolized, the negation $\neg p$ / \bar{p} can be read as "it is not the case that p ", "not that p ", or usually more simply (though not grammatically) as "not p ".

Properties

Double negation

Within a system of classical logic, double negation, that is, the negation of the negation of a proposition p , is logically equivalent to p . Expressed in symbolic terms, $\neg\neg p \Leftrightarrow p$. In intuitionistic logic, a proposition implies its double negation but not conversely. This marks one important difference between classical and intuitionistic negation. Algebraically, classical negation is called an involution of period two.

However, in intuitionistic logic we do have the equivalence of $\neg\neg\neg p$ and $\neg p$. Moreover, in the propositional case, a sentence is classically provable if its double negation is intuitionistically provable. This result is known as Glivenko's theorem.

Distributivity

De Morgan's laws provide a way of distributing negation over disjunction and conjunction :

$$\begin{aligned}\neg(a \vee b) &\equiv (\neg a \wedge \neg b), \text{ and} \\ \neg(a \wedge b) &\equiv (\neg a \vee \neg b).\end{aligned}$$

Linearity

In Boolean algebra, a linear function is one such that:

If there exists $a_0, a_1, \dots, a_n \in \{0,1\}$ such that $f(b_1, \dots, b_n) = a_0 \oplus (a_1 \wedge b_1) \oplus \dots \oplus (a_n \wedge b_n)$, for all $b_1, \dots, b_n \in \{0,1\}$.

Another way to express this is that each variable always makes a difference in the truth-value of the operation or it never makes a difference. Negation is a linear logical operator.

Self dual

In Boolean algebra a self dual function is one such that:

$f(a_1, \dots, a_n) = \neg f(\neg a_1, \dots, \neg a_n)$ for all $a_1, \dots, a_n \in \{0,1\}$. Negation is a self dual logical operator.

Rules of inference

There are a number of equivalent ways to formulate rules for negation. One usual way to formulate classical negation in a natural deduction setting is to take as primitive rules of inference *negation introduction* (from a derivation of p to both q and $\neg q$, infer $\neg p$; this rule also being called *reductio ad absurdum*), *negation elimination* (from p and $\neg p$ infer q ; this rule also being called *ex falso quodlibet*), and *double negation elimination* (from $\neg\neg p$ infer p). One obtains the rules for intuitionistic negation the same way but by excluding double negation elimination.

Negation introduction states that if an absurdity can be drawn as conclusion from p then p must not be the case (i.e. p is false (classically) or refutable (intuitionistically) or etc.). Negation elimination states that anything follows from an absurdity. Sometimes negation elimination is formulated using a primitive absurdity sign \perp . In this case the rule says that from p and $\neg p$ follows an absurdity. Together with double negation elimination one may infer our originally formulated rule, namely that anything follows from an absurdity.

Typically the intuitionistic negation $\neg p$ of p is defined as $p \rightarrow \perp$. Then negation introduction and elimination are just special cases of implication introduction (conditional proof) and elimination (modus ponens). In this case one must also add as a primitive rule *ex falso quodlibet*.

Programming

As in mathematics, negation is used in computer science to construct logical statements.

```
if (! (r == t))
{
    /*...statements executed when r does NOT equal t...*/
}
```

The "!" signifies logical NOT in B, C, and languages with a C-inspired syntax such as C++, Java, JavaScript, Perl, and PHP. "NOT" is the operator used in ALGOL 60, BASIC, and languages with an ALGOL- or BASIC-inspired syntax such as Pascal, Ada, Eiffel and Seed7. Some languages (C++, Perl, etc.) provide more than one operator for negation. A few languages like PL/I and Ratfor use \neg for negation. Some modern computers and operating systems will display \neg as ! on files encoded in ASCII. Most modern languages allow the above statement to be shortened from `if (! (r == t))` to `if (r != t)`, which allows sometimes, when the compiler/interpreter is not able to optimize it, faster programs.

In computer science there is also *bitwise negation*. This takes the value given and switches all the binary 1s to 0s and 0s to 1s. See bitwise operation. This is often used to create ones' complement or " \sim " in C or C++ and two's complement (just simplified to " $-$ " or the negative sign since this is equivalent to taking the arithmetic negative value of the number) as it basically creates the opposite (negative value equivalent) or mathematical complement of the value (where both values are added together they create a whole).

To get the absolute (positive equivalent) value of a given integer the following would work as the " $-$ " changes it from negative to positive (it is negative because " $x < 0$ " yields true)

```
unsigned int abs(int x)
{
    if (x < 0)
        return -x;
```

```
    else
        return x;
}
```

To demonstrate logical negation:

```
unsigned int abs(int x)
{
    if (! (x < 0))
        return x;
    else
        return -x;
}
```

Inverting the condition and reversing the outcomes produces code that is logically equivalent to the original code, i.e. will have identical results for any input (note that depending on the compiler used, the actual instructions performed by the computer may differ).

This convention occasionally surfaces in written speech, as computer-related slang for *not*. The phrase `!voting`, for example, means "not voting".

Kripke semantics

In Kripke semantics where the semantic values of formulae are sets of possible worlds, negation can be taken to mean set-theoretic complementation.^[citation needed] (See also possible world semantics.)

References

Further reading

- Gabbay, Dov, and Wansing, Heinrich, eds., 1999. *What is Negation?*, Kluwer.
- Horn, L., 2001. *A Natural History of Negation*, University of Chicago Press.
- G. H. von Wright, 1953–59, "On the Logic of Negation", *Commentationes Physico-Mathematicae* 22.
- Wansing, Heinrich, 2001, "Negation", in Goble, Lou, ed., *The Blackwell Guide to Philosophical Logic*, Blackwell.
- Marco Tettamanti, Rosa Manenti, Pasquale A. Della Rosa, Andrea Falini, Daniela Perani, Stefano F. Cappa and Andrea Moro (2008). "Negation in the brain: Modulating action representation", NeuroImage Volume 43, Issue 2, 1 November 2008, pages 358–367, <http://dx.doi.org/10.1016/j.neuroimage.2008.08.004/>

External links

- Hazewinkel, Michiel, ed. (2001), "Negation" (<http://www.encyclopediaofmath.org/index.php?title=p/n066170>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- NOT (<http://mathworld.wolfram.com/NOT.html>), on MathWorld

Lupanov representation

Lupanov's (k, s) -representation, named after Oleg Lupanov, is a way of representing Boolean circuits so as to show that the reciprocal of the Shannon effect. Shannon had showed that almost all Boolean functions of n variables need a circuit of size at least $2^n n^{-1}$. The reciprocal is that:

All Boolean functions of n variables can be computed with a circuit of at most $2^n n^{-1} + o(2^n n^{-1})$ gates.

Definition

The idea is to represent the values of a boolean function f in a table of 2^k rows, representing the possible values of the k first variables x_1, \dots, x_k , and 2^{n-k} columns representing the values of the other variables.

Let A_1, \dots, A_p be a partition of the rows of this table such that for $i < p$, $|A_i| = s$ and $|A_p| = s' \leq s$. Let $f_i(x) = f(x)$ iff $x \in A_i$.

Moreover, let $B_{i,w}$ be the set of the columns whose intersection with A_i is w .

Majority function

In Boolean logic, the **majority function** (also called the **median operator**) is a function from n inputs to one output. The value of the operation is false when $n/2$ or more arguments are false, and true otherwise. Alternatively, representing true values as 1 and false values as 0, we may use the formula

$$\text{Majority}(p_1, \dots, p_n) = \left\lfloor \frac{1}{2} + \frac{(\sum_{i=1}^n p_i) - 1/2}{n} \right\rfloor.$$

The " $-1/2$ " in the formula serves to break ties in favor of zeros when n is even. If the term " $-1/2$ " is omitted, the formula can be used for a function that breaks ties in favor of ones.

Boolean circuits

A *majority gate* is a logical gate used in circuit complexity and other applications of Boolean circuits. A majority gate returns true if and only if more than 50% of its inputs are true.

For instance, in a full adder, the carry output is found by applying a majority function to the three inputs, although frequently this part of the adder is broken down into several simpler logical gates.

A major result in circuit complexity asserts that the majority function cannot be computed by AC0 circuits of subexponential size.

Monotone formulae for majority

For $n = 1$ the median operator is just the unary identity operation x . For $n = 3$ the ternary median operator can be expressed using conjunction and disjunction as $xy + yz + zx$. Remarkably this expression denotes the same operation independently of whether the symbol $+$ is interpreted as inclusive or or exclusive or.

For an arbitrary n there exists a monotone formula for majority of size $O(n^{5.3})$ (Valiant 1984). This is proved using probabilistic method. Thus, this formula is non-constructive. However, one can obtain an explicit formula for majority of polynomial size using a sorting network of Ajtai, Komlós, and Szemerédi.

Properties

Among the properties of the ternary median operator $\langle x,y,z \rangle$ are:

1. $\langle x,y,y \rangle = y$
2. $\langle x,y,z \rangle = \langle z,x,y \rangle$
3. $\langle x,y,z \rangle = \langle x,z,y \rangle$
4. $\langle \langle x,w,y \rangle, w,z \rangle = \langle x,w, \langle y,w,z \rangle \rangle$

An abstract system satisfying these as axioms is a median algebra.

References

- Valiant, L. (1984). "Short monotone formulae for the majority function". *Journal of Algorithms* 5 (3): 363–366. doi:10.1016/0196-6774(84)90016-6 [1].
- Knuth, Donald E. (2008). *Introduction to combinatorial algorithms and Boolean functions*. The Art of Computer Programming 4a. Upper Saddle River, NJ: Addison-Wesley. pp. 64–74. ISBN 0-321-53496-4.

References

[1] [http://dx.doi.org/10.1016/0196-6774\(84\)90016-6](http://dx.doi.org/10.1016/0196-6774(84)90016-6)

Material conditional

The **material conditional** (also known as "**material implication**", "**material consequence**", or simply "**implication**", "**implies**" or "**conditional**") is a logical connective (or a binary operator) that is often symbolized by a forward arrow " \rightarrow ". The material conditional is used to form statements of the form " $p \rightarrow q$ " (termed a conditional statement) which is read as "if p then q " and conventionally compared to the English construction "If...then...". But unlike as the English construction may, the conditional statement " $p \rightarrow q$ " does not specify a causal relationship between p and q and is to be understood to mean "if p is true, then q is also true" such that the statement " $p \rightarrow q$ " is false only when p is true and q is false. The material conditional is also to be distinguished from logical consequence.

The material conditional is also symbolized using:

1. $p \supset q$ (Although this symbol is confused with the superset symbol used by algebra of sets.);
2. $p \Rightarrow q$ (Although this symbol is often used for logical consequence (i.e. logical implication) rather than for material conditional.)

With respect to the material conditionals above, p is termed the *antecedent*, and q the *consequent* of the conditional. Conditional statements may be nested such that either or both of the antecedent or the consequent may themselves be conditional statements. In the example " $(p \rightarrow q) \rightarrow (r \rightarrow s)$ " both the antecedent and the consequent are conditional statements.

In classical logic $p \rightarrow q$ is logically equivalent to $\neg(p \wedge \neg q)$ and by De Morgan's Law to $\neg p \vee q$. Whereas, in minimal logic (and therefore also intuitionistic logic) $p \rightarrow q$ only logically entails $\neg(p \wedge \neg q)$; and in intuitionistic logic (but not minimal logic) $\neg p \vee q$ entails $p \rightarrow q$.

Definitions of the material conditional

Logicians have many different views on the nature of material implication and approaches to explain its sense.

As a truth function

In classical logic, the compound $p \rightarrow q$ is logically equivalent to the negative compound: not both p and not q . Thus the compound $p \rightarrow q$ is *false* if and only if both p is true and q is false. By the same stroke, $p \rightarrow q$ is *true* if and only if either p is false or q is true (or both). Thus \rightarrow is a function from pairs of truth values of the components p, q to truth values of the compound $p \rightarrow q$, whose truth value is entirely a function of the truth values of the components. Hence, the compound $p \rightarrow q$ is called *truth-functional*. The compound $p \rightarrow q$ is logically equivalent also to $\neg p \vee q$ (either not p , or q (or both)), and to $\neg q \rightarrow \neg p$ (if not q then not p). But it is not equivalent to $\neg p \rightarrow \neg q$, which is equivalent to $q \rightarrow p$.

Truth table

The truth table associated with the material conditional **not p or q** (symbolized as $p \rightarrow q$) and the logical implication **p implies q** (symbolized as $p \rightarrow q$, or Cpq) is as follows:

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

As a formal connective

The material conditional can be considered as a symbol of a formal theory, taken as a set of sentences, satisfying all the classical inferences involving \rightarrow , in particular the following characteristic rules:

1. Modus ponens;
2. Conditional proof;
3. Classical contraposition;
4. Classical reductio.

Unlike the truth-functional one, this approach to logical connectives permits the examination of structurally identical propositional forms in various logical systems, where somewhat different properties may be demonstrated. For example, in intuitionistic logic which rejects proofs by contraposition as valid rules of inference, $(p \rightarrow q) \Rightarrow \neg p \vee q$ is not a propositional theorem, but the material conditional is used to define negation.

Formal properties

When studying logic formally, the material conditional is distinguished from the semantic consequence relation \models .

We say $A \models B$ if every interpretation that makes A true also makes B true. However, there is a close relationship between the two in most logics, including classical logic. For example, the following principles hold:

- If $\Gamma \models \psi$ then $\emptyset \models (\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \psi)$ for some $\varphi_1, \dots, \varphi_n \in \Gamma$. (This is a particular form of the deduction theorem. In words, it says that if Γ models ψ this means that ψ can be deduced just from some subset of the theorems in Γ .)
- The converse of the above
- Both \rightarrow and \models are monotonic; i.e., if $\Gamma \models \psi$ then $\Delta \cup \Gamma \models \psi$, and if $\varphi \rightarrow \psi$ then $(\varphi \wedge \alpha) \rightarrow \psi$ for any α, Δ . (In terms of structural rules, this is often referred to as weakening or *thinning*.)

These principles do not hold in all logics, however. Obviously they do not hold in non-monotonic logics, nor do they hold in relevance logics.

Other properties of implication (the following expressions are always true, for any logical values of variables):

- distributivity: $(s \rightarrow (p \rightarrow q)) \rightarrow ((s \rightarrow p) \rightarrow (s \rightarrow q))$
- transitivity: $(a \rightarrow b) \rightarrow ((b \rightarrow c) \rightarrow (a \rightarrow c))$
- reflexivity: $a \rightarrow a$
- totality: $(a \rightarrow b) \vee (b \rightarrow a)$
- truth preserving: The interpretation under which all variables are assigned a truth value of 'true' produces a truth value of 'true' as a result of material implication.
- commutativity of antecedents: $(a \rightarrow (b \rightarrow c)) \equiv (b \rightarrow (a \rightarrow c))$

Note that $a \rightarrow (b \rightarrow c)$ is logically equivalent to $(a \wedge b) \rightarrow c$; this property is sometimes called un/currying.

Because of these properties, it is convenient to adopt a right-associative notation for \rightarrow where $a \rightarrow b \rightarrow c$ denotes $a \rightarrow (b \rightarrow c)$.

Note also that comparison of the truth table shows that $a \rightarrow b$ is equivalent to $\neg a \vee b$, and it is sometimes convenient to replace one by the other in proofs. Such a replacement can be viewed as a rule of inference.

Philosophical problems with material conditional

Outside of mathematics, it is a matter of some controversy as to whether the truth function for material implication provides an adequate treatment of conditional statements in English (a sentence in the indicative mood with a conditional clause attached, i.e., an indicative conditional, or false-to-fact sentences in the subjunctive mood, i.e., a counterfactual conditional). That is to say, critics argue that in some non-mathematical cases, the truth value of a compound statement, "if p then q ", is not adequately determined by the truth values of p and q . Examples of non-truth-functional statements include: " q because p ", " p before q " and "it is possible that p ". "[Of] the sixteen possible truth-functions of A and B , material implication is the only serious candidate. First, it is uncontroversial that when A is true and B is false, "If A , B " is false. A basic rule of inference is modus ponens: from "If A , B " and A , we can infer B . If it were possible to have A true, B false and "If A , B " true, this inference would be invalid. Second, it is uncontroversial that "If A , B " is sometimes true when A and B are respectively (true, true), or (false, true), or (false, false)... Non-truth-functional accounts agree that "If A , B " is false when A is true and B is false; and they agree that the conditional is sometimes true for the other three combinations of truth-values for the components; but they deny that the conditional is always true in each of these three cases. Some agree with the truth-functionalist that when A and B are both true, "If A , B " must be true. Some do not, demanding a further relation between the facts that A and that B ."

The truth-functional theory of the conditional was integral to Frege's new logic (1879). It was taken up enthusiastically by Russell (who called it "material implication"), Wittgenstein in the *Tractatus*, and the logical positivists, and it is now found in every logic text. It is the first theory of conditionals which students encounter. Typically, it does not strike students as *obviously* correct. It is logic's first surprise. Yet, as the textbooks testify, it does a creditable job in many circumstances. And it has many defenders. It is a strikingly simple theory: "If A , B " is false when A is true and B is false. In all other cases, "If A , B " is true. It is thus equivalent to " $\sim(A \& \sim B)$ " and to " $\sim A$ or B ". " $A \supset B$ " has, by stipulation, these truth conditions.

— Dorothy Edgington, *The Stanford Encyclopedia of Philosophy*, "Conditionals"

The meaning of the material conditional can sometimes be used in the natural language English "if *condition* then *consequence*" construction (a kind of conditional sentence), where *condition* and *consequence* are to be filled with English sentences. However, this construction also implies a "reasonable" connection between the condition (protasis) and consequence (apodosis) (see Connexive logic).^[citation needed]

The material conditional can yield some unexpected truths when expressed in natural language. For example, any material conditional statement with a false antecedent is true (see vacuous truth). So the statement "if 2 is odd then 2 is even" is true. Similarly, any material conditional with a true consequent is true. So the statement "if I have a penny in my pocket then Paris is in France" is always true, regardless of whether or not there is a penny in my pocket. These problems are known as the paradoxes of material implication, though they are not really paradoxes in the strict sense; that is, they do not elicit logical contradictions. These unexpected truths arise because speakers of English (and other natural languages) are tempted to equivocate between the material conditional and the indicative conditional, or other conditional statements, like the counterfactual conditional and the material biconditional. It is not surprising that a rigorously defined truth-functional operator does not correspond exactly to all notions of implication or otherwise expressed by 'if...then...' sentences in English (or their equivalents in other natural languages). For an overview of some the various analyses, formal and informal, of conditionals, see the "References" section below.

References

Further reading

- Brown, Frank Markham (2003), *Boolean Reasoning: The Logic of Boolean Equations*, 1st edition, Kluwer Academic Publishers, Norwell, MA. 2nd edition, Dover Publications, Mineola, NY, 2003.
- Edgington, Dorothy (2001), "Conditionals", in Lou Goble (ed.), *The Blackwell Guide to Philosophical Logic*, Blackwell.
- Quine, W.V. (1982), *Methods of Logic*, (1st ed. 1950), (2nd ed. 1959), (3rd ed. 1972), 4th edition, Harvard University Press, Cambridge, MA.
- Stalnaker, Robert, "Indicative Conditionals", *Philosophia*, 5 (1975): 269–286.

External links

- Conditionals (<http://plato.stanford.edu/entries/conditionals>) entry by Edgington, Dorothy in the *Stanford Encyclopedia of Philosophy*

Material equivalence

$\leftrightarrow \Leftrightarrow \equiv$

Logical symbols
representing *iff*

In logic and related fields such as mathematics and philosophy, **if and only if** (shortened **iff**) is a biconditional logical connective between statements.

In that it is biconditional, the connective can be likened to the standard material conditional ("only if", equal to "if ... then") combined with its reverse ("if"); hence the name. The result is that the truth of either one of the connected statements requires the truth of the other, i.e., either both statements are true, or both are false. It is controversial whether the connective thus defined is properly rendered by the English "if and only if", with its pre-existing meaning. There is nothing to stop one from *stipulating* that we may read this connective as "only if and if", although this may lead to confusion.

In writing, phrases commonly used, with debatable propriety, as alternatives to P "if and only if" Q include *Q is necessary and sufficient for P*, *P is equivalent (or materially equivalent) to Q* (compare material implication), *P precisely if Q*, *P precisely (or exactly) when Q*, *P exactly in case Q*, and *P just in case Q*. Many authors regard "iff"^[citation needed] as unsuitable in formal writing; others use it freely.

In *logic formulae*, logical symbols are used instead of these phrases; see the discussion of notation.

Definition

The truth table of $p \leftrightarrow q$ is as follows:^[1]

Iff

p	q	$p \leftrightarrow q$
T	T	T
T	F	F
F	T	F
F	F	T

Note that it is equivalent to that produced by the XNOR gate, and opposite to that produced by the XOR gate.

Usage

Notation

The corresponding logical symbols are " \leftrightarrow ", " \Leftrightarrow " and " \equiv ", and sometimes "iff". These are usually treated as equivalent. However, some texts of mathematical logic (particularly those on first-order logic, rather than propositional logic) make a distinction between these, in which the first, \leftrightarrow , is used as a symbol in logic formulas, while \Leftrightarrow is used in reasoning about those logic formulas (e.g., in metalogic). In Łukasiewicz's notation, it is the prefix symbol 'E'.

Another term for this logical connective is exclusive nor.

Proofs

In most logical systems, one proves a statement of the form " P iff Q " by proving "if P , then Q " and "if Q , then P ". Proving this pair of statements sometimes leads to a more natural proof, since there are not obvious conditions in which one would infer a biconditional directly. An alternative is to prove the disjunction " $(P \text{ and } Q)$ or ($\text{not-}P$ and $\text{not-}Q$)", which itself can be inferred directly from either of its disjuncts—that is, because "iff" is truth-functional, " P iff Q " follows if P and Q have both been shown true, or both false.

Origin of iff

Usage of the abbreviation "iff" first appeared in print in John L. Kelley's 1955 book *General Topology*.^[2] Its invention is often credited to Paul Halmos, who wrote "I invented 'iff,' for 'if and only if'—but I could never believe I was really its first inventor."

Distinction from "if" and "only if"

1. "**If the fruit is an apple, then Madison will eat it.**" or "**Madison will eat the fruit if it is an apple.**"
(equivalent to "**Only if Madison will eat the fruit, is it an apple;**" or "**Madison will eat the fruit \leftarrow fruit is an apple**")

This states only that Madison will eat fruits that are apples. It does not, however, preclude the possibility that Madison might also have occasion to eat bananas. Maybe she will, maybe she will not—the sentence does not tell us. All that is known for certain is that she will eat any and all apples that she happens upon. That the fruit is an apple is a *sufficient* condition for Madison to eat the fruit.

2. "**Only if the fruit is an apple, will Madison eat it.**" or "**Madison will eat the fruit only if it is an apple.**"
(equivalent to "**If Madison will eat the fruit, then it is an apple**" or "**Madison will eat the fruit \rightarrow fruit is an apple**")

This states that the only fruit Madison will eat is an apple. It does not, however, preclude the possibility that Madison will refuse an apple if it is made available, in contrast with (1), which requires Madison to eat any available apple. In this case, that a given fruit is an apple is a *necessary* condition for Madison to be eating it. It is not a sufficient condition since Madison might not eat any and all apples she is given.

3. "**If and only if the fruit is an apple will Madison eat it**" or "**Madison will eat the fruit if and only if it is an apple**" or "**Madison will eat the fruit \leftrightarrow fruit is an apple**."

This, however, makes it quite clear that Madison will eat all and only those fruits that are apples. She will not leave any such fruit uneaten, and she will not eat any other type of fruit. That a given fruit is an apple is both a necessary and a sufficient condition for Madison to eat the fruit.

Sufficiency is the inverse of necessity. That is to say, given $P \rightarrow Q$ (i.e. if P then Q), P would be a sufficient condition for Q , and Q would be a necessary condition for P . Also, given $P \rightarrow Q$, it is true that $\neg Q \rightarrow \neg P$ (where \neg is the negation operator, i.e. "not"). This means that the relationship between P and Q , established by $P \rightarrow Q$, can be expressed in the following, all equivalent, ways:

- P is sufficient for Q
- Q is necessary for P
- $\neg Q$ is sufficient for $\neg P$
- $\neg P$ is necessary for $\neg Q$

As an example, take (1), above, which states $P \rightarrow Q$, where P is "the fruit in question is an apple" and Q is "Madison will eat the fruit in question". The following are four equivalent ways of expressing this very relationship:

If the fruit in question is an apple, then Madison will eat it.

Only if Madison will eat the fruit in question, is it an apple.

If Madison will not eat the fruit in question, then it is not an apple.

Only if the fruit in question is not an apple, will Madison not eat it.

So we see that (2), above, can be restated in the form of *if...then* as "If Madison will eat the fruit in question, then it is an apple"; taking this in conjunction with (1), we find that (3) can be stated as "If the fruit in question is an apple, then Madison will eat it; AND if Madison will eat the fruit, then it is an apple".

Advanced considerations

Philosophical interpretation

A sentence that is composed of two other sentences joined by "iff" is called a *biconditional*. "Iff" joins two sentences to form a new sentence. It should not be confused with logical equivalence which is a description of a relation between two sentences. The biconditional "*A iff B*" *uses* the sentences *A* and *B*, describing a relation between the states of affairs which *A* and *B* describe. By contrast "*A* is logically equivalent to *B*" *mentions* both sentences: it describes a logical relation between those two sentences, and not a factual relation between whatever matters they describe. See use–mention distinction for more on the difference between *using* a sentence and *mentioning* it.

The distinction is a very confusing one, and has led many a philosopherWikipedia:Avoid weasel words astray. Certainly it is the case that when *A* is logically equivalent to *B*, "*A iff B*" is true. But the converse does not hold. Reconsidering the sentence:

If and only if the fruit is an apple will Madison eat it.

There is clearly no logical equivalence between the two halves of this particular biconditional. For more on the distinction, see W. V. Quine's *Mathematical Logic*, Section 5.

One way of looking at "*A* if and only if *B*" is that it means "*A* if *B*" (*B* implies *A*) and "*A* only when *B*" (*not B* implies *not A*). "*Not B* implies *not A*" means *A* implies *B*, so then there is two way implication.

Definitions

In philosophy and logic, "iff" is used to indicate definitions, since definitions are supposed to be universally quantified biconditionals. In mathematics and elsewhere, however, the word "if" is normally used in definitions, rather than "iff". This is due to the observation that "if" in the English language has a definitional meaning, separate from its meaning as a propositional connective. This separate meaning can be explained by noting that a definition (for instance: A group is "abelian" if it satisfies the commutative law; or: A grape is a "raisin" if it is well dried) is not an equivalence to be proved, but a rule for interpreting the term defined.

Examples

Here are some examples of true statements that use "iff" - true biconditionals (the first is an example of a definition, so it would normally have been written with "if"):

- A person is a bachelor *iff* that person is a marriageable man who has never married.
- "Snow is white" in English is true *iff* "*Schnee ist weiß*" in German is true.
- For any *p*, *q*, and *r*: $(p \& q) \& r \text{ iff } p \& (q \& r)$. (Since this is written using variables and "&", the statement would usually be written using " \leftrightarrow ", or one of the other symbols used to write biconditionals, in place of "iff").
- For any real numbers *x* and *y*, $x=y+1 \text{ iff } y=x-1$.
- A subset containing *n* elements of an *n*-dimensional vector space is linearly independent *iff* it spans the vector space.
- The triangular number $\frac{n(n+1)}{2}$ is an even perfect number *iff* $n = 2^p - 1$ is a Mersenne prime, with *p* being a prime number. As of Feb 2013, only 48 such even perfect numbers and Mersenne primes have been discovered.

- $y(x)$ is a solution to the differential equation $y=f(x,y)$ if and only if the curve associated with $y(x)$ is an integral curve of the direction field associated with $y=f(x,y)$.

Analogs

Other words are also sometimes emphasized in the same way by repeating the last letter; for example *orr* for "Or and only Or" (the exclusive disjunction).

The statement "(A iff B)" is equivalent to the statement "(not A or B) and (not B or A)," and is also equivalent to the statement "(not A and not B) or (A and B)".

It is also equivalent to: $\neg[(A \text{ or } B) \text{ and } (\text{not } A \text{ or not } B)]$,

or more simply:

$$\neg [(\neg A \vee \neg B) \wedge (A \vee B)]$$

which converts into

$$[(\neg A \wedge \neg B) \vee (A \wedge B)]$$

and

$$[(\neg A \vee B) \wedge (A \vee \neg B)]$$

which were given in verbal interpretations above.

More general usage

Iff is used outside the field of logic, wherever logic is applied, especially in mathematical discussions. It has the same meaning as above: it is an abbreviation for *if and only if*, indicating that one statement is both necessary and sufficient for the other. This is an example of mathematical jargon. (However, as noted above, *if*, rather than *iff*, is more often used in statements of definition.)

The elements of X are *all and only* the elements of Y is used to mean: "for any z in the domain of discourse, z is in X if and only if z is in Y ."

Notes

[1] $p \Leftrightarrow q$ (<http://www.wolframalpha.com/input/?i=p+<=>+q>). WolframAlpha

[2] *General Topology*, reissue ISBN 978-0-387-90125-1

Material nonimplication

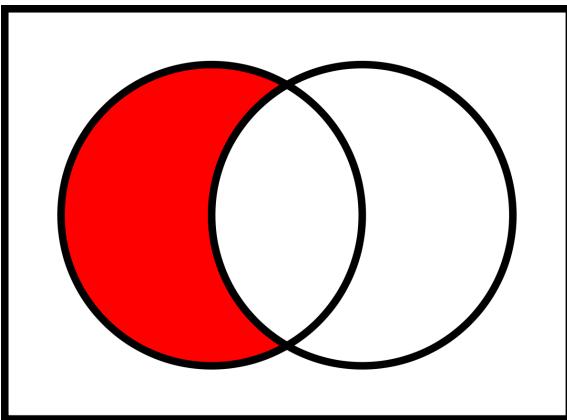
Material nonimplication or **abjunction** (latin *ab* = "from", *junctio* = "joining") is the negation of material implication. That is to say that for any two propositions P and Q, the material nonimplication from P to Q is true if and only if P does not imply Q.

It may be written using logical notation as:

$$p \Box q$$

$$Lpq$$

$$p \not\rightarrow q$$



Venn diagram of $A \not\rightarrow B$

$$\boxed{\text{O}} \wedge \boxed{\text{O}} \Leftrightarrow \boxed{\text{O}} \Leftrightarrow \neg \boxed{\text{O}}$$

Definition

Truth table

p	q	$\not\rightarrow$
T	T	F
T	F	T
F	T	F
F	F	F

Properties

falsehood-preserving: The interpretation under which all variables are assigned a truth value of 'false' produces a truth value of 'false' as a result of material nonimplication.

Symbol

The symbol for material nonimplication is simply a crossed-out material implication symbol. Its Unicode symbol is 8603 (decimal).

Natural language

Rhetorical

"It's not the case that p implies q."

"p but not q."

Colloquial

"Just because p, doesn't mean q."

Boolean Algebra

$(A' + B)'$

Computer Science

C: $(A \& (\sim B))$

References

Modal operator

A **modal connective** (or **modal operator**) is a logical connective for modal logic. It is an operator which forms propositions from propositions. In general, a modal operator has the "formal" property of being non-truth-functional, and is "intuitively" characterized by expressing a modal attitude (such as necessity, possibility, belief, or knowledge) about the proposition to which the operator is applied.

Modality interpreted

There are several ways to interpret modal operators in modal logic, including: alethic, deontic, axiological, epistemic, and doxastic.

Alethic

Alethic modal operators (M-operators) determine the fundamental conditions of possible worlds, especially causality, time-space parameters, and the action capacity of persons. They indicate the possibility, impossibility and necessity of actions, states of affairs, events, people, and qualities in the possible worlds.

Deontic

Deontic modal operators (P-operators) influence the construction of possible worlds as proscriptive or prescriptive norms, i.e. they indicate what is prohibited, obligatory, or permitted.

Axiological

Axiological modal operators (G-operators) transform the world's entities into values and disvalues as seen by a social group, a culture, or a historical period. Axiological modalities are highly subjective categories: what is good for one person may be considered as bad by another one.

Epistemic

Epistemic modal operators (K-operators) reflect the level of knowledge, ignorance and belief in the possible world.

Doxastic

Doxastic modal operators express belief in statements.

Negation

In logic, **negation**, also called **logical complement**, is an operation that essentially takes a proposition p to another proposition "not p ", written $\neg p$, which is interpreted intuitively as being true when p is false and false when p is true. Negation is thus a unary (single-argument) logical connective. It may be applied as an operation on propositions, truth values, or semantic values more generally. In classical logic, negation is normally identified with the truth function that takes *truth* to *falsity* and vice versa. In intuitionistic logic, according to the Brouwer–Heyting–Kolmogorov interpretation, the negation of a proposition p is the proposition whose proofs are the refutations of p .

Definition

Classical negation is an operation on one logical value, typically the value of a proposition, that produces a value of *true* when its operand is false and a value of *false* when its operand is true. So, if statement A is true, then $\neg A$ (pronounced "not A ") would therefore be false; and conversely, if $\neg A$ is true, then A would be false.

The truth table of $\neg p$ is as follows:

Truth table of $\neg p$

p	$\neg p$
True	False
False	True

Classical negation can be defined in terms of other logical operations. For example, $\neg p$ can be defined as $p \rightarrow F$, where " \rightarrow " is logical consequence and F is absolute falsehood. Conversely, one can define F as $p \& \neg p$ for any proposition p , where " $\&$ " is logical conjunction. The idea here is that any contradiction is false. While these ideas work in both classical and intuitionistic logic, they do not work in Brazilian logic, where contradictions are not necessarily false. But in classical logic, we get a further identity: $p \rightarrow q$ can be defined as $\neg p \vee q$, where " \vee " is logical disjunction: "not p , or q ".

Algebraically, classical negation corresponds to complementation in a Boolean algebra, and intuitionistic negation to pseudocomplementation in a Heyting algebra. These algebras provide a semantics for classical and intuitionistic logic respectively.

Notation

The negation of a proposition p is notated in different ways in various contexts of discussion and fields of application. Among these variants are the following:

Notation	Vocalization
$\neg p$	not p
\bar{p}	not p
$\sim p$	not p
Np	en p
p'	p prime, p complement
\bar{p}	p bar, bar p
$!p$	bang p not p

In Set Theory \ is also used to indicate 'not member of': $U \setminus A$ is the set of all members of U that are not members of A .

No matter how it is notated or symbolized, the negation $\neg p$ / \bar{p} can be read as "it is not the case that p ", "not that p ", or usually more simply (though not grammatically) as "not p ".

Properties

Double negation

Within a system of classical logic, double negation, that is, the negation of the negation of a proposition p , is logically equivalent to p . Expressed in symbolic terms, $\neg\neg p \Leftrightarrow p$. In intuitionistic logic, a proposition implies its double negation but not conversely. This marks one important difference between classical and intuitionistic negation. Algebraically, classical negation is called an involution of period two.

However, in intuitionistic logic we do have the equivalence of $\neg\neg\neg p$ and $\neg p$. Moreover, in the propositional case, a sentence is classically provable if its double negation is intuitionistically provable. This result is known as Glivenko's theorem.

Distributivity

De Morgan's laws provide a way of distributing negation over disjunction and conjunction :

$$\begin{aligned}\neg(a \vee b) &\equiv (\neg a \wedge \neg b), \text{ and} \\ \neg(a \wedge b) &\equiv (\neg a \vee \neg b).\end{aligned}$$

Linearity

In Boolean algebra, a linear function is one such that:

If there exists $a_0, a_1, \dots, a_n \in \{0,1\}$ such that $f(b_1, \dots, b_n) = a_0 \oplus (a_1 \wedge b_1) \oplus \dots \oplus (a_n \wedge b_n)$, for all $b_1, \dots, b_n \in \{0,1\}$.

Another way to express this is that each variable always makes a difference in the truth-value of the operation or it never makes a difference. Negation is a linear logical operator.

Self dual

In Boolean algebra a self dual function is one such that:

$f(a_1, \dots, a_n) = \neg f(\neg a_1, \dots, \neg a_n)$ for all $a_1, \dots, a_n \in \{0,1\}$. Negation is a self dual logical operator.

Rules of inference

There are a number of equivalent ways to formulate rules for negation. One usual way to formulate classical negation in a natural deduction setting is to take as primitive rules of inference *negation introduction* (from a derivation of p to both q and $\neg q$, infer $\neg p$; this rule also being called *reductio ad absurdum*), *negation elimination* (from p and $\neg p$ infer q ; this rule also being called *ex falso quodlibet*), and *double negation elimination* (from $\neg\neg p$ infer p). One obtains the rules for intuitionistic negation the same way but by excluding double negation elimination.

Negation introduction states that if an absurdity can be drawn as conclusion from p then p must not be the case (i.e. p is false (classically) or refutable (intuitionistically) or etc.). Negation elimination states that anything follows from an absurdity. Sometimes negation elimination is formulated using a primitive absurdity sign \perp . In this case the rule says that from p and $\neg p$ follows an absurdity. Together with double negation elimination one may infer our originally formulated rule, namely that anything follows from an absurdity.

Typically the intuitionistic negation $\neg p$ of p is defined as $p \rightarrow \perp$. Then negation introduction and elimination are just special cases of implication introduction (conditional proof) and elimination (modus ponens). In this case one must also add as a primitive rule *ex falso quodlibet*.

Programming

As in mathematics, negation is used in computer science to construct logical statements.

```
if (! (r == t))
{
    /*...statements executed when r does NOT equal t...*/
}
```

The "!" signifies logical NOT in B, C, and languages with a C-inspired syntax such as C++, Java, JavaScript, Perl, and PHP. "NOT" is the operator used in ALGOL 60, BASIC, and languages with an ALGOL- or BASIC-inspired syntax such as Pascal, Ada, Eiffel and Seed7. Some languages (C++, Perl, etc.) provide more than one operator for negation. A few languages like PL/I and Ratfor use \neg for negation. Some modern computers and operating systems will display \neg as ! on files encoded in ASCII. Most modern languages allow the above statement to be shortened from `if (! (r == t))` to `if (r != t)`, which allows sometimes, when the compiler/interpreter is not able to optimize it, faster programs.

In computer science there is also *bitwise negation*. This takes the value given and switches all the binary 1s to 0s and 0s to 1s. See bitwise operation. This is often used to create ones' complement or " \sim " in C or C++ and two's complement (just simplified to " $-$ " or the negative sign since this is equivalent to taking the arithmetic negative value of the number) as it basically creates the opposite (negative value equivalent) or mathematical complement of the value (where both values are added together they create a whole).

To get the absolute (positive equivalent) value of a given integer the following would work as the " $-$ " changes it from negative to positive (it is negative because " $x < 0$ " yields true)

```
unsigned int abs(int x)
{
    if (x < 0)
        return -x;
```

```
    else
        return x;
}
```

To demonstrate logical negation:

```
unsigned int abs(int x)
{
    if (! (x < 0))
        return x;
    else
        return -x;
}
```

Inverting the condition and reversing the outcomes produces code that is logically equivalent to the original code, i.e. will have identical results for any input (note that depending on the compiler used, the actual instructions performed by the computer may differ).

This convention occasionally surfaces in written speech, as computer-related slang for *not*. The phrase `!voting`, for example, means "not voting".

Kripke semantics

In Kripke semantics where the semantic values of formulae are sets of possible worlds, negation can be taken to mean set-theoretic complementation.^[citation needed] (See also possible world semantics.)

References

Further reading

- Gabbay, Dov, and Wansing, Heinrich, eds., 1999. *What is Negation?*, Kluwer.
- Horn, L., 2001. *A Natural History of Negation*, University of Chicago Press.
- G. H. von Wright, 1953–59, "On the Logic of Negation", *Commentationes Physico-Mathematicae* 22.
- Wansing, Heinrich, 2001, "Negation", in Goble, Lou, ed., *The Blackwell Guide to Philosophical Logic*, Blackwell.
- Marco Tettamanti, Rosa Manenti, Pasquale A. Della Rosa, Andrea Falini, Daniela Perani, Stefano F. Cappa and Andrea Moro (2008). "Negation in the brain: Modulating action representation", NeuroImage Volume 43, Issue 2, 1 November 2008, pages 358–367, <http://dx.doi.org/10.1016/j.neuroimage.2008.08.004/>

External links

- Hazewinkel, Michiel, ed. (2001), "Negation" (<http://www.encyclopediaofmath.org/index.php?title=p/n066170>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- NOT (<http://mathworld.wolfram.com/NOT.html>), on MathWorld

Parity function

In Boolean algebra, a **parity function** is a Boolean function whose value is 1 if the input vector has an odd number of ones.

The parity function is notable for its role in theoretical investigation of circuit complexity of Boolean functions.

Definition

The n -variable parity function is the Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with the property that $f(x) = 1$ if and only if the number of ones in the vector $x \in \{0, 1\}^n$ is odd. In other words, f is defined as follows:

$$f(x) = x_1 \oplus x_2 \oplus \dots \oplus x_n.$$

Properties

Parity only depends on the number of ones and is therefore a symmetric Boolean function.

The n -variable parity function and its negation are the only Boolean functions for which all disjunctive normal forms have the maximal number of 2^{n-1} monomials of length n and all conjunctive normal forms have the maximal number of 2^{n-1} clauses of length n .^[1]

Circuit complexity

In the early 1980s, Merrick Furst, James Saxe and Michael Sipser^[2] and independently Miklós Ajtai^[3] established super-polynomial lower bounds on the size of constant-depth Boolean circuits for the parity function, i.e., they showed that polynomial-size constant-depth circuits cannot compute the parity function. Similar results were also established for the majority, multiplication and transitive closure functions, by reduction from the parity function.

Håstad (1987) established tight exponential lower bounds on the size of constant-depth Boolean circuits for the parity function. Håstad's Switching Lemma is the key technical tool used for these lower bounds and Johan Håstad was awarded the Gödel Prize for this work in 1994. The precise result is that depth- k circuits with AND, OR, and NOT gates require size $\exp(\Omega(n^{\frac{1}{k-1}}))$ to compute the parity function. This is asymptotically almost optimal as there are depth- k circuits computing parity which have size $\exp(O(n^{\frac{1}{k-1}}))$.

Infinite version

An infinite parity function is a function $f : \{0, 1\}^\omega \rightarrow \{0, 1\}$ mapping every infinite binary string to 0 or 1, having the following property: if w and v are infinite binary strings differing only on finite number of coordinates then $f(w) = f(v)$ if and only if w and v differ on even number of coordinates.

Assuming axiom of choice it can be easily proved that parity functions exist and there are 2^c many of them - as many as the number of all functions from $\{0, 1\}^\omega$ to $\{0, 1\}$. It is enough to take one representative per equivalence class of relation \approx defined as follows: $w \approx v$ iff w and v differ at finite number of coordinates.

Having such representatives, we can map all of them to 0; the rest of f values are deducted unambiguously.

Infinite parity functions are often used in theoretical Computer Science and Set Theory because of their simple definition and - on the other hand - their descriptive complexity. For example, it can be shown that an inverse image $f^{-1}[0]$ is a non-Borel set.

References

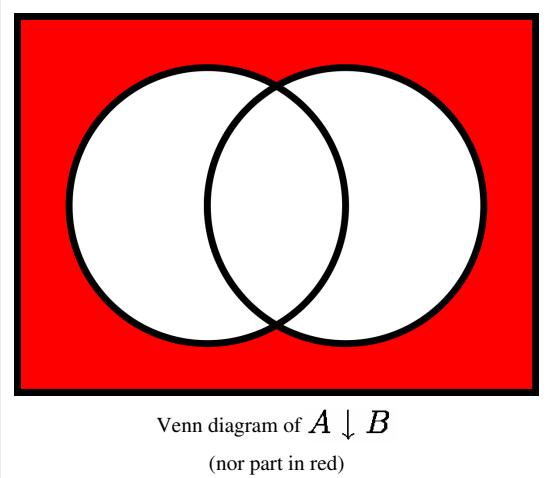
- [1] Ingo Wegener, Randall J. Pruim, *Complexity Theory*, 2005, ISBN 3-540-21045-8, p. 260 (http://books.google.com/books?id=u7DZSDSUYIQC&pg=PA261&lpg=PA261&dq=%E2%80%9Cparity+function%E2%80%9D+H%C3%A4stad&source=bl&ots=HNQ-Jx67yy&sig=qOg_lAiE3JbqsDdQO0rrmgxJmDs&hl=en&ei=U9PjSfGYDYjaswOCkNysCQ&sa=X&oi=book_result&ct=result&resnum=3#PPA260,M1)
- [2] Merrick Furst, James Saxe and Michael Sipser, "Parity, Circuits, and the Polynomial-Time Hierarchy", *Annu. Int'l. Symp. Found. Computer Sci.*, 1981, *Theory of Computing Systems*, vol. 17, no. 1, 1984, pp. 13–27,
- [3] Miklós Ajtai, " n -Formulae on Finite Structures", *Annals of Pure and Applied Logic*, 24 (1983) 1–48.
- Håstad, Johan (1987), *Computational limitations of small depth circuits* (<http://www.nada.kth.se/~johanh/thesis.pdf>), Ph.D. thesis, Massachusetts Institute of Technology.

Peirce arrow

In boolean logic, **logical nor** or **joint denial** is a truth-functional operator which produces a result that is the negation of logical or. That is, a sentence of the form $(p \text{ NOR } q)$ is true precisely when neither p nor q is true—i.e. when both of p and q are *false*. In grammar, **nor** is a coordinating conjunction.

The NOR operator is also known as **Peirce's arrow** — Charles Sanders Peirce introduced the symbol \downarrow for it, and demonstrated that the logical NOR is completely expressible: by combining uses of the logical NOR it is possible to express any logical operation on two variables. Thus, as with its dual, the NAND operator (a.k.a. the Sheffer stroke — symbolized as either \mid or !), NOR can be used by itself, without any other logical operator, to constitute a logical formal system (making NOR functionally complete). It is also known as Quine's dagger (his symbol was \dagger), the **ampheck** (from Greek αμφηκης, cutting both ways; compare *amphi-*) by Peirce,^[1] or "neither-nor".

One way of expressing $p \text{ NOR } q$ is $\overline{p \vee q}$, where the symbol \vee signifies OR and the bar signifies the negation of the expression under it: in essence, simply $\neg(p \vee q)$. Other ways of expressing $p \text{ NOR } q$ are Xpq , and $\overline{p + q}$. The computer used in the spacecraft that first carried humans to the moon, the Apollo Guidance Computer, was constructed entirely using NOR gates with three inputs. [citation needed]



Definition

The **NOR operation** is a logical operation on two logical values, typically the values of two propositions, that produces a value of *true* if and only if both operands are *false*. In other words, it produces a value of *false* if and only if at least one operand is *true*.

Truth table

The truth table of **A NOR B** (also written as **A ↓ B**) is as follows:

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

Properties

Logical NOR does not possess any of the five qualities (truth-preserving, false-preserving, linear, monotonic, self-dual) required to be absent from at least one member of a set of functionally complete operators. Thus, the set containing only NOR suffices as a complete set.

Introduction, elimination, and equivalencies

NOR has the interesting feature that all other logical operators can be expressed by interlaced NOR operations. The logical NAND operator also has this ability.

The logical NOR \downarrow is the negation of the disjunction:

$P \downarrow Q$	\Leftrightarrow	$\neg(P \vee Q)$
	\Leftrightarrow	

Expressed in terms of NOR \downarrow , the usual operators of propositional logic are:

$\neg P$	\Leftrightarrow	$P \downarrow P$
	\Leftrightarrow	
$P \rightarrow Q$	\Leftrightarrow	$((P \downarrow P) \downarrow Q)$
	\Leftrightarrow	
$P \wedge Q$	\Leftrightarrow	$(P \downarrow P) \downarrow (Q \downarrow Q)$
	\Leftrightarrow	
$P \vee Q$	\Leftrightarrow	$(P \downarrow Q) \downarrow (P \downarrow Q)$
	\Leftrightarrow	

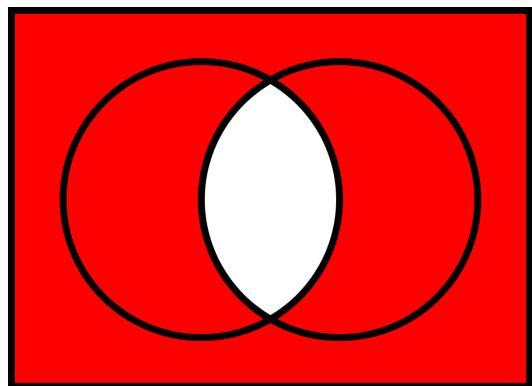
References

- [1] C.S. Peirce, CP 4.264

Sheffer stroke

In Boolean functions and propositional calculus, the **Sheffer stroke**, named after Henry M. Sheffer, written "!" (see vertical bar, not to be confused with "||" which is often used to represent disjunction), " Dpq ", or " \uparrow ", denotes a logical operation that is equivalent to the negation of the conjunction operation, expressed in ordinary language as "not both". It is also called **nand** ("not and") or the **alternative denial**, since it says in effect that at least one of its operands is false. In Boolean algebra and digital electronics it is known as the **NAND operation**.

Like its dual, the NOR operator (a.k.a. the Peirce arrow or Quine dagger), NAND can be used by itself, without any other logical operator, to constitute a logical formal system (making NAND functionally complete). This property makes the NAND gate crucial to modern digital electronics, including its use in NAND flash memory and computer processor design.



Venn diagram of $A \uparrow B$

Definition

The **NAND operation** is a logical operation on two logical values. It produces a value of true, if — and only if — at least one of the propositions is false.

Truth table

The truth table of **A NAND B** (also written as $A \mid B$, Dpq , or $A \uparrow B$) is as follows:

INPUT		OUTPUT
A	B	$A \text{ NAND } B$
0	0	1
0	1	1
1	0	1
1	1	0

History

The stroke is named after Henry M. Sheffer, who provided (Sheffer 1913) an axiomatization of Boolean algebras using the stroke, and proved its equivalence to a standard formulation thereof by Huntington employing the familiar operators of propositional logic (and, or, not). Because of self-duality of Boolean algebras, Sheffer's axioms are equally valid for either of the NAND or NOR operations in place of the stroke. Sheffer interpreted the stroke as a sign for non-disjunction (NOR) in his paper, mentioning non-conjunction only in a footnote and without a special sign for it. It was Jean Nicod who first used the stroke as a sign for non-conjunction (NAND) in a paper of 1917 and which has since become current practice.

Charles Sanders Peirce (1880) had discovered the functional completeness of NAND or NOR more than 30 years earlier, using the term *ampheck* (for 'cutting both ways'), but he never published his finding.

Properties

NAND does not possess any of the following five properties, each of which is required to be absent from, and the absence of all of which is sufficient for, at least one member of a set of functionally complete operators: truth-preservation, falsity-preservation, linearity, monotonicity, self-duality. (An operator is truth- (falsity-) preserving if its value is truth (falsity) whenever all of its arguments are truth (falsity).) Therefore {NAND} is a functionally complete set.

This can also be realized as follows: All three elements of the functionally complete set {AND, OR, NOT} can be constructed using only NAND. Thus the set {NAND} must be functionally complete as well.

Introduction, elimination, and equivalencies

The Sheffer stroke \uparrow is the negation of the conjunction:

$P \uparrow Q$	\Leftrightarrow	$\neg(P \wedge Q)$
	\Leftrightarrow	

Expressed in terms of NAND \uparrow , the usual operators of propositional logic are:

$\neg P$	\Leftrightarrow	$P \uparrow P$	$P \rightarrow Q$	\Leftrightarrow	P	\uparrow	$(Q \uparrow Q)$	\Leftrightarrow	P	\uparrow	$(P \uparrow Q)$
	\Leftrightarrow			\Leftrightarrow		\uparrow		\Leftrightarrow		\uparrow	
$P \wedge Q$	\Leftrightarrow	$(P \uparrow Q) \uparrow (P \uparrow Q)$	$P \vee Q$	\Leftrightarrow	$(P \uparrow P) \uparrow (Q \uparrow Q)$	\uparrow	$(Q \uparrow Q)$	\Leftrightarrow	$(P \uparrow P) \uparrow (Q \uparrow Q)$	\uparrow	$(P \uparrow Q)$
	\Leftrightarrow			\Leftrightarrow		\uparrow		\Leftrightarrow		\uparrow	

Formal system based on the Sheffer stroke

The following is an example of a formal system based entirely on the Sheffer stroke, yet having the functional expressiveness of the propositional logic:

Symbols

p_n for natural numbers n
 (\uparrow)

The Sheffer stroke commutes but does not associate (e.g., $(T \uparrow T) \uparrow F = T$, but $T \uparrow (T \uparrow F) = F$). Hence any formal system including the Sheffer stroke must also include a means of indicating grouping. We shall employ '(' and ')' to this effect.

We also write p, q, r, \dots instead of p_0, p_1, p_2, \dots

Syntax

Construction Rule I: For each natural number n , the symbol p_n is a well-formed formula (wff), called an atom.

Construction Rule II: If X and Y are wffs, then $(X|Y)$ is a wff.

Closure Rule: Any formulae which cannot be constructed by means of the first two Construction Rules are not wffs.

The letters U, V, W, X , and Y are metavariables standing for wffs.

A decision procedure for determining whether a formula is well-formed goes as follows: "deconstruct" the formula by applying the Construction Rules backwards, thereby breaking the formula into smaller subformulae. Then repeat this recursive deconstruction process to each of the subformulae. Eventually the formula should be reduced to its atoms, but if some subformula cannot be so reduced, then the formula is not a wff.

Calculus

All wffs of the form

$$((U|(V|W))|((Y|(YY))|((X|V)|((U|X)|(U|X))))))$$

are axioms. Instances of

$$(U|(V|W)), U \vdash W$$

are inference rules.

Simplification

Since the only connective of this logic is $|$, the symbol $|$ could be discarded altogether, leaving only the parentheses to group the letters. A pair of parentheses must always enclose a pair of wffs. Examples of theorems in this simplified notation are

$$\begin{aligned} &(p(p(q(q((pq)(pq)))))), \\ &(p(p((qq)(pp))))). \end{aligned}$$

The notation can be simplified further, by letting

$$(U) := (UU)$$

$$((U)) \equiv U$$

for any U . This simplification causes the need to change some rules:

1. More than two letters are allowed within parentheses.
2. Letters or wffs within parentheses are allowed to commute.
3. Repeated letters or wffs within a same set of parentheses can be eliminated.

The result is a parenthetical version of the Peirce existential graphs.

Another way to simplify the notation is to eliminate parenthesis by using Polish Notation. For example, the earlier examples with only parenthesis could be rewritten using only strokes as follows

$(p(p(q(q((pq)(pq))))))$ becomes

$|p|p|q|q||p|q|p|q$, and

$(p(p((qq)(pp))))$ becomes,

$|p|p||q|q|p|p$.

This follows the same rules as the parenthesis version, with opening parenthesis replaced with a Sheffer stroke and the (redundant) closing parenthesis removed.

Notes

References

- Bocheński, Józef Maria (1960), *Précis of Mathematical Logic*, translated from the French and German editions by Otto Bird, Dordrecht, South Holland: D. Reidel.
- Church, Alonzo, (1956) *Introduction to mathematical logic*, Vol. 1, Princeton: Princeton University Press.
- Nicod, Jean G. P., (1917) "A Reduction in the Number of Primitive Propositions of Logic", *Proceedings of the Cambridge Philosophical Society*, Vol. 19, pp. 32–41.
- Charles Sanders Peirce, 1880, "A Boolean[sic] Algebra with One Constant", in Hartshorne, C. and Weiss, P., eds., (1931–35) *Collected Papers of Charles Sanders Peirce*, Vol. 4: 12–20, Cambridge: Harvard University Press.
- H. M. Sheffer, 1913. "A set of five independent postulates for Boolean algebras, with application to logical constants," *Transactions of the American Mathematical Society* 14: pp. 481–488.

External links

- <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nand.html>
- implementations of 2 and 4-input NAND gates (<http://www.scs.swarthmore.edu/users/06/adem/engin/e77vlsi/lab3/>)
- Proofs of some axioms by Stroke function by Yasuo Setô (<http://projecteuclid.org/DPubS?verb=Display&version=1.0&service=UI&handle=euclid.pja/1195520940&page=record>) @ Project Euclid (<http://projecteuclid.org>)

Sole sufficient operator

In logic, a **functionally complete** set of logical connectives or Boolean operators is one which can be used to express all possible truth tables by combining members of the set into a Boolean expression.^{[1][2]} A well-known complete set of connectives is { AND, NOT }, consisting of binary conjunction and negation. The singleton sets { NAND } and { NOR } are also functionally complete.

In a context of propositional logic, functionally complete sets of connectives are also called **(expressively) adequate**.^[3]

From the point of view of digital electronics, functional completeness means that every possible logic gate can be realized as a network of gates of the types prescribed by the set. In particular, all logic gates can be assembled from either only binary NAND gates, or only binary NOR gates.

Formal definition

Given the Boolean domain $\mathbf{B} = \{0,1\}$, a set F of Boolean functions $f_i: \mathbf{B}^n \rightarrow \mathbf{B}$ is **functionally complete** if the clone on \mathbf{B} generated by the basic functions f_i contains all functions $f: \mathbf{B}^n \rightarrow \mathbf{B}$, for all *strictly positive* integers $n \geq 1$. In other words, the set is functionally complete if every Boolean function that takes at least one variable can be expressed in terms of the functions f_i . Since every Boolean function of at least one variable can be expressed in terms of binary Boolean functions, F is functionally complete if and only if every binary Boolean function can be expressed in terms of the functions in F .

A more natural condition would be that the clone generated by F consist of all functions $f: \mathbf{B}^n \rightarrow \mathbf{B}$, for all integers $n \geq 0$. However, the examples given above are not functionally complete in this stronger sense because it is not possible to write a nullary function, i.e. a constant expression, in terms of F if F itself does not contain at least one nullary function. With this stronger definition, the smallest functionally complete sets would have 2 elements.

Another natural condition would be that the clone generated by F together with the two nullary constant functions be functionally complete or, equivalently, functionally complete in the strong sense of the previous paragraph. The example of the Boolean function given by $S(x, y, z) = z$ if $x = y$ and $S(x, y, z) = x$ otherwise shows that this condition is strictly weaker than functional completeness.

Informal definition

Modern texts on logic typically take as primitive some subset of the connectives: conjunction (\wedge), or Kpq ; disjunction (\vee), or Apq ; negation (\neg), Np ; or material conditional (\rightarrow), or Cpq ; and possibly the biconditional (\leftrightarrow), or Epq . These connectives are functionally complete. However, they do not form a minimal functionally complete set, as the conditional and biconditional may be defined as:

$$\begin{aligned} A \rightarrow B &:= \neg A \vee B \\ A \leftrightarrow B &:= (A \rightarrow B) \wedge (B \rightarrow A). \end{aligned}$$

So $\{\neg, \wedge, \vee\}$ is also functionally complete. But then, \vee can be defined as

$$A \vee B := \neg(\neg A \wedge \neg B).$$

\wedge can also be defined in terms of \vee in a similar manner.

It is also the case that \vee can be defined in terms of \rightarrow as follows:

$$A \vee B := (A \rightarrow B) \rightarrow B.$$

No further simplifications are possible. Hence \neg and one of $\{\wedge, \vee, \rightarrow\}$ are each minimal functionally complete subsets of $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$.

Characterization of functional completeness

Emil Post proved that a set of logical connectives is functionally complete if and only if it is not a subset of any of the following sets of connectives:

- The monotonic connectives; changing the truth value of any connected variables from **F** to **T** without changing any from **T** to **F** never makes these connectives change their return value from **T** to **F**, e.g. $\vee, \wedge, \top, \perp$.
- The affine connectives, such that each connected variable either always or never affects the truth value these connectives return, e.g. $\neg, \top, \perp, \leftrightarrow, \not\leftrightarrow$.
- The **self-dual** connectives, which are equal to their own de Morgan dual; if the truth values of all variables are reversed, so is the truth value these connectives return, e.g. $\neg, MAJ(p,q,r)$.
- The **truth-preserving** connectives; they return the truth value **T** under any interpretation which assigns **T** to all variables, e.g. $\vee, \wedge, \top, \rightarrow, \leftrightarrow$.
- The **falsity-preserving** connectives; they return the truth value **F** under any interpretation which assigns **F** to all variables, e.g. $\vee, \wedge, \perp, \not\rightarrow, \not\leftrightarrow$.

In fact, Post gave a complete description of the lattice of all clones (sets of operations closed under composition and containing all projections) on the two-element set $\{\mathbf{T}, \mathbf{F}\}$, nowadays called Post's lattice, which implies the above result as a simple corollary: the five mentioned sets of connectives are exactly the maximal clones.

Minimal functionally complete operator sets

When a single logical connective or Boolean operator is functionally complete by itself, it is called a **Sheffer function**^[4] or sometimes a **sole sufficient operator**. There are no unary operators with this property, and the only binary Sheffer functions — NAND and NOR are dual. These were discovered but not published by Charles Sanders Peirce around 1880, and rediscovered independently and published by Henry M. Sheffer in 1913. In digital electronics terminology, the binary NAND gate and the binary NOR gate are the only binary universal logic gates.

The following are the minimal functionally complete sets of logical connectives with arity ≤ 2 :^[5]

One element

{NAND}, {NOR}.

Two elements

{ \vee, \neg }, { \wedge, \neg }, { \rightarrow, \neg }, { \leftarrow, \neg }, { \rightarrow, \perp }, { \leftarrow, \perp }, { $\rightarrow, \not\leftrightarrow$ }, { $\leftarrow, \not\leftrightarrow$ }, { $\rightarrow, \not\rightarrow$ }, { $\rightarrow, \not\leftarrow$ }, { $\leftarrow, \not\rightarrow$ }, { $\leftarrow, \not\leftarrow$ }, { $\not\rightarrow, \neg$ }, { $\not\leftarrow, \neg$ }, { $\not\rightarrow, \top$ }, { $\not\leftarrow, \top$ }, { $\not\rightarrow, \leftrightarrow$ }, { $\not\leftarrow, \leftrightarrow$ }.

Three elements

{ $\vee, \leftrightarrow, \perp$ }, { $\vee, \leftrightarrow, \not\leftrightarrow$ }, { $\vee, \not\leftrightarrow, \top$ }, { $\wedge, \leftrightarrow, \perp$ }, { $\wedge, \leftrightarrow, \not\leftrightarrow$ }, { $\wedge, \not\leftrightarrow, \top$ }, { $\rightarrow, \leftrightarrow, \perp$ }, { $\rightarrow, \leftrightarrow, \not\leftrightarrow$ }, { $\rightarrow, \not\leftrightarrow, \top$ }, { $\leftarrow, \leftrightarrow, \perp$ }, { $\leftarrow, \leftrightarrow, \not\leftrightarrow$ }, { $\leftarrow, \not\leftrightarrow, \top$ }, { $\not\rightarrow, \leftrightarrow, \perp$ }, { $\not\rightarrow, \leftrightarrow, \not\leftrightarrow$ }, { $\not\rightarrow, \not\leftrightarrow, \top$ }, { $\not\leftarrow, \leftrightarrow, \perp$ }, { $\not\leftarrow, \leftrightarrow, \not\leftrightarrow$ }, { $\not\leftarrow, \not\leftrightarrow, \top$ }.

There are no minimal functionally complete sets of more than three at most binary logical connectives. Constant unary or binary connectives and binary connectives that depend only on one of the arguments have been suppressed to keep the list readable. E.g. the set consisting of binary \vee and the binary connective given by negation of the first argument (ignoring the second) is another minimal functionally complete set.

Examples

- Examples of using the NAND completeness. As illustrated by,^[6]
 - $\neg A = A \text{ NAND } A$
 - $A \wedge B = \neg(A \text{ NAND } B) = (A \text{ NAND } B) \text{ NAND } (A \text{ NAND } B)$
 - $A \vee B = (A \text{ NAND } A) \text{ NAND } (B \text{ NAND } B)$
- Examples of using the NOR completeness. As illustrated by,^[7]
 - $\neg A = A \text{ NOR } A$
 - $A \wedge B = (A \text{ NOR } A) \text{ NOR } (B \text{ NOR } B)$
 - $A \vee B = (A \text{ NOR } B) \text{ NOR } (A \text{ NOR } B)$

Note that, an electronic circuit or a software function is optimized by the reuse, that reduce the number of gates. For instance, the " $A \wedge B$ " operation, when expressed by NAND gates, is implemented with the reuse of " $A \text{ NAND } B$ ",

$$X = (A \text{ NAND } B); A \wedge B = X \text{ NAND } X$$

In other domains

Apart from logical connectives (Boolean operators), functional completeness can be introduced in other domains. For example, a set of reversible gates is called functionally complete, if it can express every reversible operator.

The 3-input Fredkin gate is functionally complete reversible gate by itself – a sole sufficient operator. There are many other three-input universal logic gates, such as the Toffoli gate.

Set theory

There is an isomorphism between the Algebra of sets and the Boolean algebra, that is, they have the same structure. Then, if we map boolean operators into set operators, the "translated" above text are valid also for sets: there are many "minimal complete set of set-theory operators" that can generate any other set relations. The more popular "Minimal complete operator sets" are { \neg, \cap } and { \neg, \cup }.

References

- [1] . ("Complete set of logical connectives").
- [2] . ("[F]unctional completeness of [a] set of logical operators").
- [3] . (Defines "expressively adequate", shortened to "adequate set of connectives" in a section heading.)
- [4] The term was originally restricted to *binary* operations, but since the end of the 20th century it is used more generally. .
- [5] Wernick, William (1942) "Complete Sets of Logical Functions," *Transactions of the American Mathematical Society* 51: 117–32. In his list on the last page of the article, Wernick does not distinguish between \leftarrow and \rightarrow , or between UNIQ-math-0-a68bc4e06c66b481-QINU and UNIQ-math-1-a68bc4e06c66b481-QINU .
- [6] "NAND Gate Operations" at <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nand.html>
- [7] "NOR Gate Operations" at <http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/nor.html>
- Wernick, William (1942) "Complete Sets of Logical Functions," *Transactions of the American Mathematical Society* 51: 117–32.

Statement (logic)

In logic a **statement** is either (a) a meaningful declarative sentence that is either true or false, or (b) that which a true or false declarative sentence asserts. In the latter case, a statement is distinct from a sentence in that a sentence is only one formulation of a statement, whereas there may be many other formulations expressing the same statement.

Philosopher of language, Peter Strawson advocated the use of the term "statement" in sense (b) in preference to proposition. Strawson used the term "Statement" to make the point that two declarative sentences can make the same statement if they say the same thing in different ways. Thus in the usage advocated by Strawson, "All men are mortal." and "Every man is mortal." are two different sentences that make the same statement.

In either case a statement is viewed as a truth bearer.

Examples of sentences that are (or make) statements:

- "Socrates is a man."
- "A triangle has three sides."
- "Madrid is the capital of Spain."

Examples of sentences that are not (or do not make) statements:

- "Who are you?"
- "Run!"
- "Greenness perambulates"
- "I had one grunch but the eggplant over there."
- "The King of France is wise."
- "Broccoli tastes good."
- "Pegasus exists."

The first two examples are not **declarative** sentences and therefore are not (or do not make) statements. The third and fourth are declarative sentences but, lacking meaning, are neither true nor false and therefore are not (or do not make) statements. The fifth and sixth examples are meaningful declarative sentences, but are not statements but rather matters of opinion or taste. Whether or not the sentence "Pegasus exists." is a statement is a subject of debate among philosophers. Bertrand Russell held that it is a (false) statement. Strawson held it is not a statement at all.

Statement as an abstract entity

In some treatments "statement" is introduced in order to distinguish a sentence from its informational content. A statement is regarded as the information content of an information-bearing sentence. Thus, a sentence is related to the statement it bears like a numeral to the number it refers to. Statements are abstract logical entities, while sentences are grammatical entities.

Notes

References

- A. G. Hamilton, *Logic for Mathematicians*, Cambridge University Press, 1980, ISBN 0-521-29291-3.
- Rouse, David L., "Sentences, Statements and Arguments" ([http://people.uvawise.edu/philosophy/Logic Text/Chapter 2 Sentences, Statements and Arguments.pdf](http://people.uvawise.edu/philosophy/Logic%20Text/Chapter%202%20Sentences,%20Statements%20and%20Arguments.pdf)) (PDF), *A Practical Introduction to Formal Logic*.
- Ruzsa, Imre (2000), *Bevezetés a modern logikába*, Osiris tankönyvek, Budapest: Osiris, ISBN 963-379-978-3.
- Jasa Xenakis, "Sentence and Statement", "Analysis" Vol. 16, No. 4 (Mar., 1956), pp. 91-94 <http://www.jstor.org/pss/3326478/>
- Peter Millican, "Statements and Modality: Strawson, Quine and Wolfram", <http://philpapers.org/rec/MILSAM-2/>
- P. F. Strawson, "On Referring" in *Mind*, Vol 59 No 235 (Jul 1950) P. F. Strawson (<http://www.sol.lu.se/common/courses/LINC04/VT2010/Strawson1950.pdf/>)

Strict conditional

In logic, a **strict conditional** is a modal operator, that is, a logical connective of modal logic. It is logically equivalent to the material conditional of classical logic, combined with the necessity operator from modal logic. For any two propositions p and q , the formula $p \rightarrow q$ says that p materially implies q while $\Box(p \rightarrow q)$ says that p strictly implies q .^[1] Strict conditionals are the result of Clarence Irving Lewis's attempt to find a conditional for logic that can adequately express indicative conditionals in natural language.^[2] They have also been used in studying Molinist theology.^[3]

Avoiding paradoxes

The strict conditionals may avoid paradoxes of material implication. The following statement, for example, is not correctly formalized by material implication:

If Bill Gates had graduated in Medicine, then Elvis never died.

This condition should clearly be false: the degree of Bill Gates has nothing to do with whether Elvis is still alive. However, the direct encoding of this formula in classical logic using material implication leads to:

Bill Gates graduated in Medicine \rightarrow Elvis never died.

This formula is true because a formula $A \rightarrow B$ is true whenever the antecedent A is false. Hence, this formula is not an adequate translation of the original sentence. An encoding using the strict conditional is:

$\Box(\text{Bill Gates graduated in Medicine} \rightarrow \text{Elvis never died.})$

In modal logic, this formula means (roughly) that, in every possible world in which Bill Gates graduated in Medicine, Elvis never died. Since one can easily imagine a world where Bill Gates is a Medicine graduate and Elvis is dead, this formula is false. Hence, this formula seems a correct translation of the original sentence.

Problems

Although the strict conditional is much closer to being able to express natural language conditionals than the material conditional, it has its own problems with consequents that are necessarily true (such as $2 + 2 = 4$) or antecedents that are necessarily false.^[4] The following sentence, for example, is not correctly formalized by a strict conditional:

If Bill Gates graduated in Medicine, then $2 + 2 = 4$.

Using strict conditionals, this sentence is expressed as:

$\square(\text{Bill Gates graduated in Medicine} \rightarrow 2 + 2 = 4)$

In modal logic, this formula means that, in every possible world where Bill Gates graduated in medicine, it holds that $2 + 2 = 4$. Since $2 + 2$ is equal to 4 in all possible worlds, this formula is true, although it does not seem that the original sentence should be. A similar situation arises with $2 + 2 = 5$, which is necessarily false:

If $2 + 2 = 5$, then Bill Gates graduated in Medicine.

Some logicians view this situation as indicating that the strict conditional is still unsatisfactory. Others have noted that the strict conditional cannot adequately express counterfactual conditionals,^[5] and that it does not satisfy certain logical properties.^[6] In particular, the strict conditional is transitive, while the counterfactual conditional is not.^[7]

Some logicians, such as Paul Grice, have used conversational implicature to argue that, despite apparent difficulties, the material conditional is just fine as a translation for the natural language 'if...then...'. Others still have turned to relevance logic to supply a connection between the antecedent and consequent of provable conditionals.

References

- [1] Graham Priest, *An Introduction to Non-Classical Logic: From if to is*, 2nd ed, Cambridge University Press, 2008, ISBN 0-521-85433-4, p. 72. (<http://books.google.com/books?id=rMXVbmAw3YwC&pg=PA72>)
- [2] Nicholas Bunnin and Jiyuan Yu (eds), *The Blackwell Dictionary of Western Philosophy*, Wiley, 2004, ISBN 1-4051-0679-4, "strict implication," p. 660 (<http://books.google.com/books?id=OskKWIIYA7AC&pg=PA660>).
- [3] Jonathan L. Kvanvig, "Creation, Deliberation, and Molinism," in *Destiny and Deliberation: Essays in Philosophical Theology*, Oxford University Press, 2011, ISBN 0-19-969657-8, p. 127–136 (<http://books.google.com/books?id=nQliRGPVpTwC&pg=PA127>).
- [4] Roy A. Sorensen, *A Brief History of the Paradox: Philosophy and the labyrinths of the mind*, Oxford University Press, 2003, ISBN 0-19-515903-9, p. 105 (<http://books.google.com/books?id=PB8I0kHeKy4C&pg=PA105>).
- [5] Jens S. Allwood, Lars-Gunnar Andersson, and Östen Dahl, *Logic in Linguistics*, Cambridge University Press, 1977, ISBN 0-521-29174-7, p. 120 (<http://books.google.com/books?id=hXIxFPttDjgC&pg=PA120>).
- [6] Hans Rott and Vítězslav Horák, *Possibility and Reality: Metaphysics and Logic*, ontos verlag, 2003, ISBN 3-937202-24-2, p. 271 (<http://books.google.com/books?id=ov9kN3HyltAC&pg=PA271>).
- [7] John Bigelow and Robert Pargetter, *Science and Necessity*, Cambridge University Press, 1990, ISBN 0-521-39027-3, p. 116 (<http://books.google.com/books?id=O-onBdR7TPAC&pg=PA116>).

Bibliography

- Edgington, Dorothy, 2001, "Conditionals," in Goble, Lou, ed., *The Blackwell Guide to Philosophical Logic*. Blackwell.

For an introduction to non-classical logic as an attempt to find a better translation of the conditional, see:

- Priest, Graham, 2001. *An Introduction to Non-Classical Logic*. Cambridge Univ. Press.

For an extended philosophical discussion of the issues mentioned in this article, see:

- Mark Sainsbury, 2001. *Logical Forms*. Blackwell Publishers.
- Jonathan Bennett, 2003. *A Philosophical Guide to Conditionals*. Oxford Univ. Press.

Symmetric Boolean function

In mathematics, a **symmetric Boolean function** is a Boolean function whose value does not depend on the permutation of its input bits, i.e., it depends only on the number of ones in the input.^[1]

The definition implies that instead of the truth table, traditionally used to represent Boolean functions, one may use a more compact representation for an n -variable symmetric Boolean function: the $(n + 1)$ -vector, whose i -th entry ($i = 0, \dots, n$) is the value of the function on an input vector with i ones.

Special cases

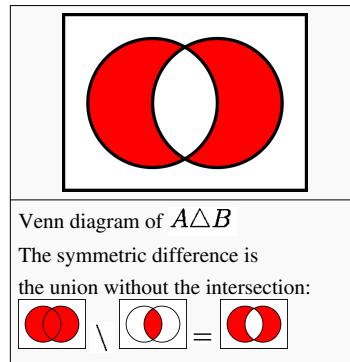
A number of special cases are recognized.

- **Threshold functions:** their value is 1 on input vectors with k or more ones for a fixed k
- **Exact-value functions:** their value is 1 on input vectors with k ones for a fixed k
- **Counting functions:** their value is 1 on input vectors with the number of ones congruent to $k \bmod m$ for fixed k, m
- **Parity functions:** their value is 1 if the input vector has odd number of ones.

References

[1] Ingo Wegener, "The Complexity of Symmetric Boolean Functions", in: *Computation Theory and Logic, Lecture Notes in Computer Science*, vol. 270, 1987, pp. 433-442

Symmetric difference



In mathematics, the **symmetric difference** of two sets is the set of elements which are in either of the sets and not in their intersection. The symmetric difference of the sets A and B is commonly denoted by

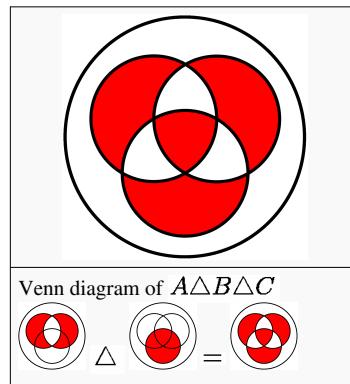
$$A \Delta B$$

or

$$A \ominus B.$$

For example, the symmetric difference of the sets $\{1, 2, 3\}$ and $\{3, 4\}$ is $\{1, 2, 4\}$. The symmetric difference of the set of all students and the set of all females consists of all male students together with all female non-students. The power set of any set becomes an abelian group under the operation of symmetric difference, with the empty set as the neutral element of the group and every element in this group being its own inverse. The power set of any set becomes a Boolean ring with symmetric difference as the addition of the ring and intersection as the multiplication of the ring.

Properties



The symmetric difference is equivalent to the union of both relative complements, that is:

$$A \triangle B = (A \setminus B) \cup (B \setminus A),$$

and it can also be expressed as the union of the two sets, minus their intersection:

$$A \triangle B = (A \cup B) \setminus (A \cap B),$$

or with the XOR operation:

$$A \triangle B = \{x : (x \in A) \oplus (x \in B)\}.$$

In particular, $A \triangle B \subseteq A \cup B$.

The symmetric difference is commutative and associative:

$$A \triangle B = B \triangle A,$$

$$(A \triangle B) \triangle C = A \triangle (B \triangle C).$$

Thus, the repeated symmetric difference is an operation on a multiset of sets giving the set of elements which are in an odd number of sets.

The symmetric difference of two repeated symmetric differences is the repeated symmetric difference of the join of the two multisets, where for each double set both can be removed. In particular:

$$(A \triangle B) \triangle (B \triangle C) = A \triangle C.$$

This implies a sort of triangle inequality: the symmetric difference of A and C is contained in the union of the symmetric difference of A and B and that of B and C . (But note that for the diameter of the symmetric difference the triangle inequality does not hold.)

The empty set is neutral, and every set is its own inverse:

$$A \triangle \emptyset = A,$$

$$A \triangle A = \emptyset.$$

Taken together, we see that the power set of any set X becomes an abelian group if we use the symmetric difference as operation. Because every element in this group is its own inverse, this is in fact a vector space over the field with 2 elements \mathbf{Z}_2 . If X is finite, then the singletons form a basis of this vector space, and its dimension is therefore equal to the number of elements of X . This construction is used in graph theory, to define the cycle space of a graph.

Intersection distributes over symmetric difference:

$$A \cap (B \triangle C) = (A \cap B) \triangle (A \cap C),$$

and this shows that the power set of X becomes a ring with symmetric difference as addition and intersection as multiplication. This is the prototypical example of a Boolean ring.

Further properties of the symmetric difference:

- $A \Delta B = A^c \Delta B^c$, where A^c , B^c is A 's complement, B 's complement, respectively, relative to any (fixed) set that contains both.
- $\left(\bigcup_{\alpha \in \mathcal{I}} A_\alpha\right) \Delta \left(\bigcup_{\alpha \in \mathcal{I}} B_\alpha\right) \subseteq \bigcup_{\alpha \in \mathcal{I}} (A_\alpha \Delta B_\alpha)$, where \mathcal{I} is an arbitrary non-empty index set.

The symmetric difference can be defined in any Boolean algebra, by writing

$$x \Delta y = (x \vee y) \wedge \neg(x \wedge y) = (x \wedge \neg y) \vee (y \wedge \neg x) = x \oplus y.$$

This operation has the same properties as the symmetric difference of sets.

n-ary symmetric difference

As above, the symmetric difference of a collection of sets contains just elements which are in an odd number of the sets in the collection:

$$\Delta M = \left\{ a \in \bigcup M : |\{A \in M : a \in A\}| \text{ is odd} \right\}.$$

Evidently, this is well-defined only when each element of the union $\bigcup M$ is contributed by a finite number of elements of M .

Suppose $M = \{M_1, M_2, \dots, M_n\}$ is a multiset and $n \geq 2$. Then there is a formula for $|\Delta M|$, the number of elements in ΔM , given solely in terms of intersections of elements of M :

$$|\Delta M| = \sum_{l=1}^n (-2)^{l-1} \sum_{i_1 \neq i_2 \neq \dots \neq i_l} |M_{i_1} \cap M_{i_2} \cap \dots \cap M_{i_l}|,$$

where $i_1 \neq i_2 \neq \dots \neq i_l$ is meant to indicate that $\{i_1, i_2, \dots, i_l\}$ is a subset of distinct elements of $\{1, 2, \dots, n\}$, of which there are $\binom{n}{l}$.

Symmetric difference on measure spaces

As long as there is a notion of "how big" a set is, the symmetric difference between two sets can be considered a measure of how "far apart" they are. Formally, if μ is a σ -finite measure defined on a σ -algebra Σ , the function

$$d(X, Y) = \mu(X \Delta Y)$$

is a pseudometric on Σ . d becomes a metric if Σ is considered modulo the equivalence relation $X \sim Y$ if and only if $\mu(X \Delta Y) = 0$. The resulting metric space is separable if and only if $L^2(\mu)$ is separable.

Let $S = (\Omega, \mathcal{A}, \mu)$ be some measure space and let $F, G \in \mathcal{A}$ and $\mathcal{D}, \mathcal{E} \subseteq \mathcal{A}$.

Symmetric difference is measurable: $F \Delta G \in \mathcal{A}$.

We write $F = G$ [\mathcal{A}, μ] iff $\mu(F \Delta G) = 0$. The relation " $=$ " [\mathcal{A}, μ] is an equivalence relation on the \mathcal{A} -measurable sets.

We write $\mathcal{D} \subseteq \mathcal{E}$ [\mathcal{A}, μ] iff to each $D \in \mathcal{D}$ there's some $E \in \mathcal{E}$ such that $D = E$ [\mathcal{A}, μ]. The relation " \subseteq " [\mathcal{A}, μ] is a partial order on the family of subsets of \mathcal{A} .

We write $\mathcal{D} = \mathcal{E}$ [\mathcal{A}, μ] iff $\mathcal{D} \subseteq \mathcal{E}$ [\mathcal{A}, μ] and $\mathcal{E} \subseteq \mathcal{D}$ [\mathcal{A}, μ]. The relation " $=$ " [\mathcal{A}, μ] is an equivalence relationship between the subsets of \mathcal{A} .

The "symmetric closure" of \mathcal{D} is the collection of all \mathcal{A} -measurable sets that are $=$ [\mathcal{A}, μ] to some $D \in \mathcal{D}$.

The symmetric closure of \mathcal{D} contains \mathcal{D} . If \mathcal{D} is a sub- σ -algebra of \mathcal{A} , so is the symmetric closure of \mathcal{D} .

$F = G$ [\mathcal{A}, μ] iff $|\mathbf{1}_F - \mathbf{1}_G| = 0$ [\mathcal{A}, μ]-a.e.

References

- Halmos, Paul R. (1960). *Naive set theory*. The University Series in Undergraduate Mathematics. van Nostrand Company. Zbl 0087.04403 ^[1].
- Symmetric difference ^[2], PlanetMath.org.
- Weisstein, Eric W., "Symmetric Difference ^[3]", *MathWorld*.
- *Symmetric difference of sets* ^[4]. In Encyclopaedia of Mathematics

References

- [1] <http://www.zentralblatt-math.org/zmath/en/search/?format=complete&q=an:0087.04403>
- [2] <http://planetmath.org/?op=getobj&from=objects&id=916>
- [3] <http://mathworld.wolfram.com/SymmetricDifference.html>
- [4] http://www.encyclopediaofmath.org/index.php/Symmetric_difference_of_sets

Tautology (logic)

In logic, a **tautology** (from the Greek word ταυτολογία) is a formula which is true in every possible interpretation. Philosopher Ludwig Wittgenstein first applied the term to redundancies of propositional logic in 1921; it had been used earlier to refer to rhetorical tautologies, and continues to be used in that alternate sense. A formula is satisfiable if it is true under at least one interpretation, and thus a tautology is a formula whose negation is unsatisfiable. Unsatisfiable statements, both through negation and affirmation, are known formally as contradictions. A formula that is neither a tautology nor a contradiction is said to be logically contingent. Such a formula can be made either true or false based on the values assigned to its propositional variables. The double turnstile notation $\models S$ is used to indicate that S is a tautology. Tautology is sometimes symbolized by " $\vee p q$ ", and contradiction by " $\neg p \neg q$ ". The tee symbol \top is sometimes used to denote an arbitrary tautology, with the dual symbol \perp (falsum) representing an arbitrary contradiction.

Tautologies are a key concept in propositional logic, where a tautology is defined as a propositional formula that is true under any possible Boolean valuation of its propositional variables. A key property of tautologies in propositional logic is that an effective method exists for testing whether a given formula is always satisfied (or, equivalently, whether its negation is unsatisfiable).

The definition of *tautology* can be extended to sentences in predicate logic, which may contain quantifiers, unlike sentences of propositional logic. In propositional logic, there is no distinction between a tautology and a logically valid formula. In the context of predicate logic, many authors define a tautology to be a sentence that can be obtained by taking a tautology of propositional logic and uniformly replacing each propositional variable by a first-order formula (one formula per propositional variable). The set of such formulas is a proper subset of the set of logically valid sentences of predicate logic (which are the sentences that are true in every model).

History

The word *tautology* was used by the ancient Greeks to describe a statement that was true merely by virtue of saying the same thing twice, a pejorative meaning that is still used for rhetorical tautologies. Between 1800 and 1940, the word gained new meaning in logic, and is currently used in mathematical logic to denote a certain type of propositional formula, without the pejorative connotations it originally possessed.

In 1800, Immanuel Kant wrote in his book *Logic*:

"The identity of concepts in analytical judgments can be either *explicit* (*explicata*) or *non-explicit* (*implicita*)."

In the former case analytic propositions are *tautological*."

Here *analytic proposition* refers to an analytic truth, a statement in natural language that is true solely because of the terms involved.

In 1884, Gottlob Frege proposed in his *Grundlagen* that a truth is analytic exactly if it can be derived using logic. But he maintained a distinction between analytic truths (those true based only on the meanings of their terms) and tautologies (statements devoid of content).

In 1921, in his *Tractatus Logico-Philosophicus*, Ludwig Wittgenstein proposed that statements that can be deduced by logical deduction are tautological (empty of meaning) as well as being analytic truths. Henri Poincaré had made similar remarks in *Science and Hypothesis* in 1905. Although Bertrand Russell at first argued against these remarks by Wittgenstein and Poincaré, claiming that mathematical truths were not only non-tautologous but were synthetic, he later spoke in favor of them in 1918:

"Everything that is a proposition of logic has got to be in some sense or the other like a tautology. It has got to be something that has some peculiar quality, which I do not know how to define, that belongs to logical propositions but not to others."

Here *logical proposition* refers to a proposition that is provable using the laws of logic.

During the 1930s, the formalization of the semantics of propositional logic in terms of truth assignments was developed. The term *tautology* began to be applied to those propositional formulas that are true regardless of the truth or falsity of their propositional variables. Some early books on logic (such as *Symbolic Logic* by C. I. Lewis and Langford, 1932) used the term for any proposition (in any formal logic) that is universally valid. It is common in presentations after this (such as Stephen Kleene 1967 and Herbert Enderton 2002) to use *tautology* to refer to a logically valid propositional formula, but to maintain a distinction between *tautology* and *logically valid* in the context of first-order logic (see below).

Background

Propositional logic begins with **propositional variables**, atomic units that represent concrete propositions. A **formula** consists of propositional variables connected by logical connectives in a meaningful way, so that the truth of the overall formula can be uniquely deduced from the truth or falsity of each variable. A **valuation** is a function that assigns each propositional variable either T (for truth) or F (for falsity). So, for example, using the propositional variables A and B , the binary connectives \vee and \wedge representing disjunction and conjunction respectively, and the unary connective \neg representing negation, the following formula can be obtained: $(A \wedge B) \vee (\neg A) \vee (\neg B)$. A valuation here must assign to each of A and B either T or F. But no matter how this assignment is made, the overall formula will come out true. For if the first conjunction $(A \wedge B)$ is not satisfied by a particular valuation, then one of A and B is assigned F, which will cause the corresponding later disjunct to be T.

Definition and examples

A formula of propositional logic is a **tautology** if the formula itself is always true regardless of which valuation is used for the propositional variables.

There are infinitely many tautologies. Examples include:

- $(A \vee \neg A)$ ("A or not A"), the law of the excluded middle. This formula has only one propositional variable, A . Any valuation for this formula must, by definition, assign A one of the truth values *true* or *false*, and assign $\neg A$ the other truth value.
- $(A \rightarrow B) \Leftrightarrow (\neg B \rightarrow \neg A)$ ("if A implies B then not- B implies not- A ", and vice versa), which expresses the law of contraposition.
- $((\neg A \rightarrow B) \wedge (\neg A \rightarrow \neg B)) \rightarrow A$ ("if not- A implies both B and its negation not- B , then not- A must be false, then A must be true"), which is the principle known as *reductio ad absurdum*.
- $\neg(A \wedge B) \Leftrightarrow (\neg A \vee \neg B)$ ("if not both A and B , then either not- A or not- B ", and vice versa), which is known as de Morgan's law.
- $((A \rightarrow B) \wedge (B \rightarrow C)) \rightarrow (A \rightarrow C)$ ("if A implies B and B implies C , then A implies C "), which is the principle known as syllogism.
- $((A \vee B) \wedge (A \rightarrow C) \wedge (B \rightarrow C)) \rightarrow C$ (if at least one of A or B is true, and each implies C , then C must be true as well), which is the principle known as proof by cases.

A minimal tautology is a tautology that is not the instance of a shorter tautology.

- $(A \vee B) \rightarrow (A \vee B)$ is a tautology, but not a minimal one, because it is an instantiation of $C \rightarrow C$.

Verifying tautologies

The problem of determining whether a formula is a tautology is fundamental in propositional logic. If there are n variables occurring in a formula then there are 2^n distinct valuations for the formula. Therefore the task of determining whether or not the formula is a tautology is a finite, mechanical one: one need only evaluate the truth value of the formula under each of its possible valuations. One algorithmic method for verifying that every valuation causes this sentence to be true is to make a truth table that includes every possible valuation.

For example, consider the formula

$$((A \wedge B) \rightarrow C) \Leftrightarrow (A \rightarrow (B \rightarrow C)).$$

There are 8 possible valuations for the propositional variables A , B , C , represented by the first three columns of the following table. The remaining columns show the truth of subformulas of the formula above, culminating in a column showing the truth value of the original formula under each valuation.

A	B	C	$A \wedge B$	$(A \wedge B) \rightarrow C$	$B \rightarrow C$	$A \rightarrow (B \rightarrow C)$	$((A \wedge B) \rightarrow C) \Leftrightarrow (A \rightarrow (B \rightarrow C))$
T	T	T	T	T	T	T	T
T	T	F	F	F	F	F	T
T	F	T	F	T	T	T	T
T	F	F	F	T	T	T	T
F	T	T	F	T	T	T	T
F	T	F	F	T	F	T	T
F	F	T	F	T	T	T	T
F	F	F	F	T	T	T	T

Because each row of the final column shows T , the sentence in question is verified to be a tautology.

It is also possible to define a deductive system (proof system) for propositional logic, as a simpler variant of the deductive systems employed for first-order logic (see Kleene 1957, Sec 1.9 for one such system). A proof of a tautology in an appropriate deduction system may be much shorter than a complete truth table (a formula with n propositional variables requires a truth table with 2^n lines, which quickly becomes infeasible as n increases). Proof systems are also required for the study of intuitionistic propositional logic, in which the method of truth tables cannot be employed because the law of the excluded middle is not assumed.

Tautological implication

A formula R is said to **tautologically imply** a formula S if every valuation that causes R to be true also causes S to be true. This situation is denoted $R \models S$. It is equivalent to the formula $R \rightarrow S$ being a tautology (Kleene 1967 p. 27).

For example, let S be $A \wedge (B \vee \neg B)$. Then S is not a tautology, because any valuation that makes A false will make S false. But any valuation that makes A true will make S true, because $B \vee \neg B$ is a tautology. Let R be the formula $A \wedge C$. Then $R \models S$, because any valuation satisfying R makes A true and thus makes S true.

It follows from the definition that if a formula R is a contradiction then R tautologically implies every formula, because there is no truth valuation that causes R to be true and so the definition of tautological implication is trivially satisfied. Similarly, if S is a tautology then S is tautologically implied by every formula.

Substitution

There is a general procedure, the **substitution rule**, that allows additional tautologies to be constructed from a given tautology (Kleene 1967 sec. 3). Suppose that S is a tautology and for each propositional variable A in S a fixed sentence S_A is chosen. Then the sentence obtained by replacing each variable A in S with the corresponding sentence S_A is also a tautology.

For example, let S be $(A \wedge B) \vee (\neg A) \vee (\neg B)$, a tautology. Let S_A be $C \vee D$ and let S_B be $C \rightarrow E$. It follows from the substitution rule that the sentence

$$((C \vee D) \wedge (C \rightarrow E)) \vee (\neg(C \vee D)) \vee (\neg(C \rightarrow E))$$

is a tautology.

Efficient verification and the Boolean satisfiability problem The problem of constructing practical algorithms to determine whether sentences with large numbers of propositional variables are tautologies is an area of contemporary research in the area of automated theorem proving.

The method of truth tables illustrated above is provably correct – the truth table for a tautology will end in a column with only T , while the truth table for a sentence that is not a tautology will contain a row whose final column is F , and the valuation corresponding to that row is a valuation that does not satisfy the sentence being tested. This method for verifying tautologies is an effective procedure, which means that given unlimited computational resources it can always be used to mechanistically determine whether a sentence is a tautology. This means, in particular, the set of tautologies over a fixed finite or countable alphabet is a decidable set.

As an efficient procedure, however, truth tables are constrained by the fact that the number of valuations that must be checked increases as 2^k , where k is the number of variables in the formula. This exponential growth in the computation length renders the truth table method useless for formulas with thousands of propositional variables, as contemporary computing hardware cannot execute the algorithm in a feasible time period.

The problem of determining whether there is any valuation that makes a formula true is the **Boolean satisfiability problem**; the problem of checking tautologies is equivalent to this problem, because verifying that a sentence S is a tautology is equivalent to verifying that there is no valuation satisfying $\neg S$. It is known that the Boolean satisfiability problem is NP complete, and widely believed that there is no polynomial-time algorithm that can

perform it. Current research focuses on finding algorithms that perform well on special classes of formulas, or terminate quickly on average even though some inputs may cause them to take much longer.

Tautologies versus validities in first-order logic

The fundamental definition of a tautology is in the context of propositional logic. The definition can be extended, however, to sentences in first-order logic (see Enderton (2002, p. 114) and Kleene (1967 secs. 17–18)). These sentences may contain quantifiers, unlike sentences of propositional logic. In the context of first-order logic, a distinction is maintained between **logical validities**, sentences that are true in every model, and **tautologies**, which are a proper subset of the first-order logical validities. In the context of propositional logic, these two terms coincide.

A tautology in first-order logic is a sentence that can be obtained by taking a tautology of propositional logic and uniformly replacing each propositional variable by a first-order formula (one formula per propositional variable). For example, because $A \vee \neg A$ is a tautology of propositional logic, $(\forall x(x = x)) \vee (\neg \forall x(x = x))$ is a tautology in first order logic. Similarly, in a first-order language with a unary relation symbols R, S, T , the following sentence is a tautology:

$$(((\exists xRx) \wedge \neg(\exists xSx)) \rightarrow \forall xTx) \Leftrightarrow ((\exists xRx) \rightarrow ((\neg \exists xSx) \rightarrow \forall xTx)).$$

It is obtained by replacing A with $\exists xRx$, B with $\neg \exists xSx$, and C with $\forall xTx$ in the propositional tautology $((A \wedge B) \rightarrow C) \Leftrightarrow (A \rightarrow (B \rightarrow C))$.

Not all logical validities are tautologies in first-order logic. For example, the sentence

$$(\forall xRx) \rightarrow \neg \exists x \neg Rx$$

is true in any first-order interpretation, but it corresponds to the propositional sentence $A \rightarrow B$ which is not a tautology of propositional logic.

References

- Bocheński, J. M. (1959) *Précis of Mathematical Logic*, translated from the French and German editions by Otto Bird, Dordrecht, South Holland: D. Reidel.
- Enderton, H. B. (2002) *A Mathematical Introduction to Logic*, Harcourt/Academic Press, ISBN 0-12-238452-0.
- Kleene, S. C. (1967) *Mathematical Logic*, reprinted 2002, Dover Publications, ISBN 0-486-42533-9.
- Reichenbach, H. (1947). *Elements of Symbolic Logic*, reprinted 1980, Dover, ISBN 0-486-24004-5
- Wittgenstein, L. (1921). "Logisch-philosophische Abhandlung", *Annalen der Naturphilosophie* (Leipzig), v. 14, pp. 185–262, reprinted in English translation as *Tractatus logico-philosophicus*, New York and London, 1922.

External links

- Hazewinkel, Michiel, ed. (2001), "Tautology" ^[1], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Weisstein, Eric W., "Tautology" ^[2], *MathWorld*.

References

- [1] <http://www.encyclopediaofmath.org/index.php?title=p/t092290>
[2] <http://mathworld.wolfram.com/Tautology.html>

Zhegalkin polynomial

Zhegalkin (also **Zegalkin** or **Gegalkine**) **polynomials** form one of many possible representations of the operations of boolean algebra. Introduced by the Russian mathematician I. I. Zhegalkin in 1927, they are the polynomials of ordinary high school algebra interpreted over the integers mod 2. The resulting degeneracies of modular arithmetic result in Zhegalkin polynomials being simpler than ordinary polynomials, requiring neither coefficients nor exponents. Coefficients are redundant because 1 is the only nonzero coefficient. Exponents are redundant because in arithmetic mod 2, $x^2 = x$. Hence a polynomial such as $3x^2y^5z$ is congruent to, and can therefore be rewritten as, xyz .

Boolean equivalent

Prior to 1927 boolean algebra had been considered a calculus of logical values with logical operations of conjunction, disjunction, negation, etc. Zhegalkin showed that all boolean operations could be written as ordinary numeric polynomials, thinking of the logical constants 0 and 1 as integers mod 2. The logical operation of conjunction is realized as the arithmetic operation of multiplication xy , and logical exclusive-or as arithmetic addition mod 2, (written here as $x \oplus y$ to avoid confusion with the common use of $+$ as a synonym for inclusive-or \vee). Logical complement $\neg x$ is then derived from 1 and \oplus as $x \oplus 1$. Since \wedge and \neg form a sufficient basis for the whole of boolean algebra, meaning that all other logical operations are obtainable as composites of these basic operations, it follows that the polynomials of ordinary algebra can represent all boolean operations, allowing boolean reasoning to be performed reliably by appealing to the familiar laws of high school algebra without the distraction of the differences from high school algebra that arise with disjunction in place of addition mod 2.

An example application is the representation of the boolean 2-out-of-3 threshold or median operation as the Zhegalkin polynomial $xy \oplus yz \oplus zx$, which is 1 when at least two of the variables are 1 and 0 otherwise.

Formal properties

Formally a *Zhegalkin monomial* is the product of a finite set of distinct variables (hence square-free), including the empty set whose product is denoted 1. There are 2^n possible Zhegalkin monomials in n variables, since each monomial is fully specified by the presence or absence of each variable. A *Zhegalkin polynomial* is the sum (exclusive-or) of a set of Zhegalkin monomials, with the empty set denoted by 0. A given monomial's presence or absence in a polynomial corresponds to that monomial's coefficient being 1 or 0 respectively. The Zhegalkin monomials, being linearly independent, span a 2^n -dimensional vector space over the Galois field $\text{GF}(2)$ (NB: not $\text{GF}(2^n)$, whose multiplication is quite different). The 2^n vectors of this space, i.e. the linear combinations of those monomials as unit vectors, constitute the Zhegalkin polynomials. The exact agreement with the number of boolean operations on n variables, which exhaust the n -ary operations on $\{0,1\}$, furnishes a direct counting argument for completeness of the Zhegalkin polynomials as a boolean basis.

This vector space is not equivalent to the free boolean algebra on n generators because it lacks complementation (bitwise logical negation) as an operation (equivalently, because it lacks the top element as a constant). This is not to say that the space is not closed under complementation or lacks top (the all-ones vector) as an element, but rather that the linear transformations of this and similarly constructed spaces need not preserve complement and top. Those that do preserve them correspond to the boolean homomorphisms, e.g. there are four linear transformations from the vector space of Zhegalkin polynomials over one variable to that over none, only two of which are boolean homomorphisms.

Related work

In the same year as Zhegalkin's paper (1927) the American mathematician E.T. Bell published a sophisticated arithmetization of boolean algebra based on Dedekind's ideal theory and general modular arithmetic (as opposed to arithmetic mod 2). The much simpler arithmetic character of Zhegalkin polynomials was first noticed in the west (independently, communication between Soviet and western mathematicians being very limited in that era) by the American mathematician Marshall Stone in 1936 when he observed while writing up his celebrated Stone duality theorem that the supposedly loose analogy between boolean algebras and rings could in fact be formulated as an exact equivalence holding for both finite and infinite algebras, leading him to substantially reorganize his paper.

References

- Bell, Eric (1927). "Arithmetic of Logic". *Transactions of the American Mathematical Society* (Transactions of the American Mathematical Society, Vol. 29, No. 3) **29** (3): 597–611. doi:10.2307/1989098^[1]. JSTOR 1989098^[2].
- Gindikin, S.G. (1972). *Algebraic Logic (Russian: алгебра логики в задачах)*. Moscow: Nauka (English translation Springer-Verlag 1985). ISBN 0-387-96179-8.
- Stone, Marshall (1936). "The Theory of Representations for Boolean Algebras". *Transactions of the American Mathematical Society* (Transactions of the American Mathematical Society, Vol. 40, No. 1) **40** (1): 37–111. doi:10.2307/1989664^[8]. ISSN 0002-9947^[9]. JSTOR 1989664^[10].
- Zhegalkin, Ivan Ivanovich (1927). "On the Technique of Calculating Propositions in Symbolic Logic". *Matematicheskii Sbornik* **43**: 9–28.

References

[1] <http://dx.doi.org/10.2307%2F1989098>

[2] <http://www.jstor.org/stable/1989098>

Syntax

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]

$$\begin{aligned} & (\textcolor{red}{1} \oplus \textcolor{blue}{x})(\textcolor{green}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y}) \\ & \textcolor{red}{1}(\textcolor{green}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y}) \oplus \textcolor{red}{x}(\textcolor{green}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y}) \\ & (\textcolor{red}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y}) \oplus (\textcolor{blue}{x} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{xy}) \\ & \textcolor{red}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y} \oplus \textcolor{blue}{xy} \\ & \textcolor{red}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y} \oplus \textcolor{blue}{xy} \end{aligned}$$

OR (logical disjunction) uses either $\textcolor{red}{1} \oplus (\textcolor{red}{1} \oplus \textcolor{blue}{a})(\textcolor{red}{1} \oplus \textcolor{blue}{b})$ ^[3] (easier when both operands have purely true terms) or $\textcolor{red}{a} \oplus \textcolor{blue}{b} \oplus \textcolor{blue}{ab}$ ^[4] (easier otherwise):

$$\begin{aligned} & (\textcolor{red}{1} \oplus \textcolor{blue}{x}) + (\textcolor{green}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y}) \\ & \textcolor{red}{1} \oplus (\textcolor{red}{1} \oplus \textcolor{red}{1} \oplus \textcolor{blue}{x})(\textcolor{red}{1} \oplus \textcolor{green}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{y}) \\ & \textcolor{red}{1} \oplus \textcolor{blue}{x}(\textcolor{blue}{x} \oplus \textcolor{blue}{y}) \\ & \textcolor{red}{1} \oplus \textcolor{blue}{x} \oplus \textcolor{blue}{xy} \end{aligned}$$

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:

$$\begin{aligned} & \textcolor{blue}{x} + (\textcolor{blue}{y} \cdot \neg \textcolor{blue}{z}) \\ & \textcolor{blue}{x} + (\textcolor{blue}{y}(\textcolor{red}{1} \oplus \textcolor{blue}{z})) \\ & \textcolor{blue}{x} + (\textcolor{blue}{y} \oplus \textcolor{blue}{yz}) \\ & \textcolor{blue}{x} \oplus (\textcolor{blue}{y} \oplus \textcolor{blue}{yz}) \oplus \textcolor{blue}{x}(\textcolor{blue}{y} \oplus \textcolor{blue}{yz}) \\ & \textcolor{blue}{x} \oplus \textcolor{blue}{y} \oplus \textcolor{blue}{xy} \oplus \textcolor{blue}{yz} \oplus \textcolor{blue}{xyz} \end{aligned}$$

Formal representation

ANF is sometimes described in an equivalent way:

$$\begin{aligned} f(x_1, x_2, \dots, x_n) = & \textcolor{blue}{a}_0 \oplus \\ & \textcolor{blue}{a}_1 x_1 \oplus \textcolor{blue}{a}_2 x_2 \oplus \dots \oplus \textcolor{blue}{a}_n x_n \oplus \\ & \textcolor{blue}{a}_{1,2} x_1 x_2 \oplus \dots \oplus \textcolor{blue}{a}_{n-1,n} x_{n-1} x_n \oplus \\ & \dots \oplus \\ & \textcolor{blue}{a}_{1,2,\dots,n} x_1 x_2 \dots x_n \end{aligned}$$

where $\textcolor{blue}{a}_0, \textcolor{blue}{a}_1, \dots, \textcolor{blue}{a}_{1,2,\dots,n} \in \{0, 1\}^*$ 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:

$$f(x_1, x_2, \dots, x_n) = g(x_2, \dots, x_n) \oplus x_1 h(x_2, \dots, x_n), \text{ where}$$

$$\bullet \quad g(x_2, \dots, x_n) = f(0, x_2, \dots, x_n)$$

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

Indeed,

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

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: $\neg a = 1 \oplus a$ (<http://www.wolframalpha.com/input/?i=simplify+1+xor+a>)
- [2] WolframAlpha AND-equivalence demonstration: $(a \oplus b)(c \oplus d) = ac \oplus ad \oplus bc \oplus bd$ ([http://www.wolframalpha.com/input/?i=\(a+xor+b\)+and+\(c+xor+d\)+in+anf](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: $a + b = a \oplus b \oplus ab$ ([http://www.wolframalpha.com/input/?i=simplify+a+xor+b+xor+\(a+and+b\)](http://www.wolframalpha.com/input/?i=simplify+a+xor+b+xor+(a+and+b)))

Boolean conjunctive query

In the theory of relational databases, a **Boolean conjunctive query** is a conjunctive query without distinguished predicates, i.e., a query in the form $R_1(t_1) \wedge \dots \wedge R_n(t_n)$, where each R_i is a relation symbol and each t_i is a tuple of variables and constants; the number of elements in t_i is equal to the arity of R_i . Such a query evaluates to either true or false depending on whether the relations in the database contains the appropriate tuples of values.

As an example, if a database schema contains the relation symbols *Father* (binary, who's the father of whom) and *Employed* (unary, who is employed), a conjunctive query could be $Father(Mark, x) \wedge Employed(x)$.

This query evaluates to true if there exists an individual x who is a child of Mark and employed. In other words, this query expresses the question: "does Mark have employed children?"

References

- G. Gottlob, N. Leone, F. Scarcello (2001). "The complexity of acyclic conjunctive queries". *Journal of the ACM (JACM)* **48** (3): 431–498. doi:10.1145/382780.382783^[1].

References

- [1] <http://dx.doi.org/10.1145%2F382780.382783>

Canonical form (Boolean algebra)

In Boolean algebra, any Boolean function can be put into the **canonical disjunctive normal form (CDNF)** or **minterm canonical form** and its dual **canonical conjunctive normal form (CCNF)** or **maxterm canonical form**. Other canonical forms include the complete sum of prime implicants or Blake canonical form (and its dual), and the algebraic normal form (also called Zhegalkin or Reed–Muller).

Minterms are called products because they are the logical AND of a set of variables, and *maxterms* are called sums because they are the logical OR of a set of variables. These concepts are dual because of their complementary-symmetry relationship as expressed by De Morgan's laws.

Two dual canonical forms of *any* Boolean function are a "sum of minterms" and a "product of maxterms." The term "**Sum of Products**" or "**SoP**" is widely used for the canonical form that is a disjunction (OR) of minterms. Its De Morgan dual is a "**Product of Sums**" or "**PoS**" for the canonical form that is a conjunction (AND) of maxterms. These forms allow for greater analysis into the simplification of these functions, which is of great importance in the minimization or other optimization of Boolean formulas in general and digital circuits in particular.

Summary

One application of Boolean algebra is digital circuit design. The goal may be to minimize the number of gates, to minimize the settling time, etc.

There are sixteen possible functions of two variables, but in digital logic hardware, the simplest gate circuits implement only four of them: *conjunction* (AND), *disjunction* (inclusive OR), and the complements of those (NAND and NOR).

Most gate circuits accept more than 2 input variables; for example, the spaceborne Apollo Guidance Computer, which pioneered the application of integrated circuits in the 1960s, was built with only one type of gate, a 3-input NOR, whose output is true only when all 3 inputs are false.^[1]

Minterms

For a boolean function of n variables x_1, \dots, x_n , a product term in which each of the n variables appears **once** (in either its complemented or uncomplemented form) is called a *minterm*. Thus, a *minterm* is a logical expression of n variables that employs only the *complement* operator and the *conjunction* operator.

For example, abc , $ab'c$ and abc' are 3 examples of the 8 minterms for a Boolean function of the three variables a , b , and c . The customary reading of the last of these is *a AND b AND NOT-c*.

There are 2^n minterms of n variables, since a variable in the minterm expression can be in either its direct or its complemented form—two choices per n variables.

Indexing minterms

Minterms are often numbered by a binary encoding of the complementation pattern of the variables, where the variables are written in a standard order, usually alphabetical. This convention assigns the value 1 to the direct form (x_i) and 0 to the complemented form (x'_i); the minterm is then the sum of $2^i \text{value}(x_i)$. For example, minterm abc' is numbered $110_2 = 6_{10}$ and denoted m_6 .

Functional equivalence

A given minterm n gives a true value (i.e., 1) for just one combination of the input variables. For example, minterm 5, $a' b' c$, is true only when a and c both are true and b is false—the input arrangement where $a = 1, b = 0, c = 1$ results in 1.

Given the truth table of a logical function, it is possible to write the function as a "sum of products". This is a special form of disjunctive normal form. For example, if given the truth table for the arithmetic sum bit u of one bit position's logic of an adder circuit, as a function of x and y from the addends and the carry in, ci :

ci	x	y	$u(ci,x,y)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Observing that the rows that have an output of 1 are the 2nd, 3rd, 5th, and 8th, we can write u as a sum of minterms m_1, m_2, m_4 , and m_7 . If we wish to verify this:

$u(ci, x, y) = m_1 + m_2 + m_4 + m_7 = (ci', x', y) + (ci', x, y') + (ci, x', y') + (ci, x, y)$ evaluated for all 8 combinations of the three variables will match the table.

Maxterms

For a boolean function of n variables x_1, \dots, x_n , a sum term in which each of the n variables appears **once** (in either its complemented or uncomplemented form) is called a *maxterm*. Thus, a *maxterm* is a logical expression of n variables that employs only the *complement* operator and the *disjunction* operator. Maxterms are a dual of the minterm idea (i.e., exhibiting a complementary symmetry in all respects). Instead of using ANDs and complements, we use ORs and complements and proceed similarly.

For example, the following are two of the eight maxterms of three variables:

$$a + b' + c$$

$$a' + b + c$$

There are again 2^n maxterms of n variables, since a variable in the maxterm expression can also be in either its direct or its complemented form—two choices per n variables.

Indexing maxterms

Each maxterm is assigned an index based on the opposite conventional binary encoding used for minterms. The maxterm convention assigns the value 0 to the direct form (x_i) and 1 to the complemented form (x'_i). For example, we assign the index 6 to the maxterm $a' + b' + c$ (110) and denote that maxterm as M_6 . Similarly M_0 of these three variables is $a + b + c$ (000) and M_7 is $a' + b' + c'$ (111).

Functional equivalence

It is apparent that maxterm n gives a *false* value (i.e., 0) for just one combination of the input variables. For example, maxterm 5, $a' + b + c'$, is false only when a and c both are true and b is false—the input arrangement where $a = 1$, $b = 0$, $c = 1$ results in 0.

If one is given a truth table of a logical function, it is possible to write the function as a "product of sums". This is a special form of conjunctive normal form. For example, if given the truth table for the carry-out bit co of one bit position's logic of an adder circuit, as a function of x and y from the addends and the carry in, ci :

ci	x	y	$co(ci,x,y)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Observing that the rows that have an output of 0 are the 1st, 2nd, 3rd, and 5th, we can write co as a product of maxterms M_0 , M_1 , M_2 , and M_4 . If we wish to verify this: $co(ci, x, y) = M_0M_1M_2M_4 = (ci + x + y)(ci + x + y')(ci + x' + y)(ci' + x + y)$ evaluated for all 8 combinations of the three variables will match the table.

Dualization

The complement of a minterm is the respective maxterm. This can be easily verified by using de Morgan's law. For example: $M_5 = a' + b + c' = (ab'c)' = m'_5$

Non-canonical PoS and SoP forms

It is often the case that the canonical minterm form can be simplified to an equivalent SoP form. This simplified form would still consist of a sum of product terms. However, in the simplified form, it is possible to have fewer product terms and/or product terms that contain fewer variables. For example, the following 3-variable function:

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

has the canonical minterm representation: $f = a'b'c + abc$, but it has an equivalent simplified form: $f = bc$. In this trivial example, it is obvious that $bc = a'b'c + abc$, but the simplified form has both fewer product terms, and the term has fewer variables. The most simplified SoP representation of a function is referred to as a *minimal SoP form*.

In a similar manner, a canonical maxterm form can have a simplified PoS form.

While this example was easily simplified by applying normal algebraic methods [$f = (a' + a)bc$], in less obvious cases a convenient method for finding the minimal PoS/SoP form of a function with up to four variables is using a Karnaugh map.

The minimal PoS and SoP forms are very important for finding optimal implementations of boolean functions and minimizing logic circuits.

Application example

The sample truth tables for minterms and maxterms above are sufficient to establish the canonical form for a single bit position in the addition of binary numbers, but are not sufficient to design the digital logic unless your inventory of gates includes AND and OR. Where performance is an issue (as in the Apollo Guidance Computer), the available parts are more likely to be NAND and NOR because of the complementing action inherent in transistor logic. The values are defined as voltage states, one near ground and one near the DC supply voltage V_{cc} , e.g. +5 VDC. If the higher voltage is defined as the 1 "true" value, a NOR gate is the simplest possible useful logical element.

Specifically, a 3-input NOR gate may consist of 3 bipolar junction transistors with their emitters all grounded, their collectors tied together and linked to V_{cc} through a load impedance. Each base is connected to an input signal, and the common collector point presents the output signal. Any input that is a 1 (high voltage) to its base shorts its transistor's emitter to its collector, causing current to flow through the load impedance, which brings the collector voltage (the output) very near to ground. That result is independent of the other inputs. Only when all 3 input signals are 0 (low voltage) do the emitter-collector impedances of all 3 transistors remain very high. Then very little current flows, and the voltage-divider effect with the load impedance imposes on the collector point a high voltage very near to V_{cc} .

The complementing property of these gate circuits may seem like a drawback when trying to implement a function in canonical form, but there is a compensating bonus: such a gate with only one input implements the complementing function, which is required frequently in digital logic.

This example assumes the Apollo parts inventory: 3-input NOR gates only, but the discussion is simplified by supposing that 4-input NOR gates are also available (in Apollo, those were compounded out of pairs of 3-input NORs).

Canonical and non-canonical consequences of NOR gates

Fact #1: a set of 8 NOR gates, if their inputs are all combinations of the direct and complement forms of the 3 input variables ci , x , and y , always produce minterms, never maxterms—that is, of the 8 gates required to process all combinations of 3 input variables, only one has the output value 1. That's because a NOR gate, despite its name, could better be viewed (using De Morgan's law) as the AND of the complements of its input signals.

Fact #2: the reason Fact #1 is not a problem is the duality of minterms and maxterms, i.e. each maxterm is the complement of the like-indexed minterm, and vice versa.

In the minterm example above, we wrote $u(ci, x, y) = m_1 + m_2 + m_4 + m_7$ but to perform this with a 4-input NOR gate we need to restate it as a product of sums (PoS), where the sums are the opposite maxterms. That is,

$u(ci, x, y) = \text{AND}(M_0, M_3, M_5, M_6) = \text{NOR}(m_0, m_3, m_5, m_6)$. Truth tables:

ci	x	y	M ₀	M ₃	M ₅	M ₆	AND	u(ci,x,y)
0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	1	0	1	0	0
1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1

ci	x	y	m ₀	m ₃	m ₅	m ₆	NOR	u(ci,x,y)
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	1	1
0	1	1	0	1	0	0	0	0
1	0	0	0	0	0	0	1	1
1	0	1	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	0	1	1

In the maxterm example above, we wrote $co(ci, x, y) = M_0 M_1 M_2 M_4$ but to perform this with a 4-input NOR gate we need to notice the equality to the NOR of the same minterms. That is,

$co(ci, x, y) = \text{AND}(M_0, M_1, M_2, M_4) = \text{NOR}(m_0, m_1, m_2, m_4)$. Truth tables:

ci	x	y	M₀	M₁	M₂	M₄	AND	co(ci,x,y)
0	0	0	0	1	1	1	0	0
0	0	1	1	0	1	1	0	0
0	1	0	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	0	0	0
1	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

ci	x	y	m₀	m₁	m₂	m₄	NOR	co(ci,x,y)
0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0
0	1	1	0	0	0	0	1	1
1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	1	0	0	0	0	1	1

Design trade-offs considered in addition to canonical forms

One might suppose that the work of designing an adder stage is now complete, but we haven't addressed the fact that all 3 of the input variables have to appear in both their direct and complement forms. There's no difficulty about the addends x and y in this respect, because they are static throughout the addition and thus are normally held in latch circuits that routinely have both direct and complement outputs. (The simplest latch circuit made of NOR gates is a pair of gates cross-coupled to make a flip-flop: the output of each is wired as one of the inputs to the other.) There is also no need to create the complement form of the sum u . However, the carry out of one bit position must be passed as the carry into the next bit position in both direct and complement forms. The most straightforward way to do this is to pass co through a 1-input NOR gate and label the output co' , but that would add a gate delay in the worst possible place, slowing down the rippling of carries from right to left. An additional 4-input NOR gate building the canonical form of co' (out of the opposite minterms as co) solves this problem.

$$co'(ci, x, y) = \text{AND}(M_3, M_5, M_6, M_7) = \text{NOR}(m_3, m_5, m_6, m_7).$$

Truth tables:

ci	x	y	M_3	M_5	M_6	M_7	AND	$co'(ci,x,y)$
0	0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	0	0
1	1	0	1	1	0	1	0	0
1	1	1	1	1	1	0	0	0

ci	x	y	m_3	m_5	m_6	m_7	NOR	$co'(ci,x,y)$
0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	1	0	0

The trade-off to maintain full speed in this way includes an unexpected cost (in addition to having to use a bigger gate). If we'd just used that 1-input gate to complement co , there would have been no use for the minterm m_7 , and the gate that generated it could have been eliminated. Nevertheless, it's still a good trade.

Now we could have implemented those functions exactly according to their SoP and PoS canonical forms, by turning NOR gates into the functions specified. A NOR gate is made into an OR gate by passing its output through a 1-input NOR gate; and it is made into an AND gate by passing each of its inputs through a 1-input NOR gate. However, this approach not only increases the number of gates used, but also doubles the number of gate delays processing the signals, cutting the processing speed in half. Consequently, whenever performance is vital, going beyond canonical forms and doing the Boolean algebra to make the unenhanced NOR gates do the job is well worthwhile.

Top-down vs. bottom-up design

We have now seen how the minterm/maxterm tools can be used to design an adder stage in canonical form with the addition of some Boolean algebra, costing just 2 gate delays for each of the outputs. That's the "top-down" way to design the digital circuit for this function, but is it the best way? The discussion has focused on identifying "fastest" as "best," and the augmented canonical form meets that criterion flawlessly, but sometimes other factors predominate. The designer may have a primary goal of minimizing the number of gates, and/or of minimizing the fanouts of signals to other gates since big fanouts reduce resilience to a degraded power supply or other environmental factors. In such a case, a designer may develop the canonical-form design as a baseline, then try a bottom-up development, and finally compare the results.

The bottom-up development involves noticing that $u = ci \text{ XOR } (x \text{ XOR } y)$, where XOR means eXclusive OR [true when either input is true but not when both are true], and that $co = ci \cdot x + x \cdot y + y \cdot ci$. One such development takes twelve NOR gates in all: six 2-input gates and two 1-input gates to produce u in 5 gate delays, plus three 2-input gates and one 3-input gate to produce co' in 2 gate delays. The canonical baseline took eight 3-input NOR gates plus

three 4-input NOR gates to produce u , co and co' in 2 gate delays. If the circuit inventory actually includes 4-input NOR gates, the top-down canonical design looks like a winner in both gate count and speed. But if (contrary to our convenient supposition) the circuits are actually 3-input NOR gates, of which two are required for each 4-input NOR function, then the canonical design takes 14 gates compared to 12 for the bottom-up approach, but still produces the sum digit u considerably faster. The fanout comparison is tabulated as:

Variables	Top-down	Bottom-up
x	4	1
x'	4	3
y	4	1
y'	4	3
ci	4	1
ci'	4	3
M or m	4@1,4@2	N/A
x XOR y	N/A	2
Misc	N/A	5@1
Max	4	3

What's a decision-maker to do? An observant one will have noticed that the description of the bottom-up development mentions co' as an output but not co . Does that design simply never need the direct form of the carry out? Well, yes and no. At each stage, the calculation of co' depends only on ci' , x' and y' , which means that the carry propagation ripples along the bit positions just as fast as in the canonical design without ever developing co . The calculation of u , which does require ci to be made from ci' by a 1-input NOR, is slower but for any word length the design only pays that penalty once (when the leftmost sum digit is developed). That's because those calculations overlap, each in what amounts to its own little pipeline without affecting when the next bit position's sum bit can be calculated. And, to be sure, the co' out of the leftmost bit position will probably have to be complemented as part of the logic determining whether the addition overflowed. But using 3-input NOR gates, the bottom-up design is very nearly as fast for doing parallel addition on a non-trivial word length, cuts down on the gate count, and uses lower fanouts ... so it wins if gate count and/or fanout are paramount!

We'll leave the exact circuitry of the bottom-up design of which all these statements are true as an exercise for the interested reader, assisted by one more algebraic formula: $u = ci(x \text{ XOR } y) + ci'(x \text{ XOR } y)'J$. Decoupling the carry propagation from the sum formation in this way is what elevates the performance of a *carry-lookahead adder* over that of a *ripple carry adder*.

To see how NOR gate logic was used in the Apollo Guidance Computer's ALU, visit http://klabs.org/history/ech/agc_schematics/index.htm, select any of the 4-BIT MODULE entries in the Index to Drawings, and expand images as desired.

Footnotes

[1] Eldon C. Hall, Journey to the Moon: The History of the Apollo Guidance Computer, AIAA 1996. ISBN 1-56347-185-X

References

- Edward A. Bender, S. Gill Williamson, 2005, *A Short Course in Discrete Mathematics*, Dover Publications, Inc., Mineola, NY, ISBN 0-486-43946-1. The authors demonstrate a proof that any Boolean (logic) function can be expressed in either disjunctive or conjunctive normal form (cf pages 5–6); the proof simply proceeds by creating all 2^N rows of N Boolean variables and demonstrates that each row ("minterm" or "maxterm") has a unique

Boolean expression. Any Boolean function of the N variables can be derived from a composite of the rows whose minterm or maxterm are logical 1s ("trues").

- E. J. McCluskey, 1965, *Introduction to the Theory of Switching Circuits*, McGraw–Hill Book Company, NY, Library of Congress Catalog Card Number 65-17394. Canonical expressions are defined and described on pages 78ff.
- Fredrick J. Hill, and Gerald R. Peterson, 1974, *Introduction to Switching Theory and Logical Design, Second Edition*, John Wiley & Sons, NY, ISBN 0-471-39882-9. Minterm and maxterm designation of functions appears on page 101ff.

External links

- George Boole, 1848, "The Calculus of Logic, (<http://www.maths.tcd.ie/pub/HistMath/People/Boole/CalcLogic/CalcLogic.html>)" *Cambridge and Dublin Mathematical Journal III: 183–98*.

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**:

$$\begin{aligned} &\neg A \wedge (B \vee C) \\ &(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E) \\ &A \vee B \\ &A \wedge B \end{aligned}$$

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**:

$$\begin{aligned} &\neg(B \vee C) \\ &(A \wedge B) \vee C \\ &A \wedge (B \vee (D \wedge E)). \end{aligned}$$

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

$$\begin{aligned} &\neg B \wedge \neg C \\ &(A \vee C) \wedge (B \vee C) \\ &A \wedge (B \vee D) \wedge (B \vee E). \end{aligned}$$

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:

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

In particular, the generated formula is:

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

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, \dots, Z_n as follows:

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

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 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

$$(l_{11} \vee \dots \vee l_{1n_1}) \wedge \dots \wedge (l_{m1} \vee \dots \vee l_{mn_m}), \text{ with } l_{ij} \text{ literals, is commonly represented as a set of sets}$$

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

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 $k \geq 3$) by replacing each conjunct with more than k variables $X_1 \vee \dots \vee X_k \vee \dots \vee X_n$ by two conjuncts $X_1 \vee \dots \vee X_{k-1} \vee Z$ and $\neg Z \vee X_k \dots \vee X_n$ 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 $\neg P \vee Q$; replace $P \leftrightarrow Q$ with $(P \vee \neg Q) \wedge (\neg P \vee Q)$. Eventually, this will eliminate all occurrences of \rightarrow and \leftrightarrow .
2. Move NOTs inwards by repeatedly applying De Morgan's Law. Specifically, replace $\neg(P \vee Q)$ with $(\neg P) \wedge (\neg Q)$; replace $\neg(P \wedge Q)$ with $(\neg P) \vee (\neg Q)$; 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 $(\forall x P(x)) \vee (\exists x Q(x))$ 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, $\forall x [\exists y Animal(y) \wedge \neg Loves(x, y)] \vee [\exists y Loves(y, x)]$ is renamed to $\forall x [\exists y Animal(y) \wedge \neg Loves(x, y)] \vee \exists z Loves(z, x)$.

3. Skolemize the statement
 1. Move quantifiers outwards: repeatedly replace $P \wedge (\forall x Q(x))$ with $\forall x (P \wedge Q(x))$; replace $P \vee (\forall x Q(x))$ with $\forall x (P \vee Q(x))$; replace $P \wedge (\exists x Q(x))$ with $\exists x (P \wedge Q(x))$; replace $P \vee (\exists x Q(x))$ with $\exists x (P \vee Q(x))$. 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 , \wedge or \vee .
 2. Repeatedly replace $\forall x_1 \dots \forall x_n \exists y P(y)$ with $\forall x_1 \dots \forall x_n P(f(x_1, \dots, x_n))$, 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 $P \vee (Q \wedge R)$ with $(P \vee Q) \wedge (P \vee R)$.

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 ($\exists y$ $Loves(y, x)$)
($\forall y$ $\neg Animal(y) \vee Loves(x, y)$) \rightarrow ($\exists y$ $Loves(y, x)$)
($\forall y$ $\neg Animal(y) \vee Loves(x, y)$) \vee ($\exists y$ $Loves(y, x)$)
($\exists y \neg (\neg Animal(y) \vee Loves(x, y)) \vee (\exists y Loves(y, x))$
($\exists y \neg \neg Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y Loves(y, x))$
($\exists y Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y Loves(y, x))$
($\exists y Animal(y) \wedge \neg Loves(x, y)) \vee (\exists y Loves(z, x))$
($\exists y Animal(y) \wedge \neg Loves(x, y)) \vee Loves(z, x)$
 $\exists y (Animal(y) \wedge \neg Loves(x, y)) \vee Loves(z, x)$
 $\exists y (Animal(y) \wedge \neg Loves(x, y)) \vee Loves(g(x), x)$
($Animal(f(x)) \wedge \neg Loves(x, f(x)) \vee Loves(g(x), x)$
 $Animal(f(x)) \vee Loves(g(x), x) \wedge (\neg Loves(x, f(x)) \vee Loves(g(x), x))$
 $Animal(f(x)) , Loves(g(x), x) \} , \{ \neg Loves(x, f(x)) , Loves(g(x), x)$

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 below 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:

$$\begin{aligned} & A \wedge B \\ & A \\ & (A \wedge B) \vee C \\ & (A \wedge \neg B \wedge \neg C) \vee (\neg D \wedge E \wedge F) \end{aligned}$$

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

$$\begin{aligned} & \neg(A \vee B) — \text{NOT is the outermost operator} \\ & A \vee (B \wedge (C \vee D)) — \text{an OR is nested within an AND} \end{aligned}$$

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:

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

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* \rightarrow *conjunct*
2. *disjunct* \rightarrow *disjunct* \vee *conjunct*
3. *conjunct* \rightarrow *literal*
4. *conjunct* \rightarrow (*conjunct* \wedge *literal*)
5. *literal* \rightarrow *variable*
6. *literal* \rightarrow \neg *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>

Formal system

A **formal system** is broadly defined as any well-defined system of abstract thought based on the model of mathematics. Euclid's *Elements* is often held to be the first formal system and displays the characteristic of a formal system. The entailment of the system by its logical foundation is what distinguishes a formal system from others which may have some basis in an abstract model. Often the formal system will be the basis for or even identified with a larger theory or field (e.g. Euclidean geometry) consistent with the usage in modern mathematics such as model theory. A formal system need not be mathematical as such, Spinoza's Ethics for example imitates the form of Euclid's Elements.

Each formal system has a formal language, which is composed by primitive symbols. These symbols act on certain rules of formation and are developed by inference from a set of axioms. The system thus consists of any number of formulas built up through finite combinations of the primitive symbols—combinations that are formed from the axioms in accordance with the stated rules.^[1]

Formal systems in mathematics consist of the following elements:

1. A finite set of symbols (i.e. the alphabet), that can be used for constructing formulas (i.e. finite strings of symbols).
2. A grammar, which tells how well-formed formulas (abbreviated *wff*) are constructed out of the symbols in the alphabet. It is usually required that there be a decision procedure for deciding whether a formula is well formed or not.
3. A set of axioms or axiom schemata: each axiom must be a wff.
4. A set of inference rules.

A formal system is said to be recursive (i.e. effective) if the set of axioms and the set of inference rules are decidable sets or semidecidable sets, according to context.

Some theorists use the term *formalism* as a rough synonym for *formal system*, but the term is also used to refer to a particular style of *notation*, for example, Paul Dirac's bra-ket notation.

Related subjects

Logical system

A *logical system* or, for short, *logic*, is a formal system together with a form of semantics, usually in the form of model-theoretic interpretation, which assigns truth values to sentences of the formal language, that is, formulae that contain no free variables. A logic is sound if all sentences that can be derived are true in the interpretation, and complete if, conversely, all true sentences can be derived.

Formal proofs

Formal proofs are sequences of wffs. For a wff to qualify as part of a proof, it might either be an axiom or be the product of applying an inference rule on previous wffs in the proof sequence. The last wff in the sequence is recognized as a theorem.

The point of view that generating formal proofs is all there is to mathematics is often called *formalism*. David Hilbert founded metamathematics as a discipline for discussing formal systems. Any language that one uses to talk about a formal system is called a *metalanguage*. The metalanguage may be nothing more than ordinary natural language, or it may be partially formalized itself, but it is generally less completely formalized than the formal language component of the formal system under examination, which is then called the *object language*, that is, the object of the discussion in question.

Once a formal system is given, one can define the set of theorems which can be proved inside the formal system. This set consists of all wffs for which there is a proof. Thus all axioms are considered theorems. Unlike the grammar for wffs, there is no guarantee that there will be a decision procedure for deciding whether a given wff is a theorem or not. The notion of *theorem* just defined should not be confused with *theorems about the formal system*, which, in order to avoid confusion, are usually called metatheorems.

Formal language

In mathematics, logic, and computer science, a formal language is a language that is defined by precise mathematical or machine processable formulas. Like languages in linguistics, formal languages generally have two aspects:

- the syntax of a language is what the language looks like (more formally: the set of possible expressions that are valid utterances in the language)
- the semantics of a language are what the utterances of the language mean (which is formalized in various ways, depending on the type of language in question)

A special branch of mathematics and computer science exists that is devoted exclusively to the theory of language syntax: formal language theory. In formal language theory, a language is nothing more than its syntax; questions of semantics are not included in this specialty.

Formal grammar

In computer science and linguistics a formal grammar is a precise description of a formal language: a set of strings. The two main categories of formal grammar are that of generative grammars, which are sets of rules for how strings in a language can be generated, and that of analytic grammars, which are sets of rules for how a string can be analyzed to determine whether it is a member of the language. In short, an analytic grammar describes how to *recognize* when strings are members in the set, whereas a generative grammar describes how to *write* only those strings in the set.

References

[1] Encyclopædia Britannica, Formal system (<http://www.britannica.com/eb/article-9034889/formal-system>) definition, 2007.

Further reading

- Raymond M. Smullyan, 1961. *Theory of Formal Systems: Annals of Mathematics Studies*, Princeton University Press (April 1, 1961) 156 pages ISBN 0-691-08047-X
- S. C. Kleene, 1967. *Mathematical Logic* Reprinted by Dover, 2002. ISBN 0-486-42533-9
- Douglas Hofstadter, 1979., *Gödel, Escher, Bach: An Eternal Golden Braid* ISBN 978-0-465-02656-2. 777 pages.

External links

- Encyclopædia Britannica, Formal system (<http://www.britannica.com/eb/article-9034889/formal-system>) definition, 2007.
- Christer Blomqvist, an introduction to formal systems (<http://hemsidor.torget.se/users/m/mauritz/math/logic/inform.htm>), webpage 1997.
- What is a Formal System? (<http://www.cs.indiana.edu/~port/teach/641/formal.sys.haug.html>): Some quotes from John Haugeland's 'Artificial Intelligence: The Very Idea' (1985), pp. 48–64.
- Heinrich Herre Formal Language and systems (<http://www-ls.informatik.uni-tuebingen.de/psh/forschung/publikationen/RoutledgeFLS1995.pdf>), 1997.
- Peter Suber, Formal Systems and Machines: An Isomorphism (<http://www.earlham.edu/~peters/courses/logsyst/machines.htm>), 1997.

Normal Forms

Blake canonical form

In Boolean logic, a formula for a Boolean function f is in **Blake canonical form**, also called a **complete sum (of prime implicants)**,^[1] if it is a disjunction of all the prime implicants of f .^[2] Blake canonical form is a disjunctive normal form. It is not necessarily minimal.

It was introduced in 1937 by Archie Blake, who called it the "simplified canonical form";^[3] it was named in honor of Blake by Frank Markham Brown in 1990.

Blake discussed three methods for calculating the canonical form: exhaustion of implicants, iterated consensus, and multiplication. The iterated consensus method was rediscovered by Samson and Mills, Quine, and Bing.

Notes

- [1] Tsutomu Sasao, "Ternary Decision Diagrams and their Applications", in Tsutomu Sasao, Masahira Fujita, eds., *Representations of Discrete Functions* ISBN 0792397207, 1996, p. 278
- [2] Frank Markham Brown, "The Blake Canonical Form", chapter 4 of *Boolean Reasoning: The Logic of Boolean Equations*, ISBN 0486427854, 2nd edition, 2012, p. 77ff (first edition, 1990)
- [3] "Canonical expressions in Boolean algebra", Dissertation, Dept. of Mathematics, U. of Chicago, 1937, reviewed in J. C. C. McKinsey, *The Journal of Symbolic Logic* 3:2:93 (June 1938)

Canonical normal form

In Boolean algebra, any Boolean function can be put into the **canonical disjunctive normal form (CDNF)** or **minterm canonical form** and its dual **canonical conjunctive normal form (CCNF)** or **maxterm canonical form**. Other canonical forms include the complete sum of prime implicants or Blake canonical form (and its dual), and the algebraic normal form (also called Zhegalkin or Reed–Muller).

Minterms are called products because they are the logical AND of a set of variables, and *maxterms* are called sums because they are the logical OR of a set of variables. These concepts are dual because of their complementary-symmetry relationship as expressed by De Morgan's laws.

Two dual canonical forms of *any* Boolean function are a "sum of minterms" and a "product of maxterms." The term "**Sum of Products**" or "**SoP**" is widely used for the canonical form that is a disjunction (OR) of minterms. Its De Morgan dual is a "**Product of Sums**" or "**PoS**" for the canonical form that is a conjunction (AND) of maxterms. These forms allow for greater analysis into the simplification of these functions, which is of great importance in the minimization or other optimization of Boolean formulas in general and digital circuits in particular.

Summary

One application of Boolean algebra is digital circuit design. The goal may be to minimize the number of gates, to minimize the settling time, etc.

There are sixteen possible functions of two variables, but in digital logic hardware, the simplest gate circuits implement only four of them: *conjunction* (AND), *disjunction* (inclusive OR), and the complements of those (NAND and NOR).

Most gate circuits accept more than 2 input variables; for example, the spaceborne Apollo Guidance Computer, which pioneered the application of integrated circuits in the 1960s, was built with only one type of gate, a 3-input NOR, whose output is true only when all 3 inputs are false.^[1]

Minterms

For a boolean function of n variables x_1, \dots, x_n , a product term in which each of the n variables appears **once** (in either its complemented or uncomplemented form) is called a *minterm*. Thus, a *minterm* is a logical expression of n variables that employs only the *complement* operator and the *conjunction* operator.

For example, abc , $ab'c$ and abc' are 3 examples of the 8 minterms for a Boolean function of the three variables a , b , and c . The customary reading of the last of these is a AND b AND NOT- c .

There are 2^n minterms of n variables, since a variable in the minterm expression can be in either its direct or its complemented form—two choices per n variables.

Indexing minterms

Minterms are often numbered by a binary encoding of the complementation pattern of the variables, where the variables are written in a standard order, usually alphabetical. This convention assigns the value 1 to the direct form (x_i) and 0 to the complemented form (x'_i); the minterm is then the sum of $2^i \text{value}(x_i)$. For example, minterm abc' is numbered $110_2 = 6_{10}$ and denoted m_6 .

Functional equivalence

A given minterm n gives a true value (i.e., 1) for just one combination of the input variables. For example, minterm 5, $a b' c$, is true only when a and c both are true and b is false—the input arrangement where $a = 1$, $b = 0$, $c = 1$ results in 1.

Given the truth table of a logical function, it is possible to write the function as a "sum of products". This is a special form of disjunctive normal form. For example, if given the truth table for the arithmetic sum bit u of one bit position's logic of an adder circuit, as a function of x and y from the addends and the carry in, ci :

ci	x	y	u(ci,x,y)
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Observing that the rows that have an output of 1 are the 2nd, 3rd, 5th, and 8th, we can write u as a sum of minterms m_1, m_2, m_4 , and m_7 . If we wish to verify this: $u(ci, x, y) = m_1 + m_2 + m_4 + m_7 = (ci', x', y) + (ci', x, y') + (ci, x', y') + (ci, x, y)$ evaluated for all 8 combinations of the three variables will match the table.

Maxterms

For a boolean function of n variables x_1, \dots, x_n , a sum term in which each of the n variables appears **once** (in either its complemented or uncomplemented form) is called a *maxterm*. Thus, a *maxterm* is a logical expression of n variables that employs only the *complement* operator and the *disjunction* operator. Maxterms are a dual of the minterm idea (i.e., exhibiting a complementary symmetry in all respects). Instead of using ANDs and complements, we use ORs and complements and proceed similarly.

For example, the following are two of the eight maxterms of three variables:

$$a + b' + c$$

$$a' + b + c$$

There are again 2^n maxterms of n variables, since a variable in the maxterm expression can also be in either its direct or its complemented form—two choices per n variables.

Indexing maxterms

Each maxterm is assigned an index based on the opposite conventional binary encoding used for minterms. The maxterm convention assigns the value 0 to the direct form (x_i) and 1 to the complemented form (x'_i). For example, we assign the index 6 to the maxterm $a' + b' + c$ (110) and denote that maxterm as M_6 . Similarly M_0 of these three variables is $a + b + c$ (000) and M_7 is $a' + b' + c'$ (111).

Functional equivalence

It is apparent that maxterm n gives a *false* value (i.e., 0) for just one combination of the input variables. For example, maxterm 5, $a' + b + c'$, is false only when a and c both are true and b is false—the input arrangement where $a = 1, b = 0, c = 1$ results in 0.

If one is given a truth table of a logical function, it is possible to write the function as a "product of sums". This is a special form of conjunctive normal form. For example, if given the truth table for the carry-out bit co of one bit position's logic of an adder circuit, as a function of x and y from the addends and the carry in, ci :

ci	x	y	co(ci,x,y)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Observing that the rows that have an output of 0 are the 1st, 2nd, 3rd, and 5th, we can write co as a product of maxterms M_0, M_1, M_2 , and M_4 . If we wish to verify this: $co(ci, x, y) = M_0M_1M_2M_4 = (ci + x + y)(ci + x + y')(ci + x' + y)(ci' + x + y)$ evaluated for all 8 combinations of the three variables will match the table.

Dualization

The complement of a minterm is the respective maxterm. This can be easily verified by using de Morgan's law. For example: $M_5 = a' + b + c' = (ab'c)' = m'_5$

Non-canonical PoS and SoP forms

It is often the case that the canonical minterm form can be simplified to an equivalent SoP form. This simplified form would still consist of a sum of product terms. However, in the simplified form, it is possible to have fewer product terms and/or product terms that contain fewer variables. For example, the following 3-variable function:

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

has the canonical minterm representation: $f = a'bc + abc$, but it has an equivalent simplified form: $f = bc$.

In this trivial example, it is obvious that $bc = a'bc + abc$, but the simplified form has both fewer product terms, and the term has fewer variables. The most simplified SoP representation of a function is referred to as a *minimal SoP form*.

In a similar manner, a canonical maxterm form can have a simplified PoS form.

While this example was easily simplified by applying normal algebraic methods [$f = (a' + a)bc$], in less obvious cases a convenient method for finding the minimal PoS/SoP form of a function with up to four variables is using a Karnaugh map.

The minimal PoS and SoP forms are very important for finding optimal implementations of boolean functions and minimizing logic circuits.

Application example

The sample truth tables for minterms and maxterms above are sufficient to establish the canonical form for a single bit position in the addition of binary numbers, but are not sufficient to design the digital logic unless your inventory of gates includes AND and OR. Where performance is an issue (as in the Apollo Guidance Computer), the available parts are more likely to be NAND and NOR because of the complementing action inherent in transistor logic. The values are defined as voltage states, one near ground and one near the DC supply voltage V_{cc} , e.g. +5 VDC. If the higher voltage is defined as the 1 "true" value, a NOR gate is the simplest possible useful logical element.

Specifically, a 3-input NOR gate may consist of 3 bipolar junction transistors with their emitters all grounded, their collectors tied together and linked to V_{cc} through a load impedance. Each base is connected to an input signal, and the common collector point presents the output signal. Any input that is a 1 (high voltage) to its base shorts its transistor's emitter to its collector, causing current to flow through the load impedance, which brings the collector voltage (the output) very near to ground. That result is independent of the other inputs. Only when all 3 input signals are 0 (low voltage) do the emitter-collector impedances of all 3 transistors remain very high. Then very little current flows, and the voltage-divider effect with the load impedance imposes on the collector point a high voltage very near

to V_{cc} .

The complementing property of these gate circuits may seem like a drawback when trying to implement a function in canonical form, but there is a compensating bonus: such a gate with only one input implements the complementing function, which is required frequently in digital logic.

This example assumes the Apollo parts inventory: 3-input NOR gates only, but the discussion is simplified by supposing that 4-input NOR gates are also available (in Apollo, those were compounded out of pairs of 3-input NORs).

Canonical and non-canonical consequences of NOR gates

Fact #1: a set of 8 NOR gates, if their inputs are all combinations of the direct and complement forms of the 3 input variables ci , x , and y , always produce minterms, never maxterms—that is, of the 8 gates required to process all combinations of 3 input variables, only one has the output value 1. That's because a NOR gate, despite its name, could better be viewed (using De Morgan's law) as the AND of the complements of its input signals.

Fact #2: the reason Fact #1 is not a problem is the duality of minterms and maxterms, i.e. each maxterm is the complement of the like-indexed minterm, and vice versa.

In the minterm example above, we wrote $u(ci, x, y) = m_1 + m_2 + m_4 + m_7$ but to perform this with a 4-input NOR gate we need to restate it as a product of sums (PoS), where the sums are the opposite maxterms. That is,

$u(ci, x, y) = \text{AND}(M_0, M_3, M_5, M_6) = \text{NOR}(m_0, m_3, m_5, m_6)$. Truth tables:

ci	x	y	M_0	M_3	M_5	M_6	AND	$u(ci, x, y)$
0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1
0	1	1	1	0	1	1	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	1	0	1	0	0
1	1	0	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1

ci	x	y	m_0	m_3	m_5	m_6	NOR	$u(ci, x, y)$
0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	1	1
0	1	1	0	1	0	0	0	0
1	0	0	0	0	0	0	1	1
1	0	1	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	0	1	1

In the maxterm example above, we wrote $co(ci, x, y) = M_0 M_1 M_2 M_4$ but to perform this with a 4-input NOR gate we need to notice the equality to the NOR of the same minterms. That is,

$co(ci, x, y) = \text{AND}(M_0, M_1, M_2, M_4) = \text{NOR}(m_0, m_1, m_2, m_4)$. Truth tables:

ci	x	y	M₀	M₁	M₂	M₄	AND	co(ci,x,y)
0	0	0	0	1	1	1	0	0
0	0	1	1	0	1	1	0	0
0	1	0	1	1	0	1	0	0
0	1	1	1	1	1	1	1	1
1	0	0	1	1	1	0	0	0
1	0	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

ci	x	y	m₀	m₁	m₂	m₄	NOR	co(ci,x,y)
0	0	0	1	0	0	0	0	0
0	0	1	0	1	0	0	0	0
0	1	0	0	0	1	0	0	0
0	1	1	0	0	0	0	1	1
1	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	1	1
1	1	0	0	0	0	0	1	1
1	1	1	0	0	0	0	1	1

Design trade-offs considered in addition to canonical forms

One might suppose that the work of designing an adder stage is now complete, but we haven't addressed the fact that all 3 of the input variables have to appear in both their direct and complement forms. There's no difficulty about the addends x and y in this respect, because they are static throughout the addition and thus are normally held in latch circuits that routinely have both direct and complement outputs. (The simplest latch circuit made of NOR gates is a pair of gates cross-coupled to make a flip-flop: the output of each is wired as one of the inputs to the other.) There is also no need to create the complement form of the sum u . However, the carry out of one bit position must be passed as the carry into the next bit position in both direct and complement forms. The most straightforward way to do this is to pass co through a 1-input NOR gate and label the output co' , but that would add a gate delay in the worst possible place, slowing down the rippling of carries from right to left. An additional 4-input NOR gate building the canonical form of co' (out of the opposite minterms as co) solves this problem.

$$co'(ci, x, y) = \text{AND}(M_3, M_5, M_6, M_7) = \text{NOR}(m_3, m_5, m_6, m_7).$$

Truth tables:

ci	x	y	M_3	M_5	M_6	M_7	AND	$co'(ci,x,y)$
0	0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1
0	1	0	1	1	1	1	1	1
0	1	1	0	1	1	1	0	0
1	0	0	1	1	1	1	1	1
1	0	1	1	0	1	1	0	0
1	1	0	1	1	0	1	0	0
1	1	1	1	1	1	0	0	0

ci	x	y	m_3	m_5	m_6	m_7	NOR	$co'(ci,x,y)$
0	0	0	0	0	0	0	1	1
0	0	1	0	0	0	0	1	1
0	1	0	0	0	0	0	1	1
0	1	1	1	0	0	0	0	0
1	0	0	0	0	0	0	1	1
1	0	1	0	1	0	0	0	0
1	1	0	0	0	1	0	0	0
1	1	1	0	0	0	1	0	0

The trade-off to maintain full speed in this way includes an unexpected cost (in addition to having to use a bigger gate). If we'd just used that 1-input gate to complement co , there would have been no use for the minterm m_7 , and the gate that generated it could have been eliminated. Nevertheless, it's still a good trade.

Now we could have implemented those functions exactly according to their SoP and PoS canonical forms, by turning NOR gates into the functions specified. A NOR gate is made into an OR gate by passing its output through a 1-input NOR gate; and it is made into an AND gate by passing each of its inputs through a 1-input NOR gate. However, this approach not only increases the number of gates used, but also doubles the number of gate delays processing the signals, cutting the processing speed in half. Consequently, whenever performance is vital, going beyond canonical forms and doing the Boolean algebra to make the unenhanced NOR gates do the job is well worthwhile.

Top-down vs. bottom-up design

We have now seen how the minterm/maxterm tools can be used to design an adder stage in canonical form with the addition of some Boolean algebra, costing just 2 gate delays for each of the outputs. That's the "top-down" way to design the digital circuit for this function, but is it the best way? The discussion has focused on identifying "fastest" as "best," and the augmented canonical form meets that criterion flawlessly, but sometimes other factors predominate. The designer may have a primary goal of minimizing the number of gates, and/or of minimizing the fanouts of signals to other gates since big fanouts reduce resilience to a degraded power supply or other environmental factors. In such a case, a designer may develop the canonical-form design as a baseline, then try a bottom-up development, and finally compare the results.

The bottom-up development involves noticing that $u = ci \text{ XOR } (x \text{ XOR } y)$, where XOR means eXclusive OR [true when either input is true but not when both are true], and that $co = ci \cdot x + x \cdot y + y \cdot ci$. One such development takes twelve NOR gates in all: six 2-input gates and two 1-input gates to produce u in 5 gate delays, plus three 2-input gates and one 3-input gate to produce co' in 2 gate delays. The canonical baseline took eight 3-input NOR gates plus

three 4-input NOR gates to produce u , co and co' in 2 gate delays. If the circuit inventory actually includes 4-input NOR gates, the top-down canonical design looks like a winner in both gate count and speed. But if (contrary to our convenient supposition) the circuits are actually 3-input NOR gates, of which two are required for each 4-input NOR function, then the canonical design takes 14 gates compared to 12 for the bottom-up approach, but still produces the sum digit u considerably faster. The fanout comparison is tabulated as:

Variables	Top-down	Bottom-up
x	4	1
x'	4	3
y	4	1
y'	4	3
ci	4	1
ci'	4	3
M or m	4@1,4@2	N/A
x XOR y	N/A	2
Misc	N/A	5@1
Max	4	3

What's a decision-maker to do? An observant one will have noticed that the description of the bottom-up development mentions co' as an output but not co . Does that design simply never need the direct form of the carry out? Well, yes and no. At each stage, the calculation of co' depends only on ci' , x' and y' , which means that the carry propagation ripples along the bit positions just as fast as in the canonical design without ever developing co . The calculation of u , which does require ci to be made from ci' by a 1-input NOR, is slower but for any word length the design only pays that penalty once (when the leftmost sum digit is developed). That's because those calculations overlap, each in what amounts to its own little pipeline without affecting when the next bit position's sum bit can be calculated. And, to be sure, the co' out of the leftmost bit position will probably have to be complemented as part of the logic determining whether the addition overflowed. But using 3-input NOR gates, the bottom-up design is very nearly as fast for doing parallel addition on a non-trivial word length, cuts down on the gate count, and uses lower fanouts ... so it wins if gate count and/or fanout are paramount!

We'll leave the exact circuitry of the bottom-up design of which all these statements are true as an exercise for the interested reader, assisted by one more algebraic formula: $u = ci(x \text{ XOR } y) + ci'(x \text{ XOR } y)'J$. Decoupling the carry propagation from the sum formation in this way is what elevates the performance of a *carry-lookahead adder* over that of a *ripple carry adder*.

To see how NOR gate logic was used in the Apollo Guidance Computer's ALU, visit http://klabs.org/history/ech/agc_schematics/index.htm, select any of the 4-BIT MODULE entries in the Index to Drawings, and expand images as desired.

Footnotes

[1] Eldon C. Hall, Journey to the Moon: The History of the Apollo Guidance Computer, AIAA 1996. ISBN 1-56347-185-X

References

- Edward A. Bender, S. Gill Williamson, 2005, *A Short Course in Discrete Mathematics*, Dover Publications, Inc., Mineola, NY, ISBN 0-486-43946-1. The authors demonstrate a proof that any Boolean (logic) function can be expressed in either disjunctive or conjunctive normal form (cf pages 5–6); the proof simply proceeds by creating all 2^N rows of N Boolean variables and demonstrates that each row ("minterm" or "maxterm") has a unique

Boolean expression. Any Boolean function of the N variables can be derived from a composite of the rows whose minterm or maxterm are logical 1s ("trues").

- E. J. McCluskey, 1965, *Introduction to the Theory of Switching Circuits*, McGraw–Hill Book Company, NY, Library of Congress Catalog Card Number 65-17394. Canonical expressions are defined and described on pages 78ff.
- Fredrick J. Hill, and Gerald R. Peterson, 1974, *Introduction to Switching Theory and Logical Design, Second Edition*, John Wiley & Sons, NY, ISBN 0-471-39882-9. Minterm and maxterm designation of functions appears on page 101ff.

External links

- George Boole, 1848, "The Calculus of Logic, (<http://www.maths.tcd.ie/pub/HistMath/People/Boole/CalcLogic/CalcLogic.html>)" *Cambridge and Dublin Mathematical Journal III: 183–98*.

Herbrand normal form

The **Herbrandization** of a logical formula (named after Jacques Herbrand) is a construction that is dual to the Skolemization of a formula. Thoralf Skolem had considered the Skolemizations of formulas in prenex form as part of his proof of the Löwenheim-Skolem theorem (Skolem 1920). Herbrand worked with this dual notion of Herbrandization, generalized to apply to non-prenex formulas as well, in order to prove Herbrand's theorem (Herbrand 1930).

The resulting formula is not necessarily equivalent to the original one. As with Skolemization which only preserves satisfiability, Herbrandization being Skolemization's dual preserves validity: the resulting formula is valid if and only if the original one is.

Let F be a formula in the language of first-order logic. We may assume that F contains no variable that is bound by two different quantifier occurrences, and that no variable occurs both bound and free. (That is, F could be relettered to ensure these conditions, in such a way that the result is an equivalent formula).

The *Herbrandization* of F is then obtained as follows:

- First, replace any free variables in F by constant symbols.
- Second, delete all quantifiers on variables that are either (1) universally quantified and within an even number of negation signs, or (2) existentially quantified and within an odd number of negation signs.
- Finally, replace each such variable v with a function symbol $f_v(x_1, \dots, x_k)$, where x_1, \dots, x_k are the variables that are still quantified, and whose quantifiers govern v .

For instance, consider the formula $F := \forall y \exists x [R(y, x) \wedge \neg \exists z S(x, z)]$. There are no free variables to replace.

The variables y, z are the kind we consider for the second step, so we delete the quantifiers $\forall y$ and $\exists z$. Finally, we then replace y with a constant c_y (since there were no other quantifiers governing y), and we replace z with a function symbol $f_z(x)$:

$$F^H = \exists x [R(c_y, x) \wedge \neg S(x, f_z(x))].$$

The *Skolemization* of a formula is obtained similarly, except that in the second step above, we would delete quantifiers on variables that are either (1) existentially quantified and within an even number of negations, or (2) universally quantified and within an odd number of negations. Thus, considering the same F from above, its Skolemization would be:

$$F^S = \forall y [R(y, f_x(y)) \wedge \neg \exists z S(f_x(y), z)].$$

To understand the significance of these constructions, see Herbrand's theorem or the Löwenheim-Skolem theorem.

References

- Skolem, T. "Logico-combinatorial investigations in the satisfiability or provability of mathematical propositions: A simplified proof of a theorem by L. Löwenheim and generalizations of the theorem". (In van Heijenoort 1967, 252-63.)
- Herbrand, J. "Investigations in proof theory: The properties of true propositions". (In van Heijenoort 1967, 525-81.)
- van Heijenoort, J. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*. Harvard University Press, 1967.

Herbrandization

The **Herbrandization** of a logical formula (named after Jacques Herbrand) is a construction that is dual to the Skolemization of a formula. Thoralf Skolem had considered the Skolemizations of formulas in prenex form as part of his proof of the Löwenheim-Skolem theorem (Skolem 1920). Herbrand worked with this dual notion of Herbrandization, generalized to apply to non-prenex formulas as well, in order to prove Herbrand's theorem (Herbrand 1930).

The resulting formula is not necessarily equivalent to the original one. As with Skolemization which only preserves satisfiability, Herbrandization being Skolemization's dual preserves validity: the resulting formula is valid if and only if the original one is.

Let F be a formula in the language of first-order logic. We may assume that F contains no variable that is bound by two different quantifier occurrences, and that no variable occurs both bound and free. (That is, F could be relettered to ensure these conditions, in such a way that the result is an equivalent formula).

The *Herbrandization* of F is then obtained as follows:

- First, replace any free variables in F by constant symbols.
- Second, delete all quantifiers on variables that are either (1) universally quantified and within an even number of negation signs, or (2) existentially quantified and within an odd number of negation signs.
- Finally, replace each such variable v with a function symbol $f_v(x_1, \dots, x_k)$, where x_1, \dots, x_k are the variables that are still quantified, and whose quantifiers govern v .

For instance, consider the formula $F := \forall y \exists x [R(y, x) \wedge \neg \exists z S(x, z)]$. There are no free variables to replace.

The variables y, z are the kind we consider for the second step, so we delete the quantifiers $\forall y$ and $\exists z$. Finally, we then replace y with a constant c_y (since there were no other quantifiers governing y), and we replace z with a function symbol $f_z(x)$:

$$F^H = \exists x [R(c_y, x) \wedge \neg S(x, f_z(x))].$$

The *Skolemization* of a formula is obtained similarly, except that in the second step above, we would delete quantifiers on variables that are either (1) existentially quantified and within an even number of negations, or (2) universally quantified and within an odd number of negations. Thus, considering the same F from above, its Skolemization would be:

$$F^S = \forall y [R(y, f_x(y)) \wedge \neg \exists z S(f_x(y), z)].$$

To understand the significance of these constructions, see Herbrand's theorem or the Löwenheim-Skolem theorem.

References

- Skolem, T. "Logico-combinatorial investigations in the satisfiability or provability of mathematical propositions: A simplified proof of a theorem by L. Löwenheim and generalizations of the theorem". (In van Heijenoort 1967, 252-63.)
- Herbrand, J. "Investigations in proof theory: The properties of true propositions". (In van Heijenoort 1967, 525-81.)
- van Heijenoort, J. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*. Harvard University Press, 1967.

Horn clause

In mathematical logic, a **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**. 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.

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:	can be read intuitively as:
definite clause:	$\neg p \vee \neg q \vee \dots \vee \neg t \vee u$	$u \leftarrow p \wedge q \wedge \dots \wedge t$	assume that u holds if p and q and ... and t all hold
fact:	u	u	assume that u holds
goal clause:	$\neg p \vee \neg q \vee \dots \vee \neg t$	$false \leftarrow p \wedge q \wedge \dots \wedge t$	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:

$$\neg \text{human}(X) \vee \text{mortal}(X)$$

stands for:

$$\forall X (\neg \text{human}(X) \vee \text{mortal}(X))$$

which is logically equivalent to:

$$\forall X (\text{human}(X) \rightarrow \text{mortal}(X))$$

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:

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

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:

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

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:

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

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

$$\forall X (\text{false} \leftarrow p \wedge q \wedge \dots \wedge t)$$

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.

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*^[2]. Journal of ACM, Vol. 23, 733–742.
- [1] Like in resolution theorem proving, intuitive meanings "show φ " and "assume $\neg\varphi$ " are synonymous (indirect proof); they both correspond to the same formula, viz. $\neg\varphi$. This way, a mechanical proving tool needs to maintain only one set of formulas (assumptions), rather than two sets (assumptions and (sub)goals).
- [2] http://www.doc.ic.ac.uk/~rak/papers/kowalski-van_emden.pdf

Negation normal form

Negation normal form is an elementary canonical form in mathematical logic. There are similar requirements for negation normal form in different logic fragments. Note that the equivalence relation used is not equality, since $(A \vee \neg A)$ does not get rewritten to True.

In predicate logic, a logical formula is in **negation normal form** if negation occurs only immediately above elementary propositions, and $\{\neg, \vee, \wedge\}$ are the only allowed Boolean connectives. In classical logic each formula can be brought into this form by replacing implications and equivalences by their definitions, using De Morgan's laws to push negation inside, and eliminating double negations. This process can be represented using the following rewrite rules:

$$\begin{aligned}\neg(\forall x.G) &\rightarrow \exists x.\neg G \\ \neg(\exists x.G) &\rightarrow \forall x.\neg G \\ \neg\neg G &\rightarrow G \\ \neg(G_1 \wedge G_2) &\rightarrow (\neg G_1) \vee (\neg G_2) \\ \neg(G_1 \vee G_2) &\rightarrow (\neg G_1) \wedge (\neg G_2)\end{aligned}$$

A formula in negation normal form can be put into the stronger conjunctive normal form or disjunctive normal form by applying the distributivity laws.

Examples and counterexamples

The following formulae are all **in negation normal form**:

$$\begin{aligned}(A \vee B) \wedge C \\ (A \wedge (\neg B \vee C) \wedge \neg C) \vee D \\ A \vee \neg B \\ A \wedge \neg B\end{aligned}$$

Note that whilst the first example is also in conjunctive normal form and the last two are in both conjunctive normal form and disjunctive normal form, the second example is in neither.

The following formulae are **not in negation normal form**:

$$\begin{aligned}A \Rightarrow B \\ \neg(A \vee B) \\ \neg(A \wedge B) \\ \neg(A \vee \neg C)\end{aligned}$$

They are however respectively equivalent to the following formulae **in negation normal form**:

$$\begin{aligned} & \neg A \vee B \\ & \neg A \wedge \neg B \\ & \neg A \vee \neg B \\ & \neg A \wedge C \end{aligned}$$

External links

- Java applet for converting logical formula to Negation Normal Form, showing laws used [2]

Prenex normal form

A formula of the predicate calculus is in **prenex^[1] normal form** 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

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

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

and

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

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

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,

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

but

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

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

$$\neg \exists x \phi \text{ is equivalent to } \forall x \neg \phi$$

and

$$\neg \forall x \phi \text{ is equivalent to } \exists x \neg \phi.$$

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 $\neg \phi \vee \psi$ 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:

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

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

The rules for removing quantifiers from the consequent are:

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

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

Example

Suppose that ϕ , ψ , and ρ are quantifier-free formulas and no two of these formulas share any free variable.

Consider the formula

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

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

$$\begin{aligned} & (\phi \vee \exists x \psi) \rightarrow \forall z \rho. \\ & (\exists x (\phi \vee \psi)) \rightarrow \forall z \rho, \\ & \neg (\exists x (\phi \vee \psi)) \vee \forall z \rho, \\ & (\forall x \neg (\phi \vee \psi)) \vee \forall z \rho, \\ & \forall x (\neg (\phi \vee \psi) \vee \forall z \rho), \\ & \forall x ((\phi \vee \psi) \rightarrow \forall z \rho), \\ & \forall x (\forall z ((\phi \vee \psi) \rightarrow \rho)), \\ & \forall x \forall z ((\phi \vee \psi) \rightarrow \rho). \end{aligned}$$

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

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

can be obtained:

$$\begin{aligned} & \forall z ((\phi \vee \exists x \psi) \rightarrow \rho) \\ & \forall z ((\exists x (\phi \vee \psi)) \rightarrow \rho), \\ & \forall z (\forall x ((\phi \vee \psi) \rightarrow \rho)), \\ & \forall z \forall x ((\phi \vee \psi) \rightarrow \rho). \end{aligned}$$

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

$$(\exists x\phi) \rightarrow \exists y\psi \quad (1)$$

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

$$\exists y(\exists x\phi \rightarrow \psi), \quad (2)$$

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 $\psi(y)$. 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) $(\forall x\phi) \rightarrow \psi$ implies $\exists x(\phi \rightarrow \psi)$,
- (4) $\phi \rightarrow (\exists x\psi)$ implies $\exists x(\phi \rightarrow \psi)$,
- (5) $\neg\forall x\phi$ implies $\exists x\neg\phi$,

(x does not appear as a free variable of ψ in (1) and (3); x does not appear as a free variable of ϕ 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>).

References

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

Skolem normal form

In mathematical logic, reduction to **Skolem normal form** (SNF) is a method for removing existential quantifiers from formal logic statements, often performed as the first step in an automated theorem prover.

A formula of first-order logic is in Skolem normal form (named after Thoralf Skolem) if it is in conjunctive prenex normal form with only universal first-order quantifiers. Every first-order formula can be converted into Skolem normal form while not changing its satisfiability via a process called **Skolemization** (sometimes spelled "Skolemization"). The resulting formula is not necessarily equivalent to the original one, but is equisatisfiable with it: it is satisfiable if and only if the original one is satisfiable.

The simplest form of Skolemization is for existentially quantified variables which are not inside the scope of a universal quantifier. These can simply be replaced by creating new constants. For example, $\exists x P(x)$ can be changed to $P(c)$, where c is a new constant (does not occur anywhere else in the formula).

More generally, Skolemization is performed by replacing every existentially quantified variable y with a term $f(x_1, \dots, x_n)$ whose function symbol f is new. The variables of this term are as follows. If the formula is in prenex normal form, x_1, \dots, x_n are the variables that are universally quantified and whose quantifiers precede that of y . In general, they are the variables that are universally quantified Wikipedia:Please clarify and such that $\exists y$ occurs in the scope of their quantifiers. The function f introduced in this process is called a **Skolem function** (or **Skolem constant** if it is of zero arity) and the term is called a **Skolem term**.

As an example, the formula $\forall x \exists y \forall z. P(x, y, z)$ is not in Skolem normal form because it contains the existential quantifier $\exists y$. Skolemization replaces y with $f(x)$, where f is a new function symbol, and removes the quantification over y . The resulting formula is $\forall x \forall z. P(x, f(x), z)$. The Skolem term $f(x)$ contains x but not z because the quantifier to be removed $\exists y$ is in the scope of $\forall x$ but not in that of $\forall z$; since this formula is in prenex normal form, this is equivalent to saying that, in the list of quantifiers, x precedes y while z does not. The formula obtained by this transformation is satisfiable if and only if the original formula is.

How Skolemization works

Skolemization works by applying a second-order equivalence in conjunction to the definition of first-order satisfiability. The equivalence provides a way for "moving" an existential quantifier before a universal one.

$$\forall x (g(x) \vee \exists y R(x, y)) \iff \forall x (g(x) \vee R(x, f(x)))$$

where

$f(x)$ is a function that maps x to y .

Intuitively, the sentence "for every x there exists a y such that $R(x, y)$ " is converted into the equivalent form "there exists a function f mapping every x into a y such that, for every x it holds $R(x, f(x))$ ".

This equivalence is useful because the definition of first-order satisfiability implicitly existentially quantifies over the evaluation of function symbols. In particular, a first-order formula Φ is satisfiable if there exists a model M and an evaluation μ of the free variables of the formula that evaluate the formula to *true*. The model contains the evaluation of all function symbols; therefore, Skolem functions are implicitly existentially quantified. In the example above, $\forall x. R(x, f(x))$ is satisfiable if and only if there exists a model M , which contains an evaluation for f , such that $\forall x. R(x, f(x))$ is true for some evaluation of its free variables (none in this case). This can be expressed in second order as $\exists f \forall x. R(x, f(x))$. By the above equivalence, this is the same as the satisfiability of $\forall x \exists y. R(x, y)$.

At the meta-level, first-order satisfiability of a formula Φ can be written with a little abuse of notation as $\exists M \exists \mu . (M, \mu \models \Phi)$, where M is a model, μ is an evaluation of the free variables., and \models means that Φ is a semantic consequence of M and μ . Since first-order models contain the evaluation of all function symbols, any

Skolem function Φ contains implicitly existentially quantified by $\exists M$. As a result, after replacing an existential quantifier over variables into an existential quantifiers over functions at the front of the formula, the formula can still be treated as a first-order one by removing these existential quantifiers. This final step of treating $\exists f \forall x. R(x, f(x))$ as $\forall x. R$ because functions are implicitly existentially quantified by $\exists M$ in the definition of first-order satisfiability.

Correctness of Skolemization can be shown on the example formula $F_1 = \forall x_1 \dots \forall x_n \exists y R(x_1, \dots, x_n, y)$ as follows. This formula is satisfied by a model M if and only if, for each possible value for x_1, \dots, x_n in the domain of the model there exists a value for y in the domain of the model that makes $R(x_1, \dots, x_n, y)$ true. By the axiom of choice, there exists a function f such that $y = f(x_1, \dots, x_n)$. As a result, the formula $F_2 = \forall x_1 \dots \forall x_n R(x_1, \dots, x_n, f(x_1, \dots, x_n))$ is satisfiable, because it has the model obtained by adding the evaluation of f to M . This shows that F_1 is satisfiable only if F_2 is satisfiable as well. In the other way around, if F_2 is satisfiable, then there exists a model M' that satisfies it; this model includes an evaluation for the function f such that, for every value of x_1, \dots, x_n , the formula $R(x_1, \dots, x_n, f(x_1, \dots, x_n))$ holds. As a result, F_1 is satisfied by the same model because one can choose, for every value of x_1, \dots, x_n , the value $y = f(x_1, \dots, x_n)$, where f is evaluated according to M' .

Uses of Skolemization

One of the uses of Skolemization is automated theorem proving. For example, in the method of analytic tableaux, whenever a formula whose leading quantifier is existential occurs, the formula obtained by removing that quantifier via Skolemization can be generated. For example, if $\exists x. \Phi(x, y_1, \dots, y_n)$ occurs in a tableau, where x, y_1, \dots, y_n are the free variables of $\Phi(x, y_1, \dots, y_n)$, then $\Phi(f(y_1, \dots, y_n), y_1, \dots, y_n)$ can be added to the same branch of the tableau. This addition does not alter the satisfiability of the tableau: every model of the old formula can be extended, by adding a suitable evaluation of f , to a model of the new formula.

This form of Skolemization is actually an improvement over "classical" Skolemization in that only variables that are free in the formula are placed in the Skolem term. This is an improvement because the semantics of tableau may implicitly place the formula in the scope of some universally quantified variables that are not in the formula itself; these variables are not in the Skolem term, while they would be there according to the original definition of Skolemization. Another improvement that can be used is using the same Skolem function symbol for formulae that are identical up to variable renaming.^[1]

Another use is in the resolution method for first order logic, where formulas are represented as sets of clauses understood to be universally quantified. (For an example see drinker paradox.)

Skolem theories

In general, if T is a theory and for each formula F with free variables x_1, \dots, x_n, y there is a Skolem function, then T is called a **Skolem theory**. For example, by the above, arithmetic with the Axiom of Choice is a Skolem theory.

Every Skolem theory is model complete, i.e. every substructure of a model is an elementary substructure. Given a model M of a Skolem theory T , the smallest substructure containing a certain set A is called the **Skolem hull** of A . The Skolem hull of A is an atomic prime model over A .

Notes

[1] R. Hähnle. Tableaux and related methods. Handbook of Automated Reasoning.

References

- Hodges, Wilfrid (1997), *A shorter model theory*, Cambridge University Press, ISBN 978-0-521-58713-6

External links

- Hazewinkel, Michiel, ed. (2001), "Skolem function" (<http://www.encyclopediaofmath.org/index.php?title=p/s085740>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Skolemization on PlanetMath.org (<http://planetmath.org/encyclopedia/Skolemization.html>)
- Skolemization (<http://demonstrations.wolfram.com/Skolemization/>) by Hector Zenil, The Wolfram Demonstrations Project.
- Weisstein, Eric W., " SkolemizedForm (<http://mathworld.wolfram.com/SkolemizedForm.html>)", *MathWorld*.

Theorems and specific laws

Absorption law

In algebra, the **absorption law** or **absorption identity** is an identity linking a pair of binary operations.

Two binary operations, say \bowtie and $*$, are said to be connected by the absorption law if:

$$a \bowtie (a * b) = a * (a \bowtie b) = a.$$

A set equipped with two commutative and associative binary operations \vee ("join") and \wedge ("meet") that are connected by the absorption law

$$a \vee (a \wedge b) = a \wedge (a \vee b) = a$$

is called a lattice.

Examples of lattices include Boolean algebras, the set of sets with the *union* and *intersection* operators, Heyting algebras, and ordered sets with *min* and *max* operations.

In classical logic, and in particular in Boolean algebra, the operations OR and AND, which are also denoted by \vee and \wedge , satisfy the lattice axioms, including the absorption law. The same is true for intuitionistic logic.

The absorption law does *not* hold in many other algebraic structures, such as commutative ring, e.g. the field of real numbers, relevance logics, linear logics, and substructural logics. In the last case, there is no one-to-one correspondence between the free variables of the defining pair of identities.

References

- Davey, B. A., and Priestley, H. A. (2002). *Introduction to Lattices and Order* (second ed.). Cambridge University Press. ISBN 0-521-78451-4.
- Hazewinkel, Michiel, ed. (2001), "Absorption laws" ^[1], *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Weisstein, Eric W., "Absorption Law" ^[2], *MathWorld*.

References

[1] <http://www.encyclopediaofmath.org/index.php?title=p/a010440>

[2] <http://mathworld.wolfram.com/AbsorptionLaw.html>

Boole's expansion theorem

Boole's expansion theorem, often referred to as the **Shannon expansion** or **decomposition**, is the identity: $F = x \cdot F_x + x' \cdot F'_x$, where F is any Boolean function and F_x and F'_x are F with the argument x equal to 1 and to 0 respectively.

The terms F_x and F'_x are sometimes called the positive and negative **Shannon cofactors** of F with respect to x .

It has been called the "fundamental theorem of Boolean algebra".^[1] Besides its theoretical importance, it paved the way for binary decision diagrams, satisfiability solvers, and many other techniques relevant to computer engineering and formal verification of digital circuits.

Statement of the theorem

Another way of stating the theorem is:

$$f(X_1, X_2, \dots, X_n) = X_1 \cdot f(1, X_2, \dots, X_n) + X'_1 \cdot f(0, X_2, \dots, X_n)$$

History

George Boole presented this expansion as his Proposition II, "To expand or develop a function involving any number of logical symbols", in his *Laws of Thought* (1854),^[2] and it was "widely applied by Boole and other nineteenth-century logicians".^[3]

Claude Shannon mentioned this expansion, among other Boolean identities, in a 1948 paper,^[4] and showed the switching network interpretations of the identity. In the literature of computer design and switching theory, it is often attributed to Shannon.

Application to switching circuits

It can be implemented directly in a switching circuit using a multiplexer.

Notes

[1] Paul C. Rosenbloom, *The Elements of Mathematical Logic*, 1950, p. 5

[2] George Boole, *An Investigation of the Laws of Thought: On which are Founded the Mathematical Theories of Logic and Probabilities*, 1854, p. 72 full text at Google Books (<http://books.google.com/books?id=SWgLVT0otY8C&pg=PA73&dq=to+expand+or+develop+a+function>)

[3] Frank Markham Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd edition, 2003, p. 42

[4] Claude Shannon, "The Synthesis of Two-Terminal Switching Circuits", *Bell System Technical Journal* **28**:59–98, full text (<http://www3.alcatel-lucent.com/bstj/vol28-1949/articles/bstj28-1-59.pdf>), p. 62

External links

- Shannon's Decomposition (<http://homepages.ius.edu/JFDOYLE/c421/html/Chapter6.htm>) Example with multiplexers.
- Optimizing Sequential Cycles Through Shannon Decomposition and Retiming (PDF) (<http://www1.cs.columbia.edu/~sedwards/papers/soviani2007optimizing.pdf>) Paper on application.

Boolean prime ideal theorem

In mathematics, a **prime ideal theorem** guarantees the existence of certain types of subsets in a given algebra. A common example is the **Boolean prime ideal theorem**, which states that ideals in a Boolean algebra can be extended to prime ideals. A variation of this statement for filters on sets is known as the ultrafilter lemma. Other theorems are obtained by considering different mathematical structures with appropriate notions of ideals, for example, rings and prime ideals (of ring theory), or distributive lattices and *maximal* ideals (of order theory). This article focuses on prime ideal theorems from order theory.

Although the various prime ideal theorems may appear simple and intuitive, they cannot be derived in general from the axioms of Zermelo–Fraenkel set theory without the axiom of choice (abbreviated ZF). Instead, some of the statements turn out to be equivalent to the axiom of choice (AC), while others—the Boolean prime ideal theorem, for instance—represent a property that is strictly weaker than AC. It is due to this intermediate status between ZF and ZF + AC (ZFC) that the Boolean prime ideal theorem is often taken as an axiom of set theory. The abbreviations **BPI** or **PIT** (for Boolean algebras) are sometimes used to refer to this additional axiom.

Prime ideal theorems

Recall that an order ideal is a (non-empty) directed lower set. If the considered poset has binary suprema (a.k.a. joins), as do the posets within this article, then this is equivalently characterized as a lower set I which is closed for binary suprema (i.e. x, y in I imply $x \vee y$ in I). An ideal I is prime if, whenever an infimum $x \wedge y$ is in I , one also has x in I or y in I . Ideals are proper if they are not equal to the whole poset.

Historically, the first statement relating to later prime ideal theorems was in fact referring to filters—subsets that are ideals with respect to the dual order. The ultrafilter lemma states that every filter on a set is contained within some maximal (proper) filter—an *ultrafilter*. Recall that filters on sets are proper filters of the Boolean algebra of its powerset. In this special case, maximal filters (i.e. filters that are not strict subsets of any proper filter) and prime filters (i.e. filters that with each union of subsets X and Y contain also X or Y) coincide. The dual of this statement thus assures that every ideal of a powerset is contained in a prime ideal.

The above statement led to various generalized prime ideal theorems, each of which exists in a weak and in a strong form. *Weak prime ideal theorems* state that every *non-trivial* algebra of a certain class has at least one prime ideal. In contrast, *strong prime ideal theorems* require that every ideal that is disjoint from a given filter can be extended to a prime ideal which is still disjoint from that filter. In the case of algebras that are not posets, one uses different substructures instead of filters. Many forms of these theorems are actually known to be equivalent, so that the assertion that "PIT" holds is usually taken as the assertion that the corresponding statement for Boolean algebras (BPI) is valid.

Another variation of similar theorems is obtained by replacing each occurrence of *prime ideal* by *maximal ideal*. The corresponding **maximal ideal theorems** (MIT) are often—though not always—stronger than their PIT equivalents.

Boolean prime ideal theorem

The Boolean prime ideal theorem is the strong prime ideal theorem for Boolean algebras. Thus the formal statement is:

Let B be a Boolean algebra, let I be an ideal and let F be a filter of B , such that I and F are disjoint. Then I is contained in some prime ideal of B that is disjoint from F .

The weak prime ideal theorem for Boolean algebras simply states:

Every Boolean algebra contains a prime ideal.

We refer to these statements as the weak and strong *BPI*. The two are equivalent, as the strong BPI clearly implies the weak BPI, and the reverse implication can be achieved by using the weak BPI to find prime ideals in the appropriate quotient algebra.

The BPI can be expressed in various ways. For this purpose, recall the following theorem:

For any ideal I of a Boolean algebra B , the following are equivalent:

- I is a prime ideal.
- I is a maximal proper ideal, i.e. for any proper ideal J , if I is contained in J then $I = J$.
- For every element a of B , I contains exactly one of $\{a, \neg a\}$.

This theorem is a well-known fact for Boolean algebras. Its dual establishes the equivalence of prime filters and ultrafilters. Note that the last property is in fact self-dual—only the prior assumption that I is an ideal gives the full characterization. All of the implications within this theorem can be proven in ZF.

Thus the following (strong) maximal ideal theorem (MIT) for Boolean algebras is equivalent to BPI:

Let B be a Boolean algebra, let I be an ideal and let F be a filter of B , such that I and F are disjoint. Then I is contained in some maximal ideal of B that is disjoint from F .

Note that one requires "global" maximality, not just maximality with respect to being disjoint from F . Yet, this variation yields another equivalent characterization of BPI:

Let B be a Boolean algebra, let I be an ideal and let F be a filter of B , such that I and F are disjoint. Then I is contained in some ideal of B that is maximal among all ideals disjoint from F .

The fact that this statement is equivalent to BPI is easily established by noting the following theorem: For any distributive lattice L , if an ideal I is maximal among all ideals of L that are disjoint to a given filter F , then I is a prime ideal. The proof for this statement (which can again be carried out in ZF set theory) is included in the article on ideals. Since any Boolean algebra is a distributive lattice, this shows the desired implication.

All of the above statements are now easily seen to be equivalent. Going even further, one can exploit the fact the dual orders of Boolean algebras are exactly the Boolean algebras themselves. Hence, when taking the equivalent duals of all former statements, one ends up with a number of theorems that equally apply to Boolean algebras, but where every occurrence of *ideal* is replaced by *filter*. It is worth noting that for the special case where the Boolean algebra under consideration is a powerset with the subset ordering, the "maximal filter theorem" is called the ultrafilter lemma.

Summing up, for Boolean algebras, the weak and strong MIT, the weak and strong PIT, and these statements with filters in place of ideals are all equivalent. It is known that all of these statements are consequences of the Axiom of Choice, AC, (the easy proof makes use of Zorn's lemma), but cannot be proven in **ZF** (Zermelo-Fraenkel set theory without AC), if **ZF** is consistent. Yet, the BPI is strictly weaker than the axiom of choice, though the proof of this statement, due to J. D. Halpern and Azriel Levy is rather non-trivial.

Further prime ideal theorems

The prototypical properties that were discussed for Boolean algebras in the above section can easily be modified to include more general lattices, such as distributive lattices or Heyting algebras. However, in these cases maximal ideals are different from prime ideals, and the relation between PITs and MITs is not obvious.

Indeed, it turns out that the MITs for distributive lattices and even for Heyting algebras are equivalent to the axiom of choice. On the other hand, it is known that the strong PIT for distributive lattices is equivalent to BPI (i.e. to the MIT and PIT for Boolean algebras). Hence this statement is strictly weaker than the axiom of choice. Furthermore, observe that Heyting algebras are not self dual, and thus using filters in place of ideals yields different theorems in this setting. Maybe surprisingly, the MIT for the duals of Heyting algebras is not stronger than BPI, which is in sharp contrast to the abovementioned MIT for Heyting algebras.

Finally, prime ideal theorems do also exist for other (not order-theoretical) abstract algebras. For example, the MIT for rings implies the axiom of choice. This situation requires to replace the order-theoretic term "filter" by other concepts—for rings a "multiplicatively closed subset" is appropriate.

The ultrafilter lemma

A **filter** on a set X is a collection of nonempty subsets of X that is closed under finite intersection and under superset. An **ultrafilter** is a maximal filter. The **ultrafilter lemma** states that every filter on a set X is a subset of some ultrafilter on X (a maximal filter of nonempty subsets of X). This lemma is most often used in the study of topology. An ultrafilter that does not contain finite sets is called **non-principal**. The existence of non-principal ultrafilters is due to Tarski in 1930.

The ultrafilter lemma is equivalent to the Boolean prime ideal theorem, with the equivalence provable in ZF set theory without the axiom of choice. The idea behind the proof is that the subsets of any set form a Boolean algebra partially ordered by inclusion, and any Boolean algebra is representable as an algebra of sets by Stone's representation theorem.

Applications

Intuitively, the Boolean prime ideal theorem states that there are "enough" prime ideals in a Boolean algebra in the sense that we can extend *every* ideal to a maximal one. This is of practical importance for proving Stone's representation theorem for Boolean algebras, a special case of Stone duality, in which one equips the set of all prime ideals with a certain topology and can indeed regain the original Boolean algebra (up to isomorphism) from this data. Furthermore, it turns out that in applications one can freely choose either to work with prime ideals or with prime filters, because every ideal uniquely determines a filter: the set of all Boolean complements of its elements. Both approaches are found in the literature.

Many other theorems of general topology that are often said to rely on the axiom of choice are in fact equivalent to BPI. For example, the theorem that a product of compact Hausdorff spaces is compact is equivalent to it. If we leave out "Hausdorff" we get a theorem equivalent to the full axiom of choice.

A not too well known application of the Boolean prime ideal theorem is the existence of a non-measurable set (the example usually given is the Vitali set, which requires the Axiom of Choice). From this and the fact that the BPI is strictly weaker than the Axiom of Choice, it follows that the existence of non-measurable sets is strictly weaker than the axiom of choice.

In linear algebra, the boolean prime ideal theorem can be used to prove that any two bases of a given vector space have the same cardinality.

Notes

References

- Davey, B. A.; Priestley, H. A. (2002), *Introduction to Lattices and Order* (2nd ed.), Cambridge University Press, ISBN 978-0-521-78451-1.

An easy to read introduction, showing the equivalence of PIT for Boolean algebras and distributive lattices.

- Johnstone, Peter (1982), *Stone Spaces*, Cambridge studies in advanced mathematics 3, Cambridge University Press, ISBN 978-0-521-33779-3.

The theory in this book often requires choice principles. The notes on various chapters discuss the general relation of the theorems to PIT and MIT for various structures (though mostly lattices) and give pointers to further literature.

- Banaschewski, B. (1983), "The power of the ultrafilter theorem", *Journal of the London Mathematical Society (2nd series)* 27 (2): 193–202, doi: 10.1112/jlms/s2-27.2.193 (<http://dx.doi.org/10.1112/jlms/s2-27.2.193>).

Discusses the status of the ultrafilter lemma.

- Erné, M. (2000), "Prime ideal theory for general algebras", *Applied Categorical Structures* 8: 115–144, doi: 10.1023/A:1008611926427 (<http://dx.doi.org/10.1023/A:1008611926427>).

Gives many equivalent statements for the BPI, including prime ideal theorems for other algebraic structures. PITs are considered as special instances of separation lemmas.

Compactness theorem

In mathematical logic, the **compactness theorem** states that a set of first-order sentences has a model if and only if every finite subset of it has a model. This theorem is an important tool in model theory, as it provides a useful method for constructing models of any set of sentences that is finitely consistent.

The compactness theorem for the propositional calculus is a consequence of Tychonoff's theorem (which says that the product of compact spaces is compact) applied to compact Stone spaces;^[1] hence, the theorem's name. Likewise, it is analogous to the finite intersection property characterization of compactness in topological spaces: a collection of closed sets in a compact space has a non-empty intersection if every finite subcollection has a non-empty intersection.

The compactness theorem is one of the two key properties, along with the downward Löwenheim–Skolem theorem, that is used in Lindström's theorem to characterize first-order logic. Although there are some generalizations of the compactness theorem to non-first-order logics, the compactness theorem itself does not hold in them.

History

Kurt Gödel proved the countable compactness theorem in 1930. Anatoly Maltsev proved the uncountable case in 1936.^[2] ^[3]

Applications

The compactness theorem has many applications in model theory; a few typical results are sketched here.

The compactness theorem implies Robinson's principle: If a first-order sentence holds in every field of characteristic zero, then there exists a constant p such that the sentence holds for every field of characteristic larger than p . This can be seen as follows: suppose φ is a sentence that holds in every field of characteristic zero. Then its negation $\neg\varphi$, together with the field axioms and the infinite sequence of sentences $1+1 \neq 0, 1+1+1 \neq 0, \dots$, is not satisfiable

(because there is no field of characteristic 0 in which $\neg\varphi$ holds, and the infinite sequence of sentences ensures any model would be a field of characteristic 0). Therefore, there is a finite subset A of these sentences that is not satisfiable. We can assume that A contains $\neg\varphi$, the field axioms, and, for some k , the first k sentences of the form $1+1+\dots+1 \neq 0$ (because adding more sentences doesn't change unsatisfiability). Let B contain all the sentences of A except $\neg\varphi$. Then any model of B is a field of characteristic greater than k , and $\neg\varphi$ together with B is not satisfiable. This means that φ must hold in every model of B , which means precisely that φ holds in every field of characteristic greater than k .

A second application of the compactness theorem shows that any theory that has arbitrarily large finite models, or a single infinite model, has models of arbitrary large cardinality (this is the Upward Löwenheim–Skolem theorem). So, for instance, there are nonstandard models of Peano arithmetic with uncountably many 'natural numbers'. To achieve this, let T be the initial theory and let κ be any cardinal number. Add to the language of T one constant symbol for every element of κ . Then add to T a collection of sentences that say that the objects denoted by any two distinct constant symbols from the new collection are distinct (this is a collection of κ^2 sentences). Since every *finite* subset of this new theory is satisfiable by a sufficiently large finite model of T , or by any infinite model, the entire extended theory is satisfiable. But any model of the extended theory has cardinality at least κ .

A third application of the compactness theorem is the construction of nonstandard models of the real numbers, that is, consistent extensions of the theory of the real numbers that contain "infinitesimal" numbers. To see this, let Σ be a first-order axiomatization of the theory of the real numbers. Consider the theory obtained by adding a new constant symbol ε to the language and adjoining to Σ the axiom $\varepsilon > 0$ and the axioms $\varepsilon < 1/n$ for all positive integers n . Clearly, the standard real numbers \mathbf{R} are a model for every finite subset of these axioms, because the real numbers satisfy everything in Σ and, by suitable choice of ε , can be made to satisfy any finite subset of the axioms about ε . By the compactness theorem, there is a model ${}^*\mathbf{R}$ that satisfies Σ and also contains an infinitesimal element ε . A similar argument, adjoining axioms $\omega > 0$, $\omega > 1$, etc., shows that the existence of infinitely large integers cannot be ruled out by any axiomatization Σ of the reals.

Proofs

One can prove the compactness theorem using Gödel's completeness theorem, which establishes that a set of sentences is satisfiable if and only if no contradiction can be proven from it. Since proofs are always finite and therefore involve only finitely many of the given sentences, the compactness theorem follows. In fact, the compactness theorem is equivalent to Gödel's completeness theorem, and both are equivalent to the Boolean prime ideal theorem, a weak form of the axiom of choice.^[4]

Gödel originally proved the compactness theorem in just this way, but later some "purely semantic" proofs of the compactness theorem were found, i.e., proofs that refer to *truth* but not to *provability*. One of those proofs relies on ultraproducts hinging on the axiom of choice as follows:

Proof: Fix a first-order language L , and let Σ be a collection of L -sentences such that every finite subcollection of L -sentences, $i \subseteq \Sigma$ of it has a model \mathcal{M}_i . Also let $\prod_{i \in \Sigma} \mathcal{M}_i$ be the direct product of the structures and I be the

collection of finite subsets of Σ . For each i in I let $A_i := \{ j \in I : j \supseteq i \}$. The family of all these sets A_i generates a filter, so there is an ultrafilter U containing all sets of the form A_i .

Now for any formula φ in Σ we have:

- the set $A_{\{\varphi\}}$ is in U
- whenever $j \in A_{\{\varphi\}}$, then $\varphi \in j$, hence φ holds in \mathcal{M}_j
- the set of all j with the property that φ holds in \mathcal{M}_j is a superset of $A_{\{\varphi\}}$, hence also in U

Using Łoś's theorem we see that φ holds in the ultraproduct $\prod_{i \in \Sigma} \mathcal{M}_i / U$. So this ultraproduct satisfies all formulas in Σ .

Notes

- [1] See Truss (1997).
- [2] Vaught, Robert L.: Alfred Tarski's work in model theory. *J. Symbolic Logic* 51 (1986), no. 4, 869–882.
- [3] Robinson, A.: Non-standard analysis. North-Holland Publishing Co., Amsterdam 1966. page 48.
- [4] See Hodges (1993).

References

- Boolos, George; Jeffrey, Richard; Burgess, John (2004). Computability and Logic (fourth ed.). "Cambridge University Press.
- Chang, C.C.; Keisler, H. Jerome (1989). *Model Theory* (third ed.). Elsevier. ISBN 0-7204-0692-7.
- Dawson, John W. junior (1993). "The compactness of first-order logic: From Gödel to Lindström". *History and Philosophy of Logic* 14: 15–37. doi: 10.1080/01445349308837208 (<http://dx.doi.org/10.1080/01445349308837208>).
- Hodges, Wilfrid (1993). *Model theory*. Cambridge University Press. ISBN 0-521-30442-3.
- Marker, David (2002). *Model Theory: An Introduction*. Graduate Texts in Mathematics 217. Springer. ISBN 0-387-98760-6.
- Truss, John K. (1997). *Foundations of Mathematical Analysis*. Oxford University Press. ISBN 0-19-853375-6.

Further reading

- Hummel, Christoph (1997). *Gromov's compactness theorem for pseudo-holomorphic curves*. Basel, Switzerland: Birkhäuser. ISBN 3-7643-5735-5.

Consensus theorem

Variable inputs			Function values	
x	y	z	$xy \vee \bar{x}z \vee yz$	$xy \vee \bar{x}z$
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	1

In Boolean algebra, the **consensus theorem** or **rule of consensus**^[1] is the identity:

$$xy \vee \bar{x}z \vee yz = xy \vee \bar{x}z$$

The **consensus** or **resolvent** of the terms xy and $\bar{x}z$ is yz . It is the conjunction of all the unique literals of the terms, excluding the literal which appears unnegated in one term and negated in the other.

The conjunctive dual of this equation is:

$$(x \vee y)(\bar{x} \vee z)(y \vee z) = (x \vee y)(\bar{x} \vee z)$$

Proof

$$\begin{aligned} \text{LHS} &= xy \vee \bar{x}z \vee (x \vee \bar{x})yz \\ &= xy \vee \bar{x}z \vee xyz \vee \bar{x}yz \\ &= xy \vee xyz \vee \bar{x}z \vee \bar{x}yz \\ &= xy(1 \vee z) \vee \bar{x}z(1 \vee y) \\ &= xy \vee \bar{x}z \\ &= \text{RHS} \end{aligned}$$

Consensus

The **consensus** or **consensus term** of two conjunctive terms of a disjunction is defined when one term contains the literal a and the other the literal \bar{a} , an **opposition**. The consensus is the conjunction of the two terms, omitting both a and \bar{a} , and repeated literals; the consensus is undefined if there is more than one opposition. For example, the consensus of $\bar{x}yz$ and $w\bar{y}z$ is $w\bar{x}z$.^[2]

The consensus can be derived from $(x \vee y)$ and $(\bar{x} \vee z)$ through the resolution inference rule. This shows that the LHS is derivable from the RHS (if $A \rightarrow B$ then $A \rightarrow AB$; replacing A with RHS and B with $(y \vee z)$). The RHS can be derived from the LHS simply through the conjunction elimination inference rule. Since RHS \rightarrow LHS and LHS \rightarrow RHS (in propositional calculus), then LHS = RHS (in Boolean algebra).

Digital logic circuitry

In digital logic, including the consensus term in a circuit can eliminate race hazards.

History

The concept of consensus was introduced by Archie Blake in 1937.^[3] It was rediscovered by Samson and Mills in 1954^[4] and by Quine in 1955.^[5] Quine coined the term 'consensus'. Robinson used it for clauses in 1965 as the basis of his "resolution principle".^{[6][7]}

Notes

- [1] Frank Markham Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd edition 2003, p. 44
- [2] Frank Markham Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd edition 2003, p. 81
- [3] "Canonical expressions in Boolean algebra", Dissertation, Dept. of Mathematics, U. of Chicago, 1937, reviewed in J. C. C. McKinsey, *The Journal of Symbolic Logic* 3:2:93 (June 1938)
- [4] Edward W. Samson, Burton E. Mills, Air Force Cambridge Research Center Technical Report 54-21, April 1954
- [5] W.V. Quine, "The problem of simplifying truth functions", *American Mathematical Monthly* 59:521-531, 1952
- [6] J. Alan Robinson, "A Machine-Oriented Logic Based on the Resolution Principle", *Journal of the ACM* 12:1: 23–41.
- [7] D.E. Knuth, *The Art of Computer Programming* 4A: *Combinatorial Algorithms*, part 1, p. 539

References

- Roth, Charles H. Jr. and Kinney, Larry L. (2004, 2010). "Fundamentals of Logic Design", 6th Ed., p. 66ff.

De Morgan's laws

Transformation rules
Propositional calculus

Rules of inference
• <i>Modus ponens</i>
• <i>Modus tollens</i>
• Biconditional introduction
• Biconditional elimination
• Conjunction introduction
• Simplification
• Disjunction introduction
• Disjunction elimination
• Disjunctive syllogism
• Hypothetical syllogism
• Constructive dilemma
• Destructive dilemma
• Absorption
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

In propositional logic and boolean algebra, **De Morgan's laws**^{[1][2][3]} 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:

$$\neg(P \wedge Q) \iff (\neg P) \vee (\neg Q)$$

$$\neg(P \vee Q) \iff (\neg P) \wedge (\neg Q)$$

where:

- \neg is the negation operator (NOT)
- \wedge is the conjunction operator (AND)
- \vee is the disjunction operator (OR)
- \iff 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:

$$\neg(P \wedge Q) \vdash (\neg P \vee \neg Q)$$

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

$$\neg(P \vee Q) \vdash (\neg P \wedge \neg Q)$$

In rule form: *negation of conjunction*

$$\frac{\neg(P \wedge Q)}{\therefore \neg P \vee \neg Q}$$

and *negation of disjunction*

$$\frac{\neg(P \vee Q)}{\therefore \neg P \wedge \neg Q}$$

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

$$\neg(P \wedge Q) \rightarrow (\neg P \vee \neg Q)$$

$$\neg(P \vee Q) \rightarrow (\neg P \wedge \neg Q)$$

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:

$$(P \wedge Q) \equiv \neg(\neg P \vee \neg Q)$$

$$(P \vee Q) \equiv \neg(\neg P \wedge \neg Q)$$

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",^[4] 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:

- 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:

$$\begin{aligned}\overline{\bigcap_{i \in I} A_i} &\equiv \bigcup_{i \in I} \overline{A_i} \\ \overline{\bigcup_{i \in I} A_i} &\equiv \bigcap_{i \in I} \overline{A_i}\end{aligned}$$

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".^[5]

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.

History

The law is named after Augustus De Morgan (1806–1871)^[6] 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^[7] (in the 14th century, William of Ockham wrote down the words that would result by reading the laws out),^[8] 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.^[9] 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:

$$\neg(A \vee B)$$

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:

$$(\neg A) \wedge (\neg B)$$

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:

$$\neg(A \wedge B)$$

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:

$$(\neg A) \vee (\neg B)$$

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 $x \notin A \cap B$. Because $A \cap B = \{y | y \in A \text{ and } y \in B\}$, then either $x \notin A$ or $x \notin B$. If $x \notin A$, then $x \in A^c$, so then $x \in A^c \cup B^c$. Otherwise, if $x \notin B$, 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 $x \notin (A \cap B)^c$. Then $x \in A \cap B$. It follows that $x \in A$ and $x \in B$. Then $x \notin A^c$ and $x \notin B^c$. But then $x \notin A^c \cup B^c$, 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 $P(p, q, \dots)$ depending on elementary propositions p, q, \dots to be the operator P^d defined by

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

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

$$\begin{aligned}\forall x P(x) &\equiv \neg \exists x \neg P(x), \\ \exists x P(x) &\equiv \neg \forall x \neg P(x).\end{aligned}$$

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

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

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] Copi and Cohen
- [2] Hurley
- [3] Moore and Parker
- [4] *Boolean Algebra* By R. L. Goodstein. ISBN 0-486-45894-6
- [5] 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
- [6] *DeMorgan's Theorems* (http://www.mtsu.edu/~phys2020/Lectures/L19-L25/L3/DeMorgan/body_demorgan.html) at mtsu.edu
- [7] Bocheński's *History of Formal Logic*
- [8] William of Ockham, Summa Logicae, part II, sections 32 & 33.
- [9] 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

Duality (order theory)

In the mathematical area of order theory, every partially ordered set P gives rise to a **dual** (or **opposite**) partially ordered set which is often denoted by P^{op} or P^d . This dual order P^{op} is defined to be the set with the **inverse order**, i.e. $x \leq y$ holds in P^{op} if and only if $y \leq x$ holds in P . It is easy to see that this construction, which can be depicted by flipping the Hasse diagram for P upside down, will indeed yield a partially ordered set. In a broader sense, two posets are also said to be duals if they are **dually isomorphic**, i.e. if one poset is order isomorphic to the dual of the other.

The importance of this simple definition stems from the fact that every definition and theorem of order theory can readily be transferred to the dual order. Formally, this is captured by the **Duality Principle** for ordered sets:

If a given statement is valid for all partially ordered sets, then its dual statement, obtained by inverting the direction of all order relations and by dualizing all order theoretic definitions involved, is also valid for all partially ordered sets.

If a statement or definition is equivalent to its dual then it is said to be **self-dual**. Note that the consideration of dual orders is so fundamental that it often occurs implicitly when writing \geq for the dual order of \leq without giving any prior definition of this "new" symbol.

Examples

Naturally, there are a great number of examples for concepts that are dual:

- Greatest elements and least elements
- Maximal elements and minimal elements
- Least upper bounds (suprema, \vee) and greatest lower bounds (infima, \wedge)
- Upper sets and lower sets
- Ideals and filters
- Closure operators and kernel operators.

Examples of notions which are self-dual include:

- Being a (complete) lattice
- Monotonicity of functions
- Distributivity of lattices, i.e. the lattices for which $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ holds are exactly those for which the dual statement $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ holds
- Being a Boolean algebra
- Being an order isomorphism.

Since partial orders are antisymmetric, the only ones that are self-dual are the equivalence relations.

References

- Davey, B.A.; Priestley, H. A. (2002), *Introduction to Lattices and Order* (2nd ed.), Cambridge University Press, ISBN 978-0-521-78451-1

Laws of classical logic

In mathematics and mathematical logic, **Boolean algebra** is the subarea of algebra in which the values of the variables are the truth values *true* and *false*, usually denoted 1 and 0 respectively. Instead of elementary algebra where the values of the variables are numbers, and the main operations are addition and multiplication, the main operations of Boolean algebra are the conjunction *and*, denoted \wedge , the disjunction *or*, denoted \vee , and the negation *not*, denoted \neg .

Boolean algebra was introduced in 1854 by George Boole in his book *An Investigation of the Laws of Thought*. According to Huntington the term "Boolean algebra" was first suggested by Sheffer in 1913.^[1]

Boolean algebra has been fundamental in the development of computer science and digital logic. It is also used in set theory and statistics.

History

Boole's algebra predicated the modern developments in abstract algebra and mathematical logic; it is however seen as connected to the origins of both fields. In an abstract setting, Boolean algebra was perfected in the late 19th century by Jevons, Schröder, Huntington, and others until it reached the modern conception of an (abstract) mathematical structure. For example, the empirical observation that one can manipulate expressions in the algebra of sets by translating them into expressions in Boole's algebra is explained in modern terms by saying that the algebra of sets is a Boolean algebra (note the indefinite article). In fact, M. H. Stone proved in 1936 that every Boolean algebra is isomorphic to a field of sets.

In the 1930s, while studying switching circuits, Claude Shannon observed that one could also apply the rules of Boole's algebra in this setting, and he introduced **switching algebra** as a way to analyze and design circuits by algebraic means in terms of logic gates. Shannon already had at his disposal the abstract mathematical apparatus, thus he cast his switching algebra as the two-element Boolean algebra. In circuit engineering settings today, there is little need to consider other Boolean algebras, thus "switching algebra" and "Boolean algebra" are often used interchangeably.^[2] Efficient implementation of Boolean functions is a fundamental problem in the design of combinatorial logic circuits. Modern electronic design automation tools for VLSI circuits often rely on an efficient representation of Boolean functions known as (reduced ordered) binary decision diagrams (BDD) for logic synthesis and formal verification.

Logic sentences that can be expressed in classical propositional calculus have an equivalent expression in Boolean algebra. Thus, **Boolean logic** is sometimes used to denote propositional calculus performed in this way. Boolean algebra is not sufficient to capture logic formulas using quantifiers, like those from first order logic. Although the development of mathematical logic did not follow Boole's program, the connection between his algebra and logic was later put on firm ground in the setting of algebraic logic, which also studies the algebraic systems of many other logics. The problem of determining whether the variables of a given Boolean (propositional) formula can be assigned in such a way as to make the formula evaluate to true is called the Boolean satisfiability problem (SAT), and is of importance to theoretical computer science, being the first problem shown to be NP-complete. The closely related model of computation known as a Boolean circuit relates time complexity (of an algorithm) to circuit complexity.

Values

Whereas in elementary algebra expressions denote mainly numbers, in Boolean algebra they denote the truth values *false* and *true*. These values are represented with the bits (or binary digits) being 0 and 1. They do not behave like the integers 0 and 1, for which $1 + 1 = 2$, but may be identified with the elements of the two-element field GF(2), for which $1 + 1 = 0$ with + serving as the Boolean operation XOR.

Boolean algebra also deals with functions which have their values in the set {0, 1}. A sequence of bits is a commonly used such function. Another common example is the subsets of a set E : to a subset F of E is associated the indicator function that takes the value 1 on F and 0 outside F .

As with elementary algebra, the purely equational part of the theory may be developed without considering explicit values for the variables.^[3]

Operations

Note: the truth table consisting of all 16 binary functions is located at [Boolean_algebras_canonically_defined#Truth_tables](#). There is also more information about these functions in [Truth_table](#).

Basic operations

The basic operations of Boolean algebra are the following ones:

- And (conjunction), denoted $x \wedge y$ (sometimes x AND y or K_{xy}), satisfies $x \wedge y = 1$ if $x = y = 1$ and $x \wedge y = 0$ otherwise.
- Or (disjunction), denoted $x \vee y$ (sometimes x OR y or A_{xy}), satisfies $x \vee y = 0$ if $x = y = 0$ and $x \vee y = 1$ otherwise.
- Not (negation), denoted $\neg x$ (sometimes NOT x , Nx or $!x$), satisfies $\neg x = 0$ if $x = 1$ and $\neg x = 1$ if $x = 0$.

If the truth values 0 and 1 are interpreted as integers, these operation may be expressed with the ordinary operations of the arithmetic:

$$x \wedge y = xy,$$

$$x \vee y = x + y - xy,$$

$$\neg x = 1 - x.$$

Alternatively the values of $x \wedge y$, $x \vee y$, and $\neg x$ can be expressed by tabulating their values with truth tables as follows.

x	y	$x \wedge y$	$x \vee y$
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	1

+ Figure 1. Truth tables

One may consider that only the negation and one of the two other operations are basic, because of the following identities that allow to define the conjunction in terms of the negation and the disjunction, and vice versa:

$$x \wedge y = \neg(\neg x \vee \neg y)$$

$$x \vee y = \neg(\neg x \wedge \neg y)$$

Derived operations

We have so far seen three Boolean operations. We referred to these as basic, meaning that they can be taken as a basis for other Boolean operations that can be built up from them by **composition**, the manner in which operations are combined or compounded. Here are some examples of operations composed from the basic operations.

$$\begin{aligned}x \rightarrow y &= (\neg x \vee y) \\x \oplus y &= (x \vee y) \wedge \neg(x \wedge y) \\x \equiv y &= \neg(x \oplus y)\end{aligned}$$

These definitions give rise to the following truth tables giving the values of these operations for all four possible inputs.

x	y	$x \rightarrow y$	$x \oplus y$	$x \equiv y$
0	0	1	0	1
1	0	0	1	0
0	1	1	1	0
1	1	1	0	1

The first operation, $x \rightarrow y$, or Cxy , is called **material implication**. If x is true then the value of $x \rightarrow y$ is taken to be that of y . But if x is false then we ignore the value of y ; however we must return *some* truth value and there are only two choices, so we choose the value that entails less, namely *true*. (Relevance logic addresses this by viewing an implication with a false premise as something other than either true or false.)

The second operation, $x \oplus y$, or Jxy , is called **exclusive or** to distinguish it from disjunction as the inclusive kind. It excludes the possibility of both x and y . Defined in terms of arithmetic it is addition mod 2 where $1 + 1 = 0$.

The third operation, the complement of exclusive or, is **equivalence** or Boolean equality: $x \equiv y$, or Exy , is true just when x and y have the same value. Hence $x \oplus y$ as its complement can be understood as $x \neq y$, being true just when x and y are different. Its counterpart in arithmetic mod 2 is $x + y + 1$.

Laws

A **law** of Boolean algebra is an identity such as $x \vee(y \vee z) = (x \vee y) \vee z$ between two Boolean terms, where a **Boolean term** is defined as an expression built up from variables and the constants 0 and 1 using the operations \wedge , \vee , and \neg . The concept can be extended to terms involving other Boolean operations such as \oplus , \rightarrow , and \equiv , but such extensions are unnecessary for the purposes to which the laws are put. Such purposes include the definition of a Boolean algebra as any model of the Boolean laws, and as a means for deriving new laws from old as in the derivation of $x \vee(y \wedge z) = x \vee(z \wedge y)$ from $y \wedge z = z \wedge y$ as treated in the section on axiomatization.

Monotone laws

Boolean algebra satisfies many of the same laws as ordinary algebra when we match up \vee with addition and \wedge with multiplication. In particular the following laws are common to both kinds of algebra:

(Associativity of \vee)	$x \vee (y \vee z) = (x \vee y) \vee z$
(Associativity of \wedge)	$x \wedge (y \wedge z) = (x \wedge y) \wedge z$
(Commutativity of \vee)	$x \vee y = y \vee x$
(Commutativity of \wedge)	$x \wedge y = y \wedge x$
(Distributivity of \wedge over \vee)	$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
(Identity for \vee)	$x \vee 0 = x$
(Identity for \wedge)	$x \wedge 1 = x$
(Annihilator for \wedge)	$x \wedge 0 = 0$

Boolean algebra however obeys some additional laws, in particular the following:

(Idempotence of \vee)	$x \vee x = x$
(Idempotence of \wedge)	$x \wedge x = x$
(Absorption 1)	$x \wedge (x \vee y) = x$
(Absorption 2)	$x \vee (x \wedge y) = x$
(Distributivity of \vee over \wedge)	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$
(Annihilator for \vee)	$x \vee 1 = 1$

A consequence of the first of these laws is $1 \vee 1 = 1$, which is false in ordinary algebra, where $1+1 = 2$. Taking $x = 2$ in the second law shows that it is not an ordinary algebra law either, since $2 \times 2 = 4$. The remaining four laws can be falsified in ordinary algebra by taking all variables to be 1, for example in Absorption Law 1 the left hand side is $1(1+1) = 2$ while the right hand side is 1, and so on.

All of the laws treated so far have been for conjunction and disjunction. These operations have the property that changing either argument either leaves the output unchanged or the output changes in the same way as the input. Equivalently, changing any variable from 0 to 1 never results in the output changing from 1 to 0. Operations with this property are said to be **monotone**. Thus the axioms so far have all been for monotonic Boolean logic. Nonmonotonicity enters via complement \neg as follows.

Nonmonotone laws

The complement operation is defined by the following two laws.

$$(Complementation 1) \quad x \wedge \neg x = 0$$

$$(Complementation 2) \quad x \vee \neg x = 1.$$

All properties of negation including the laws below follow from the above two laws alone.

In both ordinary and Boolean algebra, negation works by exchanging pairs of elements, whence in both algebras it satisfies the double negation law (also called involution law)

$$(Double \ negation) \quad \neg \neg x = x.$$

But whereas ordinary algebra satisfies the two laws

$$(\neg x)(\neg y) = xy$$

$$(\neg x) + (\neg y) = -(x + y),$$

Boolean algebra satisfies De Morgan's laws,

$$(\text{De Morgan 1}) \quad (\neg x) \wedge (\neg y) = \neg(x \vee y)$$

$$(\text{De Morgan 2}) \quad (\neg x) \vee (\neg y) = \neg(x \wedge y).$$

Completeness

At this point we can now claim to have defined Boolean algebra, in the sense that the laws we have listed up to now entail the rest of the subject. The laws *Complementation* 1 and 2, together with the monotone laws, suffice for this purpose and can therefore be taken as one possible *complete* set of laws or axiomatization of Boolean algebra. Every law of Boolean algebra follows logically from these axioms. Furthermore Boolean algebras can then be defined as the models of these axioms as treated in the section thereon.

To clarify, writing down further laws of Boolean algebra cannot give rise to any new consequences of these axioms, nor can it rule out any model of them. Had we stopped listing laws too soon, there would have been Boolean laws that did not follow from those on our list, and moreover there would have been models of the listed laws that were not Boolean algebras.

This axiomatization is by no means the only one, or even necessarily the most natural given that we did not pay attention to whether some of the axioms followed from others but simply chose to stop when we noticed we had enough laws, treated further in the section on axiomatizations. Or the intermediate notion of axiom can be sidestepped altogether by defining a Boolean law directly as any **tautology**, understood as an equation that holds for all values of its variables over 0 and 1. All these definitions of Boolean algebra can be shown to be equivalent.

Boolean algebra has the interesting property that $x = y$ can be proved from any non-tautology. This is because the substitution instance of any non-tautology obtained by instantiating its variables with constants 0 or 1 so as to witness its non-tautologyhood reduces by equational reasoning to $0 = 1$. For example the non-tautologyhood of $x \wedge y = x$ is witnessed by $x = 1$ and $y = 0$ and so taking this as an axiom would allow us to infer $1 \wedge 0 = 1$ as a substitution instance of the axiom and hence $0 = 1$. We can then show $x = y$ by the reasoning $x = x \wedge 1 = x \wedge 0 = 0 = 1 = y \vee 1 = y \vee 0 = y$.

Duality principle

There is nothing magical about the choice of symbols for the values of Boolean algebra. We could rename 0 and 1 to say α and β , and as long as we did so consistently throughout it would still be Boolean algebra, albeit with some obvious cosmetic differences.

But suppose we rename 0 and 1 to 1 and 0 respectively. Then it would still be Boolean algebra, and moreover operating on the same values. However it would not be identical to our original Boolean algebra because now we find \vee behaving the way \wedge used to do and vice versa. So there are still some cosmetic differences to show that we've been fiddling with the notation, despite the fact that we're still using 0s and 1s.

But if in addition to interchanging the names of the values we also interchange the names of the two binary operations, *now* there is no trace of what we have done. The end product is completely indistinguishable from what we started with. We might notice that the columns for $x \wedge y$ and $x \vee y$ in the truth tables had changed places, but that switch is immaterial.

When values and operations can be paired up in a way that leaves everything important unchanged when all pairs are switched simultaneously, we call the members of each pair **dual** to each other. Thus 0 and 1 are dual, and \wedge and \vee are dual. The **Duality Principle**, also called De Morgan duality, asserts that Boolean algebra is unchanged when all dual pairs are interchanged.

One change we did not need to make as part of this interchange was to complement. We say that complement is a **self-dual** operation. The identity or do-nothing operation x (copy the input to the output) is also self-dual. A more complicated example of a self-dual operation is $(x \wedge y) \vee (y \wedge z) \vee (z \wedge x)$. It can be shown that self-dual operations must

take an odd number of arguments; thus there can be no self-dual binary operation.

The principle of duality can be explained from a group theory perspective by fact that there are exactly four functions that are one-to-one mappings (automorphisms) of the set of Boolean polynomials back to itself: the identity function, the complement function, the dual function and the contradual function (complemented dual). These four functions form a group under function composition, isomorphic to the Klein four-group, acting on the set of Boolean polynomials.

Diagrammatic representations

Venn diagrams

A Venn diagram^[4] is a representation of a Boolean operation using shaded overlapping regions. There is one region for each variable, all circular in the examples here. The interior and exterior of region x corresponds respectively to the values 1 (true) and 0 (false) for variable x . The shading indicates the value of the operation for each combination of regions, with dark denoting 1 and light 0 (some authors use the opposite convention).

The three Venn diagrams in the figure below represent respectively conjunction $x \wedge y$, disjunction $x \vee y$, and complement $\neg x$.

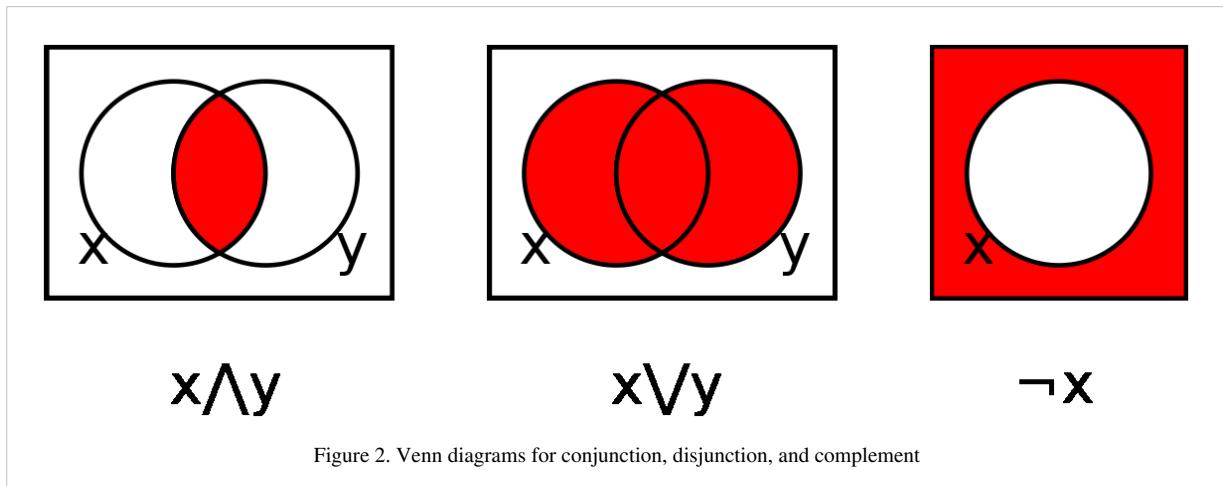


Figure 2. Venn diagrams for conjunction, disjunction, and complement

For conjunction, the region inside both circles is shaded to indicate that $x \wedge y$ is 1 when both variables are 1. The other regions are left unshaded to indicate that $x \wedge y$ is 0 for the other three combinations.

The second diagram represents disjunction $x \vee y$ by shading those regions that lie inside either or both circles. The third diagram represents complement $\neg x$ by shading the region *not* inside the circle.

While we have not shown the Venn diagrams for the constants 0 and 1, they are trivial, being respectively a white box and a dark box, neither one containing a circle. However we could put a circle for x in those boxes, in which case each would denote a function of one argument, x , which returns the same value independently of x , called a constant function. As far as their outputs are concerned, constants and constant functions are indistinguishable; the difference is that a constant takes no arguments, called a *zeroary* or *nullary* operation, while a constant function takes one argument, which it ignores, and is a *unary* operation.

Venn diagrams are helpful in visualizing laws. The commutativity laws for \wedge and \vee can be seen from the symmetry of the diagrams: a binary operation that was not commutative would not have a symmetric diagram because interchanging x and y would have the effect of reflecting the diagram horizontally and any failure of commutativity would then appear as a failure of symmetry.

Idempotence of \wedge and \vee can be visualized by sliding the two circles together and noting that the shaded area then becomes the whole circle, for both \wedge and \vee .

To see the first absorption law, $x \wedge (x \vee y) = x$, start with the diagram in the middle for $x \vee y$ and note that the portion of the shaded area in common with the x circle is the whole of the x circle. For the second absorption law, $x \vee (x \wedge y) = x$, start with the left diagram for $x \wedge y$ and note that shading the whole of the x circle results in just the x circle being shaded, since the previous shading was inside the x circle.

The double negation law can be seen by complementing the shading in the third diagram for $\neg x$, which shades the x circle.

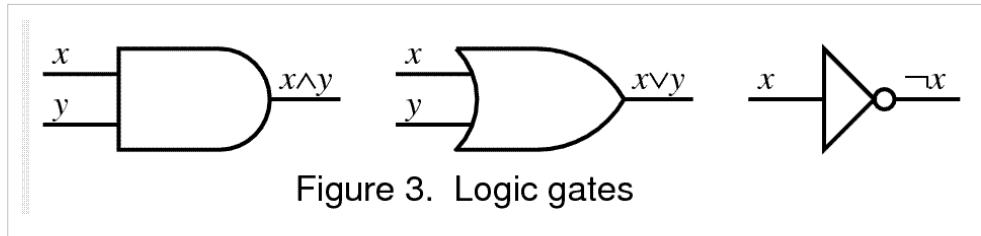
To visualize the first De Morgan's law, $(\neg x) \wedge (\neg y) = \neg(x \vee y)$, start with the middle diagram for $x \vee y$ and complement its shading so that only the region outside both circles is shaded, which is what the right hand side of the law describes. The result is the same as if we shaded that region which is both outside the x circle *and* outside the y circle, i.e. the conjunction of their exteriors, which is what the left hand side of the law describes.

The second De Morgan's law, $(\neg x) \vee (\neg y) = \neg(x \wedge y)$, works the same way with the two diagrams interchanged.

The first complement law, $x \wedge \neg x = 0$, says that the interior and exterior of the x circle have no overlap. The second complement law, $x \vee \neg x = 1$, says that everything is either inside or outside the x circle.

Digital logic gates

Digital logic is the application of the Boolean algebra of 0 and 1 to electronic hardware consisting of logic gates connected to form a circuit diagram. Each gate implements a Boolean operation, and is depicted schematically by a shape indicating the operation. The shapes associated with the gates for conjunction (AND-gates), disjunction (OR-gates), and complement (inverters) are as follows.

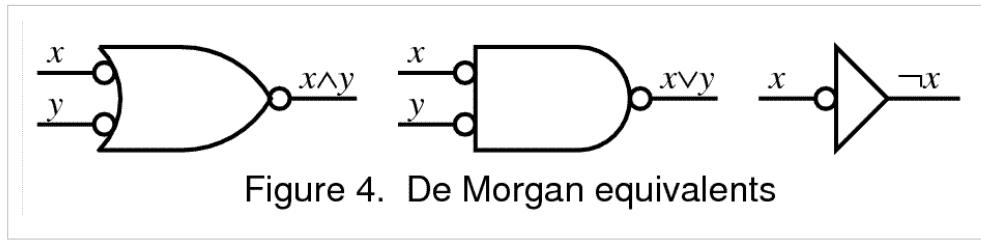


The lines on the left of each gate represent input wires or *ports*. The value of the input is represented by a voltage on the lead. For so-called "active-high" logic 0 is represented by a voltage close to zero or "ground" while 1 is represented by a voltage close to the supply voltage; active-low reverses this. The line on the right of each gate represents the output port, which normally follows the same voltage conventions as the input ports.

Complement is implemented with an inverter gate. The triangle denotes the operation that simply copies the input to the output; the small circle on the output denotes the actual inversion complementing the input. The convention of putting such a circle on any port means that the signal passing through this port is complemented on the way through, whether it is an input or output port.

There being eight ways of labeling the three ports of an AND-gate or OR-gate with inverters, this convention gives a wide range of possible Boolean operations realized as such gates so decorated. Not all combinations are distinct however: any labeling of AND-gate ports with inverters realizes the same Boolean operation as the opposite labeling of OR-gate ports (a given port of the AND-gate is labeled with an inverter if and only if the corresponding port of the OR-gate is not so labeled). This follows from De Morgan's laws.

If we complement all ports on every gate, and interchange AND-gates and OR-gates, as in Figure 4 below, we end up with the same operations as we started with, illustrating both De Morgan's laws and the Duality Principle. Note that we did not need to change the triangle part of the inverter, illustrating self-duality for complement.



Because of the pairwise identification of gates via the Duality Principle, even though 16 schematic symbols can be manufactured from the two basic binary gates AND and OR by furnishing their ports with inverters (circles), they only represent eight Boolean operations, namely those operations with an odd number of ones in their truth table. Altogether there are 16 binary Boolean operations, the other eight being those with an even number of ones in their truth table, namely the following. The constant 0, viewed as a binary operation that ignores both its inputs, has no ones, the six operations x , y , $\neg x$, $\neg y$ (as binary operations that ignore one input), $x \oplus y$, and $x \equiv y$ have two ones, and the constant 1 has four ones.

Boolean algebras

The term "algebra" denotes both a subject, namely the subject of algebra, and an object, namely an algebraic structure. Whereas the foregoing has addressed the subject of Boolean algebra, this section deals with mathematical objects called Boolean algebras, defined in full generality as any model of the Boolean laws. We begin with a special case of the notion definable without reference to the laws, namely concrete Boolean algebras, and then give the formal definition of the general notion.

Concrete Boolean algebras

A **concrete Boolean algebra** or field of sets is any nonempty set of subsets of a given set X closed under the set operations of union, intersection, and complement relative to X .

(As an aside, historically X itself was required to be nonempty as well to exclude the degenerate or one-element Boolean algebra, which is the one exception to the rule that all Boolean algebras satisfy the same equations since the degenerate algebra satisfies every equation. However this exclusion conflicts with the preferred purely equational definition of "Boolean algebra," there being no way to rule out the one-element algebra using only equations— $0 \neq 1$ does not count, being a negated equation. Hence modern authors allow the degenerate Boolean algebra and let X be empty.)

Example 1. The power set 2^X of X , consisting of all subsets of X . Here X may be any set: empty, finite, infinite, or even uncountable.

Example 2. The empty set and X . This two-element algebra shows that a concrete Boolean algebra can be finite even when it consists of subsets of an infinite set. It can be seen that every field of subsets of X must contain the empty set and X . Hence no smaller example is possible, other than the degenerate algebra obtained by taking X to be empty so as to make the empty set and X coincide.

Example 3. The set of finite and cofinite sets of integers, where a cofinite set is one omitting only finitely many integers. This is clearly closed under complement, and is closed under union because the union of a cofinite set with any set is cofinite, while the union of two finite sets is finite. Intersection behaves like union with "finite" and "cofinite" interchanged.

Example 4. For a less trivial example of the point made by Example 2, consider a Venn diagram formed by n closed curves partitioning the diagram into 2^n regions, and let X be the (infinite) set of all points in the plane not on any curve but somewhere within the diagram. The interior of each region is thus an infinite subset of X , and every point in X is in exactly one region. Then the set of all 2^n possible unions of regions (including the empty set obtained as the union of the empty set of regions and X obtained as the union of all 2^n regions) is closed under union,

intersection, and complement relative to X and therefore forms a concrete Boolean algebra. Again we have finitely many subsets of an infinite set forming a concrete Boolean algebra, with Example 2 arising as the case $n = 0$ of no curves.

Subsets as bit vectors

A subset Y of X can be identified with an indexed family of bits with index set X , with the bit indexed by $x \in X$ being 1 or 0 according to whether or not $x \in Y$. (This is the so-called characteristic function notion of a subset.) For example a 32-bit computer word consists of 32 bits indexed by the set $\{0,1,2,\dots,31\}$, with 0 and 31 indexing the low and high order bits respectively. For a smaller example, if $X = \{a,b,c\}$ where a, b, c are viewed as bit positions in that order from left to right, the eight subsets $\{\}, \{c\}, \{b\}, \{b,c\}, \{a\}, \{a,c\}, \{a,b\}$, and $\{a,b,c\}$ of X can be identified with the respective bit vectors 000, 001, 010, 011, 100, 101, 110, and 111. Bit vectors indexed by the set of natural numbers are infinite sequences of bits, while those indexed by the reals in the unit interval $[0,1]$ are packed too densely to be able to write conventionally but nonetheless form well-defined indexed families (imagine coloring every point of the interval $[0,1]$ either black or white independently; the black points then form an arbitrary subset of $[0,1]$).

From this bit vector viewpoint, a concrete Boolean algebra can be defined equivalently as a nonempty set of bit vectors all of the same length (more generally, indexed by the same set) and closed under the bit vector operations of bitwise \wedge , \vee , and \neg , as in $1010 \wedge 0110 = 0010$, $1010 \vee 0110 = 1110$, and $\neg 1010 = 0101$, the bit vector realizations of intersection, union, and complement respectively.

The prototypical Boolean algebra

The set $\{0,1\}$ and its Boolean operations as treated above can be understood as the special case of bit vectors of length one, which by the identification of bit vectors with subsets can also be understood as the two subsets of a one-element set. We call this the **prototypical** Boolean algebra, justified by the following observation.

The laws satisfied by all nondegenerate concrete Boolean algebras coincide with those satisfied by the prototypical Boolean algebra.

This observation is easily proved as follows. Certainly any law satisfied by all concrete Boolean algebras is satisfied by the prototypical one since it is concrete. Conversely any law that fails for some concrete Boolean algebra must have failed at a particular bit position, in which case that position by itself furnishes a one-bit counterexample to that law. Nondegeneracy ensures the existence of at least one bit position because there is only one empty bit vector.

The final goal of the next section can be understood as eliminating "concrete" from the above observation. We shall however reach that goal via the surprisingly stronger observation that, up to isomorphism, all Boolean algebras are concrete.

Boolean algebras: the definition

The Boolean algebras we have seen so far have all been concrete, consisting of bit vectors or equivalently of subsets of some set. Such a Boolean algebra consists of a set and operations on that set which can be *shown* to satisfy the laws of Boolean algebra.

Instead of showing that the Boolean laws are satisfied, we can instead postulate a set X , two binary operations on X , and one unary operation, and *require* that those operations satisfy the laws of Boolean algebra. The elements of X need not be bit vectors or subsets but can be anything at all. This leads to the more general *abstract* definition.

A Boolean algebra is any set with binary operations \wedge and \vee and a unary operation \neg thereon satisfying the Boolean laws.

For the purposes of this definition it is irrelevant how the operations came to satisfy the laws, whether by fiat or proof. All concrete Boolean algebras satisfy the laws (by proof rather than fiat), whence every concrete Boolean

algebra is a Boolean algebra according to our definitions. This axiomatic definition of a Boolean algebra as a set and certain operations satisfying certain laws or axioms *by fiat* is entirely analogous to the abstract definitions of group, ring, field etc. characteristic of modern or abstract algebra.

Given any complete axiomatization of Boolean algebra, such as the axioms for a complemented distributive lattice, a sufficient condition for an algebraic structure of this kind to satisfy all the Boolean laws is that it satisfy just those axioms. The following is therefore an equivalent definition.

A **Boolean algebra** is a complemented distributive lattice.

The section on axiomatization lists other axiomatizations, any of which can be made the basis of an equivalent definition.

Representable Boolean algebras

Although every concrete Boolean algebra is a Boolean algebra, not every Boolean algebra need be concrete. Let n be a square-free positive integer, one not divisible by the square of an integer, for example 30 but not 12. The operations of greatest common divisor, least common multiple, and division into n (that is, $\neg x = n/x$), can be shown to satisfy all the Boolean laws when their arguments range over the positive divisors of n . Hence those divisors form a Boolean algebra. These divisors are not subsets of a set, making the divisors of n a Boolean algebra that is not concrete according to our definitions.

However if we *represent* each divisor of n by the set of its prime factors, we find that this nonconcrete Boolean algebra is isomorphic to the concrete Boolean algebra consisting of all sets of prime factors of n , with union corresponding to least common multiple, intersection to greatest common divisor, and complement to division into n . So this example while not technically concrete is at least "morally" concrete via this representation, called an isomorphism. This example is an instance of the following notion.

A Boolean algebra is called **representable** when it is isomorphic to a concrete Boolean algebra.

The obvious next question is answered positively as follows.

Every Boolean algebra is representable.

That is, up to isomorphism, abstract and concrete Boolean algebras are the same thing. This quite nontrivial result depends on the Boolean prime ideal theorem, a choice principle slightly weaker than the axiom of choice, and is treated in more detail in the article Stone's representation theorem for Boolean algebras. This strong relationship implies a weaker result strengthening the observation in the previous subsection to the following easy consequence of representability.

The laws satisfied by all Boolean algebras coincide with those satisfied by the prototypical Boolean algebra.

It is weaker in the sense that it does not of itself imply representability. Boolean algebras are special here, for example a relation algebra is a Boolean algebra with additional structure but it is not the case that every relation algebra is representable in the sense appropriate to relation algebras.

Axiomatizing Boolean algebra

The above definition of an abstract Boolean algebra as a set and operations satisfying "the" Boolean laws raises the question, what are those laws? A simple-minded answer is "all Boolean laws," which can be defined as all equations that hold for the Boolean algebra of 0 and 1. Since there are infinitely many such laws this is not a terribly satisfactory answer in practice, leading to the next question: does it suffice to require only finitely many laws to hold?

In the case of Boolean algebras the answer is yes. In particular the finitely many equations we have listed above suffice. We say that Boolean algebra is **finitely axiomatizable** or **finitely based**.

Can this list be made shorter yet? Again the answer is yes. To begin with, some of the above laws are implied by some of the others. A sufficient subset of the above laws consists of the pairs of associativity, commutativity, and absorption laws, distributivity of \wedge over \vee (or the other distributivity law—one suffices), and the two complement laws. In fact this is the traditional axiomatization of Boolean algebra as a complemented distributive lattice.

By introducing additional laws not listed above it becomes possible to shorten the list yet further. In 1933 Edward Huntington showed that if the basic operations are taken to be $x \vee y$ and $\neg x$, with $x \wedge y$ considered a derived operation (e.g. via De Morgan's law in the form $x \wedge y = \neg(\neg x \vee \neg y)$), then the equation $\neg(\neg x \vee \neg y) \vee \neg(\neg x \vee y) = x$ along with the two equations expressing associativity and commutativity of \vee completely axiomatized Boolean algebra. When the only basic operation is the binary NAND operation $\neg(x \wedge y)$, Stephen Wolfram has proposed in his book *A New Kind of Science* the single axiom $((xy)z)(x((xz)x)) = z$ as a one-equation axiomatization of Boolean algebra, where for convenience here xy denotes the NAND rather than the AND of x and y .

Propositional logic

Propositional logic is a logical system that is intimately connected to Boolean algebra. Many syntactic concepts of Boolean algebra carry over to propositional logic with only minor changes in notation and terminology, while the semantics of propositional logic are defined via Boolean algebras in a way that the tautologies (theorems) of propositional logic correspond to equational theorems of Boolean algebra.

Syntactically, every Boolean term corresponds to a **propositional formula** of propositional logic. In this translation between Boolean algebra and propositional logic, Boolean variables x, y, \dots become **propositional variables** (or **atoms**) P, Q, \dots , Boolean terms such as $x \vee y$ become propositional formulas $P \vee Q$, 0 becomes *false* or \perp , and 1 becomes *true* or \top . It is convenient when referring to generic propositions to use Greek letters Φ, Ψ, \dots as metavariables (variables outside the language of propositional calculus, used when talking *about* propositional calculus) to denote propositions.

The semantics of propositional logic rely on **truth assignments**. The essential idea of a truth assignment is that the propositional variables are mapped to elements of a fixed Boolean algebra, and then the **truth value** of a propositional formula using these letters is the element of the Boolean algebra that is obtained by computing the value of the Boolean term corresponding to the formula. In classical semantics, only the two-element Boolean algebra is used, while in Boolean-valued semantics arbitrary Boolean algebras are considered. A **tautology** is a propositional formula that is assigned truth value 1 by every truth assignment of its propositional variables to an arbitrary Boolean algebra (or, equivalently, every truth assignment to the two element Boolean algebra).

These semantics permit a translation between tautologies of propositional logic and equational theorems of Boolean algebra. Every tautology Φ of propositional logic can be expressed as the Boolean equation $\Phi = 1$, which will be a theorem of Boolean algebra. Conversely every theorem $\Phi = \Psi$ of Boolean algebra corresponds to the tautologies $(\Phi \vee \neg \Psi) \wedge (\neg \Phi \vee \Psi)$ and $(\Phi \wedge \Psi) \vee (\neg \Phi \wedge \neg \Psi)$. If \rightarrow is in the language these last tautologies can also be written as $(\Phi \rightarrow \Psi) \wedge (\Psi \rightarrow \Phi)$, or as two separate theorems $\Phi \rightarrow \Psi$ and $\Psi \rightarrow \Phi$; if \equiv is available then the single tautology $\Phi \equiv \Psi$ can be used.

Applications

One motivating application of propositional calculus is the analysis of propositions and deductive arguments in natural language. Whereas the proposition "if $x = 3$ then $x+1 = 4$ " depends on the meanings of such symbols as + and 1, the proposition "if $x = 3$ then $x = 3$ " does not; it is true merely by virtue of its structure, and remains true whether " $x = 3$ " is replaced by " $x = 4$ " or "the moon is made of green cheese." The generic or abstract form of this tautology is "if P then P ", or in the language of Boolean algebra, " $P \rightarrow P$ ".

Replacing P by $x = 3$ or any other proposition is called **instantiation** of P by that proposition. The result of instantiating P in an abstract proposition is called an **instance** of the proposition. Thus " $x = 3 \rightarrow x = 3$ " is a tautology by virtue of being an instance of the abstract tautology " $P \rightarrow P$ ". All occurrences of the instantiated variable must be instantiated with the same proposition, to avoid such nonsense as $P \rightarrow x = 3$ or $x = 3 \rightarrow x = 4$.

Propositional calculus restricts attention to abstract propositions, those built up from propositional variables using Boolean operations. Instantiation is still possible within propositional calculus, but only by instantiating propositional variables by abstract propositions, such as instantiating Q by $Q \rightarrow P$ in $P \rightarrow (Q \rightarrow P)$ to yield the instance $P \rightarrow ((Q \rightarrow P) \rightarrow P)$.

(The availability of instantiation as part of the machinery of propositional calculus avoids the need for metavariables within the language of propositional calculus, since ordinary propositional variables can be considered within the language to denote arbitrary propositions. The metavariables themselves are outside the reach of instantiation, not being part of the language of propositional calculus but rather part of the same language for talking about it that this sentence is written in, where we need to be able to distinguish propositional variables and their instantiations as being distinct syntactic entities.)

Deductive systems for propositional logic

An axiomatization of propositional calculus is a set of tautologies called axioms and one or more inference rules for producing new tautologies from old. A *proof* in an axiom system A is a finite nonempty sequence of propositions each of which is either an instance of an axiom of A or follows by some rule of A from propositions appearing earlier in the proof (thereby disallowing circular reasoning). The last proposition is the **theorem** proved by the proof. Every nonempty initial segment of a proof is itself a proof, whence every proposition in a proof is itself a theorem. An axiomatization is **sound** when every theorem is a tautology, and **complete** when every tautology is a theorem.

Sequent calculus

Propositional calculus is commonly organized as a Hilbert system, whose operations are just those of Boolean algebra and whose theorems are Boolean tautologies, those Boolean terms equal to the Boolean constant 1. Another form is sequent calculus, which has two sorts, propositions as in ordinary propositional calculus, and pairs of lists of propositions called sequents, such as $A \vee B, A \wedge C, \dots \vdash A, B \rightarrow C, \dots$. The two halves of a sequent are called the antecedent and the succedent respectively. The customary metavariable denoting an antecedent or part thereof is Γ , and for a succedent Δ ; thus $\Gamma, A \vdash \Delta$ would denote a sequent whose succedent is a list Δ and whose antecedent is a list Γ with an additional proposition A appended after it. The antecedent is interpreted as the conjunction of its propositions, the succedent as the disjunction of its propositions, and the sequent itself as the entailment of the succedent by the antecedent.

Entailment differs from implication in that whereas the latter is a binary *operation* that returns a value in a Boolean algebra, the former is a binary *relation* which either holds or does not hold. In this sense entailment is an *external* form of implication, meaning external to the Boolean algebra, thinking of the reader of the sequent as also being external and interpreting and comparing antecedents and succedents in some Boolean algebra. The natural interpretation of \vdash is as \leq in the partial order of the Boolean algebra defined by $x \leq y$ just when $x \vee y = y$. This ability to mix external implication \vdash and internal implication \rightarrow in the one logic is among the essential differences between sequent calculus and propositional calculus.

Applications

Two-valued logic

Boolean algebra as the calculus of two values is fundamental to digital logic, computer programming, and mathematical logic, and is also used in other areas of mathematics such as set theory and statistics.

Digital logic codes its symbols in various ways: as voltages on wires in high-speed circuits and capacitive storage devices, as orientations of a magnetic domain in ferromagnetic storage devices, as holes in punched cards or paper tape, and so on. Now it is possible to code more than two symbols in any given medium. For example one might use respectively 0, 1, 2, and 3 volts to code a four-symbol alphabet on a wire, or holes of different sizes in a punched card. In practice however the tight constraints of high speed, small size, and low power combine to make noise a major factor. This makes it hard to distinguish between symbols when there are many of them at a single site. Rather than attempting to distinguish between four voltages on one wire, digital designers have settled on two voltages per wire, high and low. To obtain four symbols one uses two wires, and so on.

Programmers programming in machine code, assembly language, and other programming languages that expose the low-level digital structure of the data registers operate on whatever symbols were chosen for the hardware, invariably bit vectors in modern computers for the above reasons. Such languages support both the numeric operations of addition, multiplication, etc. performed on words interpreted as integers, as well as the logical operations of disjunction, conjunction, etc. performed bit-wise on words interpreted as bit vectors. Programmers therefore have the option of working in and applying the laws of either numeric algebra or Boolean algebra as needed. A core differentiating feature is carry propagation with the former but not the latter.

Other areas where two values is a good choice are the law and mathematics. In everyday relaxed conversation, nuanced or complex answers such as "maybe" or "only on the weekend" are acceptable. In more focused situations such as a court of law or theorem-based mathematics however it is deemed advantageous to frame questions so as to admit a simple yes-or-no answer—is the defendant guilty or not guilty, is the proposition true or false—and to disallow any other answer. However much of a straitjacket this might prove in practice for the respondent, the principle of the simple yes-no question has become a central feature of both judicial and mathematical logic, making two-valued logic deserving of organization and study in its own right.

A central concept of set theory is membership. Now an organization may permit multiple degrees of membership, such as novice, associate, and full. With sets however an element is either in or out. The candidates for membership in a set work just like the wires in a digital computer: each candidate is either a member or a nonmember, just as each wire is either high or low.

Algebra being a fundamental tool in any area amenable to mathematical treatment, these considerations combine to make the algebra of two values of fundamental importance to computer hardware, mathematical logic, and set theory.

Two-valued logic can be extended to multi-valued logic, notably by replacing the Boolean domain $\{0, 1\}$ with the unit interval $[0,1]$, in which case rather than only taking values 0 or 1, any value between and including 0 and 1 can be assumed. Algebraically, negation (NOT) is replaced with $1 - x$, conjunction (AND) is replaced with multiplication (xy), and disjunction (OR) is defined via De Morgan's law. Interpreting these values as logical truth values yields a multi-valued logic, which forms the basis for fuzzy logic and probabilistic logic. In these interpretations, a value is interpreted as the "degree" of truth – to what extent a proposition is true, or the probability that the proposition is true.

Boolean operations

The original application for Boolean operations was mathematical logic, where it combines the truth values, true or false, of individual formulas.

Natural languages such as English have words for several Boolean operations, in particular conjunction (*and*), disjunction (*or*), negation (*not*), and implication (*implies*). *But not* is synonymous with *and not*. When used to combine situational assertions such as "the block is on the table" and "cats drink milk," which naively are either true or false, the meanings of these logical connectives often have the meaning of their logical counterparts. However with descriptions of behavior such as "Jim walked through the door", one starts to notice differences such as failure of commutativity, for example the conjunction of "Jim opened the door" with "Jim walked through the door" in that order is not equivalent to their conjunction in the other order, since *and* usually means *and then* in such cases. Questions can be similar: the order "Is the sky blue, and why is the sky blue?" makes more sense than the reverse order. Conjunctive commands about behavior are like behavioral assertions, as in *get dressed and go to school*. Disjunctive commands such *love me or leave me* or *fish or cut bait* tend to be asymmetric via the implication that one alternative is less preferable. Conjoined nouns such as *tea and milk* generally describe aggregation as with set union while *tea or milk* is a choice. However context can reverse these senses, as in *your choices are coffee and tea* which usually means the same as *your choices are coffee or tea* (alternatives). Double negation as in "I don't not like milk" rarely means literally "I do like milk" but rather conveys some sort of hedging, as though to imply that there is a third possibility. "Not not P" can be loosely interpreted as "surely P", and although *P* necessarily implies "not not *P*" the converse is suspect in English, much as with intuitionistic logic. In view of the highly idiosyncratic usage of conjunctions in natural languages, Boolean algebra cannot be considered a reliable framework for interpreting them.

Boolean operations are used in digital logic to combine the bits carried on individual wires, thereby interpreting them over {0,1}. When a vector of n identical binary gates are used to combine two bit vectors each of n bits, the individual bit operations can be understood collectively as a single operation on values from a Boolean algebra with 2^n elements.

Naive set theory interprets Boolean operations as acting on subsets of a given set X . As we saw earlier this behavior exactly parallels the coordinate-wise combinations of bit vectors, with the union of two sets corresponding to the disjunction of two bit vectors and so on.

The 256-element free Boolean algebra on three generators is deployed in computer displays based on raster graphics, which use bit blit to manipulate whole regions consisting of pixels, relying on Boolean operations to specify how the source region should be combined with the destination, typically with the help of a third region called the mask. Modern video cards offer all $2^3 = 256$ ternary operations for this purpose, with the choice of operation being a one-byte (8-bit) parameter. The constants SRC = 0xaa or 10101010, DST = 0xcc or 11001100, and MSK = 0xf0 or 11110000 allow Boolean operations such as (SRC \wedge DST)&MSK (meaning XOR the source and destination and then AND the result with the mask) to be written directly as a constant denoting a byte calculated at compile time, 0x60 in the (SRC \wedge DST)&MSK example, 0x66 if just SRC \wedge DST, etc. At run time the video card interprets the byte as the raster operation indicated by the original expression in a uniform way that requires remarkably little hardware and which takes time completely independent of the complexity of the expression.

Solid modeling systems for computer aided design offer a variety of methods for building objects from other objects, combination by Boolean operations being one of them. In this method the space in which objects exist is understood as a set S of voxels (the three-dimensional analogue of pixels in two-dimensional graphics) and shapes are defined as subsets of S , allowing objects to be combined as sets via union, intersection, etc. One obvious use is in building a complex shape from simple shapes simply as the union of the latter. Another use is in sculpting understood as removal of material: any grinding, milling, routing, or drilling operation that can be performed with physical machinery on physical materials can be simulated on the computer with the Boolean operation $x \wedge \neg y$ or $x - y$, which in set theory is set difference, remove the elements of y from those of x . Thus given two shapes one to be machined and the other the material to be removed, the result of machining the former to remove the latter is described simply

as their set difference.

Boolean searches

Search engine queries also employ Boolean logic. For this application, each web page on the Internet may be considered to be an "element" of a "set". The following examples use a syntax supported by Google.^[5]

- Doublequotes are used to combine whitespace-separated words into a single search term.^[6]
- Whitespace is used to specify logical AND, as it is the default operator for joining search terms:

```
"Search term 1" "Search term 2"
```

- The OR keyword is used for logical OR:

```
"Search term 1" OR "Search term 2"
```

- The minus sign is used for logical NOT (AND NOT):

```
"Search term 1" - "Search term 2"
```

References

- [1] cf footnote on page 278: "* The name Boolean algebra (or Boolean "algebras") for the calculus originated by Boole, extended by Schröder, and perfected by Whitehead seems to have been first suggested by Sheffer, in 1913" quoted from E. V. Huntington January 1933, "NEW SETS OF INDEPENDENT POSTULATES FOR THE ALGEBRA OF LOGIC, WITH SPECIAL REFERENCE TO WHITEHEAD AND RUSSELL'S PRINCIPIA MATHEMATICA", [http://www.ams.org/journals/tran/1933-035-01/S0002-9947-1933-1501684-X.pdf](http://www.ams.org/journals/tran/1933-035-01/S0002-9947-1933-1501684-X/S0002-9947-1933-1501684-X.pdf)
- [2] , online sample (<http://www.wiley.com/college/engin/balabanian293512/pdf/ch02.pdf>)
- [3] Halmos, Paul (1963). Lectures on Boolean Algebras. van Nostrand.
- [4] J. Venn, *On the Diagrammatic and Mechanical Representation of Propositions and Reasonings*, Philosophical Magazine and Journal of Science, Series 5, vol. **10**, No. 59, July 1880.
- [5] Not all search engines support the same query syntax. Additionally, some organizations (such as Google) provide "specialized" search engines that support alternate or extended syntax. (See e.g., Syntax cheatsheet (<http://www.google.com/help/cheatsheet.html>), Google codesearch supports regular expressions (http://www.google.com/intl/en/help/faq_codesearch.html#regexp)).
- [6] Doublequote-delimited search terms are called "exact phrase" searches in the Google documentation.

Mano, Morris; Ciletti, Michael D. (2013). *Digital Design*. Pearson. ISBN 978-0-13-277420-8.

Further reading

- J. Eldon Whitesitt (1995). *Boolean algebra and its applications*. Courier Dover Publications. ISBN 978-0-486-68483-3. Suitable introduction for students in applied fields.
- Dwinger, Philip (1971). *Introduction to Boolean algebras*. Würzburg: Physica Verlag.
- Sikorski, Roman (1969). *Boolean Algebras* (3/e ed.). Berlin: Springer-Verlag. ISBN 978-0-387-04469-9.
- Bocheński, Józef Maria (1959). *A Précis of Mathematical Logic*. Translated from the French and German editions by Otto Bird. Dordrecht, South Holland: D. Reidel.

Historical perspective

- George Boole (1848). " The Calculus of Logic, (<http://www.maths.tcd.ie/pub/HistMath/People/Boole/CalcLogic/CalcLogic.html>)" *Cambridge and Dublin Mathematical Journal III: 183–98*.
- Theodore Hailperin (1986). *Boole's logic and probability: a critical exposition from the standpoint of contemporary algebra, logic, and probability theory* (2nd ed.). Elsevier. ISBN 978-0-444-87952-3.
- Dov M. Gabbay, John Woods, ed. (2004). *The rise of modern logic: from Leibniz to Frege*. Handbook of the History of Logic **3**. Elsevier. ISBN 978-0-444-51611-4., several relevant chapters by Hailperin, Valencia, and Grattan-Guinesss
- Calixto Badesa (2004). *The birth of model theory: Löwenheim's theorem in the frame of the theory of relatives*. Princeton University Press. ISBN 978-0-691-05853-5., chapter 1, "Algebra of Classes and Propositional

Calculus"

- Burris, Stanley, 2009. The Algebra of Logic Tradition (<http://plato.stanford.edu/entries/algebra-logic-tradition/>). Stanford Encyclopedia of Philosophy.
- Radomir S. Stankovic; Jaakko Astola (2011). *From Boolean Logic to Switching Circuits and Automata: Towards Modern Information Technology* (<http://books.google.com/books?id=uagvEc2jGTIC>). Springer. ISBN 978-3-642-11681-0.

External links

- How Stuff Works – Boolean Logic (<http://computer.howstuffworks.com/boolean.htm>)
- Science and Technology - Boolean Algebra (<http://oscience.info/mathematics/boolean-algebra-2/>) contains a list and proof of Boolean theorems and laws.

Peirce's law

In logic, **Peirce's law** is named after the philosopher and logician Charles Sanders Peirce. It was taken as an axiom in his first axiomatisation of propositional logic. It can be thought of as the law of excluded middle written in a form that involves only one sort of connective, namely implication.

In propositional calculus, **Peirce's law** says that $((P \rightarrow Q) \rightarrow P) \rightarrow P$. Written out, this means that P must be true if there is a proposition Q such that the truth of P follows from the truth of "if P then Q ". In particular, when Q is taken to be a false formula, the law says that if P must be true whenever it implies the false, then P is true. In this way Peirce's law implies the law of excluded middle.

Peirce's law does not hold in intuitionistic logic or intermediate logics and cannot be deduced from the deduction theorem alone.

Under the Curry–Howard isomorphism, Peirce's law is the type of continuation operators, e.g. call/cc in Scheme.^[1]

History

Here is Peirce's own statement of the law:

A *fifth icon* is required for the principle of excluded middle and other propositions connected with it. One of the simplest formulae of this kind is:

$$\{(x \rightarrow y) \rightarrow x\} \rightarrow x.$$

This is hardly axiomatical. That it is true appears as follows. It can only be false by the final consequent x being false while its antecedent $(x \rightarrow y) \rightarrow x$ is true. If this is true, either its consequent, x , is true, when the whole formula would be true, or its antecedent $x \rightarrow y$ is false. But in the last case the antecedent of $x \rightarrow y$, that is x , must be true. (Peirce, the *Collected Papers* 3.384).

Peirce goes on to point out an immediate application of the law:

From the formula just given, we at once get:

$$\{(x \rightarrow y) \rightarrow a\} \rightarrow x,$$

where the a is used in such a sense that $(x \rightarrow y) \rightarrow a$ means that from $(x \rightarrow y)$ every proposition follows. With that understanding, the formula states the principle of excluded middle, that from the falsity of the denial of x follows the truth of x . (Peirce, the *Collected Papers* 3.384).

Warning: $((x \rightarrow y) \rightarrow a) \rightarrow x$ is *not* a tautology. However, $[a \rightarrow x] \rightarrow [((x \rightarrow y) \rightarrow a) \rightarrow x]$ is a tautology.

Other proofs of Peirce's law

Showing Peirce's Law applies does not mean that $P \rightarrow Q$ or Q is true, we have that P is true but only $(P \rightarrow Q) \rightarrow P$, not $P \rightarrow (P \rightarrow Q)$ (see affirming the consequent).

$$\text{simple} \qquad \qquad \qquad \text{proof:}$$

$$(p \rightarrow q) \rightarrow p \Rightarrow \overline{p \rightarrow q} \vee p \Rightarrow \overline{\overline{p} \vee q} \vee p \Rightarrow (p \wedge \overline{q}) \vee p \Rightarrow (p \wedge \overline{q}) \vee (p \wedge 1) \Rightarrow p \wedge (\overline{q} \vee 1) \Rightarrow p \wedge 1 \Rightarrow p.$$

Using Peirce's law with the deduction theorem

Peirce's law allows one to enhance the technique of using the deduction theorem to prove theorems. Suppose one is given a set of premises Γ and one wants to deduce a proposition Z from them. With Peirce's law, one can add (at no cost) additional premises of the form $Z \rightarrow P$ to Γ . For example, suppose we are given $P \rightarrow Z$ and $(P \rightarrow Q) \rightarrow Z$ and we wish to deduce Z so that we can use the deduction theorem to conclude that $(P \rightarrow Z) \rightarrow ((P \rightarrow Q) \rightarrow Z) \rightarrow Z$ is a theorem. Then we can add another premise $Z \rightarrow Q$. From that and $P \rightarrow Z$, we get $P \rightarrow Q$. Then we apply modus ponens with $(P \rightarrow Q) \rightarrow Z$ as the major premise to get Z . Applying the deduction theorem, we get that $(Z \rightarrow Q) \rightarrow Z$ follows from the original premises. Then we use Peirce's law in the form $((Z \rightarrow Q) \rightarrow Z) \rightarrow Z$ and modus ponens to derive Z from the original premises. Then we can finish off proving the theorem as we originally intended.

- $P \rightarrow Z$ 1. hypothesis
- $(P \rightarrow Q) \rightarrow Z$ 2. hypothesis
- $Z \rightarrow Q$ 3. hypothesis
 - P 4. hypothesis
 - Z 5. modus ponens using steps 4 and 1
 - Q 6. modus ponens using steps 5 and 3
- $P \rightarrow Q$ 7. deduction from 4 to 6
- Z 8. modus ponens using steps 7 and 2
- $(Z \rightarrow Q) \rightarrow Z$ 9. deduction from 3 to 8
- $((Z \rightarrow Q) \rightarrow Z) \rightarrow Z$ 10. Peirce's law
- Z 11. modus ponens using steps 9 and 10
- $((P \rightarrow Q) \rightarrow Z) \rightarrow Z$ 12. deduction from 2 to 11
- $(P \rightarrow Z) \rightarrow ((P \rightarrow Q) \rightarrow Z) \rightarrow Z$ 13. deduction from 1 to 12 QED

Completeness of the implicational propositional calculus

One reason that Peirce's law is important is that it can substitute for the law of excluded middle in the logic which only uses implication. The sentences which can be deduced from the axiom schemas:

- $P \rightarrow (Q \rightarrow P)$
- $(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$
- $((P \rightarrow Q) \rightarrow P) \rightarrow P$
- from P and $P \rightarrow Q$ infer Q

(where P, Q, R contain only " \rightarrow " as a connective) are all the tautologies which use only " \rightarrow " as a connective.

Notes

- [1] A Formulae-as-Types Notion of Control (<http://citeseer.ist.psu.edu/griffin90formulaeasatypes.html>) - Griffin defines K on page 3 as an equivalent to Scheme's call/cc and then discusses its type being the equivalent of Peirce's law at the end of section 5 on page 9.

References

- Peirce, C.S., "On the Algebra of Logic: A Contribution to the Philosophy of Notation", *American Journal of Mathematics* 7, 180–202 (1885). Reprinted, the *Collected Papers of Charles Sanders Peirce* 3.359–403 and the *Writings of Charles S. Peirce: A Chronological Edition* 5, 162–190.
- Peirce, C.S., *Collected Papers of Charles Sanders Peirce*, Vols. 1–6, Charles Hartshorne and Paul Weiss (eds.), Vols. 7–8, Arthur W. Burks (ed.), Harvard University Press, Cambridge, MA, 1931–1935, 1958.

Poretsky's law of forms

In Boolean algebra, **Poretsky's law of forms** shows that the single Boolean equation $f(X) = 0$ is equivalent to $g(X) = h(X)$ if and only if $g = f \oplus h$, where \oplus represents exclusive or.

The law of forms was discovered by Platon Poretsky.

References

- Frank Markham Brown, *Boolean Reasoning: The Logic of Boolean Equations*, 2nd edition, 2003, p. 100

Stone's representation theorem for Boolean algebras

In mathematics, **Stone's representation theorem for Boolean algebras** states that every Boolean algebra is isomorphic to a field of sets. The theorem is fundamental to the deeper understanding of Boolean algebra that emerged in the first half of the 20th century. The theorem was first proved by Stone (1936), and thus named in his honor. Stone was led to it by his study of the spectral theory of operators on a Hilbert space.

Stone spaces

Each Boolean algebra B has an associated topological space, denoted here $S(B)$, called its **Stone space**. The points in $S(B)$ are the ultrafilters on B , or equivalently the homomorphisms from B to the two-element Boolean algebra. The topology on $S(B)$ is generated by a (closed) basis consisting of all sets of the form

$$\{x \in S(B) \mid b \in x\},$$

where b is an element of B .

For every Boolean algebra B , $S(B)$ is a compact totally disconnected Hausdorff space; such spaces are called **Stone spaces** (also *profinite spaces*). Conversely, given any topological space X , the collection of subsets of X that are clopen (both closed and open) is a Boolean algebra.

Representation theorem

A simple version of **Stone's representation theorem** states that every Boolean algebra B is isomorphic to the algebra of clopen subsets of its Stone space $S(B)$. The isomorphism sends an element $b \in B$ to the set of all ultrafilters that contain b . This is a clopen set because of the choice of topology on $S(B)$ and because B is a Boolean algebra.

Restating the theorem using the language of category theory; the theorem states that there is a duality between the category of Boolean algebras and the category of Stone spaces. This duality means that in addition to the isomorphisms between Boolean algebras and their Stone spaces, each homomorphism from a Boolean algebra A to a Boolean algebra B corresponds in a natural way to a continuous function from $S(B)$ to $S(A)$. In other words, there is a contravariant functor that gives an equivalence between the categories. This was an early example of a nontrivial duality of categories.

The theorem is a special case of Stone duality, a more general framework for dualities between topological spaces and partially ordered sets.

The proof requires either the axiom of choice or a weakened form of it. Specifically, the theorem is equivalent to the Boolean prime ideal theorem, a weakened choice principle which states that every Boolean algebra has a prime ideal.

References

- Paul Halmos, and Givant, Steven (1998) *Logic as Algebra*. Dolciani Mathematical Expositions No. 21. The Mathematical Association of America.
- Johnstone, Peter T. (1982) *Stone Spaces*. Cambridge University Press. ISBN 0-521-23893-5.
- Marshall H. Stone (1936) "The Theory of Representations of Boolean Algebras,"^[1] *Transactions of the American Mathematical Society* 40: 37-111.

A monograph available free online:

- Burris, Stanley N., and H.P. Sankappanavar, H. P.(1981) *A Course in Universal Algebra*.^[1] Springer-Verlag. ISBN 3-540-90578-2.

References

[1] <http://links.jstor.org/sici?doi=0002-9947%28193607%2940%3A1%3C37%3ATTORFB%3E2.0.CO%3B2-8>

Philosophy

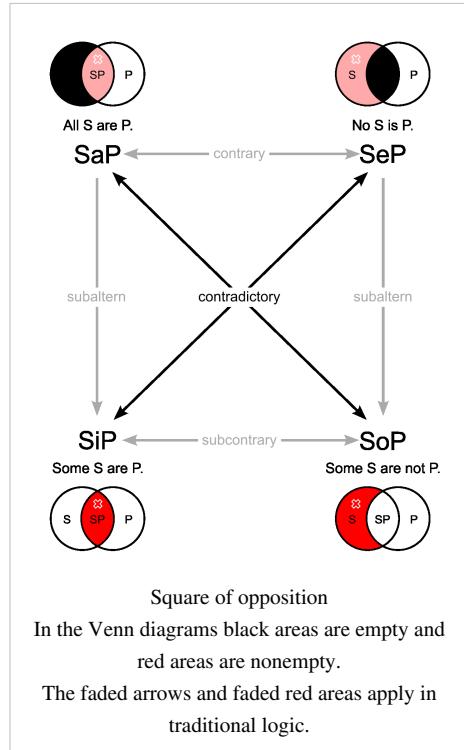
Boole's syllogistic

Boolean logic is a system of syllogistic logic invented by 19th-century British mathematician George Boole, which attempts to incorporate the "empty set", that is, a class of non-existent entities, such as round squares, without resorting to uncertain truth values.

In Boolean logic, the universal statements "all S is P" and "no S is P" (contraries in the traditional Aristotelian schema) are composable provided that the set of "S" is the empty set. "All S is P" is construed to mean that "there is nothing that is both S and not-P"; "no S is P", that "there is nothing that is both S and P". For example, since there is nothing that is a round square, it is true both that nothing is a round square and purple, and that nothing is a round square and *not*-purple. Therefore, both universal statements, that "all round squares are purple" and "no round squares are purple" are true.

Similarly, the subcontrary relationship is dissolved between the existential statements "some S is P" and "some S is not P". The former is interpreted as "there is some S such that S is P" and the latter, "there is some S such that S is not P", both of which are clearly false where S is nonexistent.

Thus, the subaltern relationship between universal and existential also does not hold, since for a nonexistent S, "All S is P" is true but does not entail "Some S is P", which is false. Of the Aristotelian square of opposition, only the contradictory relationships remain intact.



Entitative graph

An **entitative graph** is an element of the diagrammatic syntax for logic that Charles Sanders Peirce developed under the name of **qualitative logic** beginning in the 1880s, taking the coverage of the formalism only as far as the propositional or sentential aspects of logic are concerned. See 3.468, 4.434, and 4.564 in Peirce's *Collected Papers*.

The syntax is:

- The blank page;
- Single letters, phrases;
- Objects (subgraphs) enclosed by a simple closed curve called a *cut*. A cut can be empty.

The semantics are:

- The blank page denotes **False**;
- Letters, phrases, subgraphs, and entire graphs can be **True** or **False**;
- To surround objects with a cut is equivalent to Boolean complementation. Hence an empty cut denotes **Truth**;
- All objects within a given cut are tacitly joined by disjunction.

A "proof" manipulates a graph, using a short list of rules, until the graph is reduced to an empty cut or the blank page. A graph that can be so reduced is what is now called a tautology (or the complement thereof). Graphs that cannot be simplified beyond a certain point are analogues of the satisfiable formulas of first-order logic.

Peirce soon abandoned the entitative graphs for the existential graphs, whose sentential (*alpha*) part is dual to the entitative graphs. He developed the existential graphs until they became another formalism for what are now termed first-order logic and normal modal logic.

The primary algebra of G. Spencer-Brown is isomorphic to the entitative graphs.

References

- Peirce, C.S., *Collected Papers of Charles Sanders Peirce*, Vols. 1–6, Charles Hartshorne and Paul Weiss (eds.), Vols. 7–8, Arthur W. Burks, ed., Harvard University Press, Cambridge, MA, 1931–1935, 1958. Cited as CP volume.paragraph.
- Peirce, C.S., "Qualitative Logic", MS 736 (c. 1886), pp. 101–115 in *The New Elements of Mathematics by Charles S. Peirce, Volume 4, Mathematical Philosophy*, Carolyn Eisele (ed.), Mouton, The Hague, 1976.
- Peirce, C.S., "Qualitative Logic", MS 582 (1886), pp. 323–371 in *Writings of Charles S. Peirce: A Chronological Edition, Volume 5, 1884–1886*, Peirce Edition Project (eds.), Indiana University Press, Bloomington, IN, 1993.
- Peirce, C.S., "The Logic of Relatives: Qualitative and Quantitative", MS 584 (1886), pp. 372–378 in *Writings of Charles S. Peirce: A Chronological Edition, Volume 5, 1884–1886*, Peirce Edition Project (eds.), Indiana University Press, Bloomington, IN, 1993.
- Shin, Sun-Joo (2002), *The Iconic Logic of Peirce's Graphs*, MIT Press, Cambridge, MA.

Existential graph

An **existential graph** is a type of diagrammatic or visual notation for logical expressions, proposed by Charles Sanders Peirce, who wrote on graphical logic as early as 1882,^[1] and continued to develop the method until his death in 1914.

The graphs

Peirce proposed three systems of existential graphs:

- *alpha*, isomorphic to sentential logic and the two-element Boolean algebra;
- *beta*, isomorphic to first-order logic with identity, with all formulas closed;
- *gamma*, (nearly) isomorphic to normal modal logic.

Alpha nests in *beta* and *gamma*. *Beta* does not nest in *gamma*, quantified modal logic being more than even Peirce could envisage.

Alpha

The syntax is:

- The blank page;
- Single letters or phrases written anywhere on the page;
- Any graph may be enclosed by a simple closed curve called a *cut* or *sep*. A cut can be empty. Cuts can nest and concatenate at will, but must never intersect.

Any well-formed part of a graph is a **subgraph**.

The semantics are:

- The blank page denotes **Truth**;
- Letters, phrases, subgraphs, and entire graphs may be **True** or **False**;
- To enclose a subgraph with a cut is equivalent to logical negation or Boolean complementation. Hence an empty cut denotes **False**;
- All subgraphs within a given cut are tacitly conjoined.

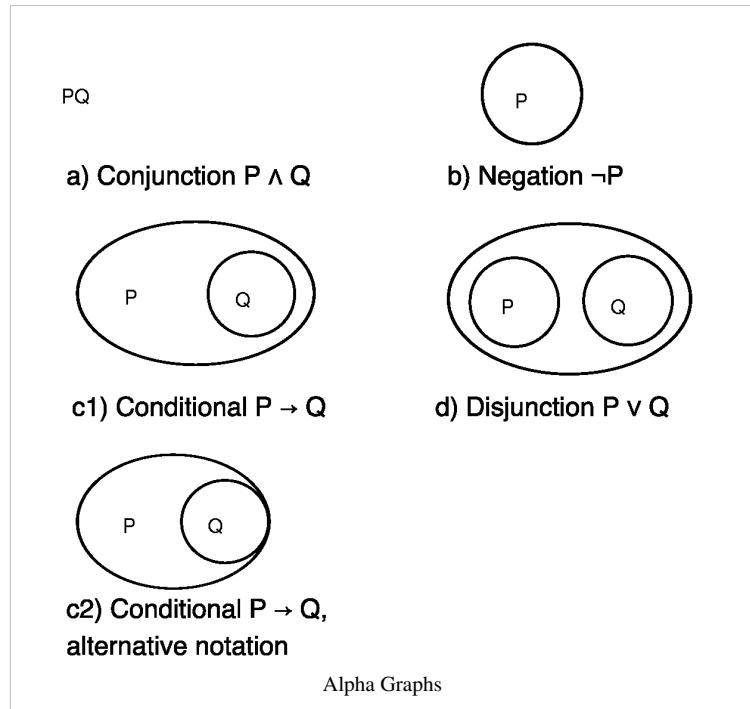
Hence the *alpha* graphs are a minimalist notation for sentential logic, grounded in the expressive adequacy of **And** and **Not**. The *alpha* graphs constitute a radical simplification of the two-element Boolean algebra and the truth functors.

The *depth* of an object is the number of cuts that enclose it.

Rules of inference:

- Insertion - Any subgraph may be inserted into an odd numbered depth.
- Erasure - Any subgraph in an even numbered depth may be erased.

Rules of equivalence:



- Double cut - A pair of cuts with nothing between them may be drawn around any subgraph. Likewise two nested cuts with nothing between them may be erased. This rule is equivalent to Boolean involution.
- Iteration/Deiteration – To understand this rule, it is best to view a graph as a tree structure having nodes and ancestors. Any subgraph P in node n may be copied into any node depending on n . Likewise, any subgraph P in node n may be erased if there exists a copy of P in some node ancestral to n (i.e., some node on which n depends). For an equivalent rule in an algebraic context, see C2 in Laws of form.

A proof manipulates a graph by a series of steps, with each step justified by one of the above rules. If a graph can be reduced by steps to the blank page or an empty cut, it is what is now called a tautology (or the complement thereof). Graphs that cannot be simplified beyond a certain point are analogues of the satisfiable formulas of first-order logic.

Beta

Peirce notated predicates using intuitive English phrases; the standard notation of contemporary logic, capital Latin letters, may also be employed. A dot asserts the existence of some individual in the domain of discourse. Multiple instances of the same object are linked by a line, called the "line of identity". There are no literal variables or quantifiers in the sense of first-order logic. A line of identity connecting two or more predicates can be read as asserting that the predicates share a common variable. The presence of lines of identity requires modifying the *alpha* rules of Equivalence.

The beta graphs can be read as a system in which all formula are to be taken as closed, because all variables are implicitly quantified. If the "shallowest" part of a line of identity has even (odd) depth, the associated variable is tacitly existentially (universally) quantified.

Zeman (1964)^[2] was the first to note that the *beta* graphs are isomorphic to first-order logic with equality (also see Zeman 1967). However, the secondary literature, especially Roberts (1973) and Shin (2002), does not agree on just how this is so. Peirce's writings do not address this question, because first-order logic was first clearly articulated only some years after his death, in the 1928 first edition of David Hilbert and Wilhelm Ackermann's *Principles of Mathematical Logic*.

Gamma

Add to the syntax of *alpha* a second kind of simple closed curve, written using a dashed rather than a solid line. Peirce proposed rules for this second style of cut, which can be read as the primitive unary operator of modal logic.

Zeman (1964)^[2] was the first to note that straightforward emendations of the *gamma* graph rules yield the well-known modal logics S4 and S5. Hence the *gamma* graphs can be read as a peculiar form of normal modal logic. This finding of Zeman's has gone unremarked to this day, but is nonetheless included here as a point of interest.

Peirce's role

The existential graphs are a curious offspring of Peirce the logician/mathematician with Peirce the founder of a major strand of semiotics. Peirce's graphical logic is but one of his many accomplishments in logic and mathematics. In a series of papers beginning in 1867, and culminating with his classic paper in the 1885 *American Journal of Mathematics*, Peirce developed much of the two-element Boolean algebra, propositional calculus, quantification and the predicate calculus, and some rudimentary set theory. Model theorists consider Peirce the first of their kind. He also extended De Morgan's relation algebra. He stopped short of metalogic (which eluded even *Principia Mathematica*).

But Peirce's evolving semiotic theory led him to doubt the value of logic formulated using conventional linear notation, and to prefer that logic and mathematics be notated in two (or even three) dimensions. His work went beyond Euler's diagrams and Venn's revision thereof. Frege's 1879 *Begriffsschrift* also employed a two-dimensional notation for logic, but one very different from Peirce's.

Peirce's first published paper on graphical logic (reprinted in Vol. 3 of his *Collected Papers*) proposed a system dual (in effect) to the *alpha* existential graphs, called the entitative graphs. He very soon abandoned this formalism in favor of the existential graphs. The graphical logic went unremarked during his lifetime, and was invariably denigrated or ignored after his death, until the Ph.D. theses by Roberts (1964) and Zeman (1964)^[2].

Notes

- [1] Peirce, C. S., "[On Junctures and Fractures in Logic]" (editors' title for MS 427 (the new numbering system), Fall–Winter 1882), and "Letter, Peirce to O. H. Mitchell" (L 294, 21 December 1882), *Writings of Charles S. Peirce*, v. 4, "Junctures" on pp. 391–3 (Google preview (<http://books.google.com/books?id=E7ZUnx3FqrcC&q=MS+427>)) and the letter on pp. 394–9 (Google preview (<http://books.google.com/books?id=E7ZUnx3FqrcC&q=L+294>)). See Sowa, John F. (1997), "Matching Logical Structure to Linguistic Structure", *Studies in the Logic of Charles Sanders Peirce*, Nathan Houser, Don D. Roberts, and James Van Evra, editors, Bloomington and Indianapolis: Indiana University Press, pp. 418–44, see 420, 425, 426, 428.
- [2] <http://www.clas.ufl.edu/users/jzeman/>

References

Primary literature

- 1931–35 & 1958. *The Collected Papers of Charles Sanders Peirce*. Volume 4, Book II: "Existential Graphs", consists of paragraphs 347–584. A discussion also begins in paragraph 617.
- Paragraphs 347–349 (II.1.1. "Logical Diagram")—Peirce's definition "Logical Diagram (or Graph)" in Baldwin's *Dictionary of Philosophy and Psychology* (1902), v. 2, p. 28 (<http://books.google.com/books?id=Dc8YAAAAIAAJ&pg=PA28>). *Classics in the History of Psychology* Eprint ([http://psychclassics.yorku.ca/Baldwin/Dictionary/defs/L4defs.htm#Logical Diagram](http://psychclassics.yorku.ca/Baldwin/Dictionary/defs/L4defs.htm#Logical%20Diagram)).
- Paragraphs 350–371 (II.1.2. "Of Euler's Diagrams")—from "Graphs" (manuscript 479) c. 1903.
- Paragraphs 372–584 Eprint (<http://www.existentialgraphs.com/#table2>).
- Paragraphs 372–393 (II.2. "Symbolic Logic")—Peirce's part of "Symbolic Logic" in Baldwin's *Dictionary of Philosophy and Psychology* (1902) v. 2, pp. 645 (<http://books.google.com/books?id=Dc8YAAAAIAAJ&pg=PA645>)–650, beginning (near second column's top) with "If symbolic logic be defined...". Paragraph 393 (Baldwin's DPP2 p. 650) is by Peirce and Christine Ladd-Franklin ("C.S.P., C.L.F.").
- Paragraphs 394–417 (II.3. "Existential Graphs")—from Peirce's pamphlet *A Syllabus of Certain Topics of Logic*, pp. 15–23, Alfred Mudge & Son, Boston (1903).
- Paragraphs 418–509 (II.4. "On Existential Graphs, Euler's Diagrams, and Logical Algebra")—from "Logical Tracts, No. 2" (manuscript 492), c. 1903.
- Paragraphs 510–529 (II.5. "The Gamma Part of Existential Graphs")—from "Lowell Lectures of 1903," Lecture IV (manuscript 467).
- Paragraphs 530–572 (II.6.)—"Prolegomena To an Apology For Pragmaticism" (1906), *The Monist*, v. XVI, n. 4, pp. 492 (<http://books.google.com/books?id=3KoLAAAIAAJ&pg=RA2-PA492-546>). Corrections (1907) in *The Monist* v. XVII, p. 160 (<http://books.google.com/books?id=RqsLAAAIAAJ&pg=PA160>).
- Paragraphs 573–584 (II.7. "An Improvement on the Gamma Graphs")—from "For the National Academy of Science, 1906 April Meeting in Washington" (manuscript 490).
- Paragraphs 617–623 (at least) (in Book III, Ch. 2, §2, paragraphs 594–642)—from "Some Amazing Mazes: Explanation of Curiosity the First", *The Monist*, v. XVIII, 1908, n. 3, pp. 416 (<http://books.google.com/books?id=CqsLAAAIAAJ&pg=PA416>), see starting p. 440 (<http://books.google.com/books?id=CqsLAAAIAAJ&pg=PA440>).
- 1992. "Lecture Three: The Logic of Relatives", *Reasoning and the Logic of Things*, pp. 146–64. Ketner, Kenneth Laine (editing and introduction), and Hilary Putnam (commentary). Harvard University Press. Peirce's 1898 lectures in Cambridge, Massachusetts.

- 1977, 2001. *Semiotic and Significs: The Correspondence between C.S. Peirce and Victoria Lady Welby*. Hardwick, C.S., ed. Lubbock TX: Texas Tech University Press. 2nd edition 2001.
- A transcription of Peirce's MS 514 (<http://www.jfsowa.com/peirce/ms514.htm>) (1909), edited with commentary by John Sowa.

Currently, the chronological critical edition of Peirce's works, the *Writings*, extends only to 1892. Much of Peirce's work on logical graphs consists of manuscripts written after that date and still unpublished. Hence our understanding of Peirce's graphical logic is likely to change as the remaining 23 volumes of the chronological edition appear.

Secondary literature

- Hammer, Eric M. (1998), "Semantics for Existential Graphs," *Journal of Philosophical Logic* 27: 489-503.
- Ketner, Kenneth Laine
 - (1981), "The Best Example of Semiosis and Its Use in Teaching Semiotics", *American Journal of Semiotics* v. I, n. 1-2, pp. 47–83. Article is an introduction to existential graphs.
 - (1990), *Elements of Logic: An Introduction to Peirce's Existential Graphs*, Texas Tech University Press, Lubbock, TX, 99 pages, spiral-bound.
- Queiroz, Joao & Stjernfelt, Frederik
 - (2011), "Diagrammatical Reasoning and Peircean Logic Representation". vol.186 (1/4). (special issue on Peirce's diagrammatic logic) (<http://www.degruyter.com/view/j/semi.2011.2011.issue-186/issue-files/semi.2011.2011.issue-186.xml>)
- Roberts, Don D.
 - (1964), "Existential Graphs and Natural Deduction" in Moore, E. C., and Robin, R. S., eds., *Studies in the Philosophy of C. S. Peirce, 2nd series*. Amherst MA: University of Massachusetts Press. The first publication to show any sympathy and understanding for Peirce's graphical logic.
 - (1973). *The Existential Graphs of C.S. Peirce*. John Benjamins. An outgrowth of his 1963 thesis.
- Shin, Sun-Joo (2002), *The Iconic Logic of Peirce's Graphs*. MIT Press.
- Zeman, J. J.
 - (1964), *The Graphical Logic of C.S. Peirce*. (<http://www.clas.ufl.edu/users/jzeman/>) Unpublished Ph.D. thesis submitted to the University of Chicago.
 - (1967), "A System of Implicit Quantification," *Journal of Symbolic Logic* 32: 480-504.

External links

- Stanford Encyclopedia of Philosophy: Peirce's Logic (<http://setis.library.usyd.edu.au/stanford/entries/peirce-logic/#SymIcoRep>) by Sun-Joo Shin and Eric Hammer.
- Dau, Frithjof, Peirce's Existential Graphs --- Readings and Links. (http://www.dr-dau.net/eg_readings.shtml) An annotated bibliography on the existential graphs.
- Gottschall, Christian, Proof Builder (<http://logik.phl.univie.ac.at/~chris/gateway/formular-uk-peirce.html>) — Java applet for deriving Alpha graphs.
- Liu, Xin-Wen, " The literature of C.S. Peirce's Existential Graphs (<http://replay.web.archive.org/20081022205810/http://philosophy.cass.cn/facu/liuxinwen/01.htm>)" (via Wayback Machine), Institute of Philosophy, Chinese Academy of Social Sciences, Beijing, PRC.
- Sowa, John F., "Laws, Facts, and Contexts: Foundations for Multimodal Reasoning" (<http://www.jfsowa.com/pubs/laws.htm>) accessdate=2009-10-23 Existential graphs and conceptual graphs.
- Van Heuveln, Bram, " Existential Graphs. (<http://www.cogsci.rpi.edu/~heuveb/research/EG/index.html>)" Dept. of Cognitive Science, Rensselaer Polytechnic Institute. Alpha only.

- Zeman, Jay J., " Existential Graphs (<http://www.existentialgraphs.com/>)". With four online papers (<http://www.existentialgraphs.com/#table2>) by Peirce.

Implicant

In Boolean logic, an **implicant** is a "covering" (sum term or product term) of one or more minterms in a sum of products (or maxterms in a product of sums) of a boolean function. Formally, a product term P in a sum of products is an **implicant** of the Boolean function F if P implies F . More precisely:

P implies F (and thus is an implicant of F) if F also takes the value 1 whenever P equals 1.

where

- F is a Boolean function of n variables.
- P is a product term.

This means that $P \Rightarrow F$ with respect to the natural ordering of the Boolean space. For instance, the function

$$f(x, y, z, w) = xy + yz + w$$

is implied by xy , by xyz , by $xyzw$, by w and many others; these are the implicants of f .

Prime implicant

A **prime implicant** of a function is an implicant that cannot be covered by a more general (more reduced - meaning with fewer literals) implicant. W.V. Quine defined a *prime implicant* of F to be an implicant that is minimal - that is, the removal of any literal from P results in a non-implicant for F . **Essential prime implicants** are prime implicants that cover an output of the function that no combination of other prime implicants is able to cover.

Using the example above, one can easily see that while xy (and others) is a prime implicant, xyz and $xyzw$ are not. From the latter, multiple literals can be removed to make it prime:

- x , y and z can be removed, yielding w .
- Alternatively, z and w can be removed, yielding xy .
- Finally, x and w can be removed, yielding yz .

The process of removing literals from a Boolean term is called **expanding** the term. Expanding by one literal doubles the number of input combinations for which the term is true (in binary Boolean algebra). Using the example function above, we may expand xyz to xy or to yz without changing the cover of f .^[1]

The sum of all prime implicants of a Boolean function is called its **complete sum**, **minimal covering sum**, or Blake canonical form.

References

[1] De Micheli, Giovanni. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, Inc., 1994

External links

- Slides explaining implicants, prime implicants and essential prime implicants (<http://www.cs.ualberta.ca/~amaral/courses/329/webslides/Topic4-KarnaughMaps/sld021.htm>)
- Examples of finding essential prime implicants using K-map (<http://web.cecs.pdx.edu/~mcnames/ECE171/Lectures/Lecture10.html>)

Laws of Form

Laws of Form (hereinafter *LoF*) is a book by G. Spencer-Brown, published in 1969, that straddles the boundary between mathematics and philosophy. *LoF* describes three distinct logical systems:

- The *primary arithmetic* (described in Chapter 4 of *LoF*), whose models include Boolean arithmetic;
- The *primary algebra* (Chapter 6 of *LoF*), whose models include the two-element Boolean algebra (hereinafter abbreviated **2**), Boolean logic, and the classical propositional calculus;
- *Equations of the second degree* (Chapter 11), whose interpretations include finite automata and Alonzo Church's Restricted Recursive Arithmetic (RRA).

Boundary algebra is Dr Philip Meguire^[1]'s (2011) term^[2] for the union of the primary algebra (hereinafter abbreviated *pa*) and the primary arithmetic. "Laws of Form" sometimes loosely refers to the *pa* as well as to *LoF*.

The book

LoF emerged out of work in electronic engineering its author did around 1960, and from subsequent lectures on mathematical logic he gave under the auspices of the University of London's extension program. *LoF* has appeared in several editions, the most recent being a 1997 German translation, and has never gone out of print.

The mathematics fills only about 55pp and is rather elementary. But *LoF*'s mystical and declamatory prose, and its love of paradox, make it a challenging read for all. Spencer-Brown was influenced by Wittgenstein and R. D. Laing. *LoF* also echoes a number of themes from the writings of Charles Sanders Peirce, Bertrand Russell, and Alfred North Whitehead.

The entire book is written in an operational way, giving instructions to the reader instead of telling him what *is*. In accordance with G. Spencer-Brown's interest in paradoxes, the only sentence that makes a statement that something is, is the statement, which says no such statements are used in this book.^[3] Except for this one sentence the book can be seen as an example of E-Prime.

Reception

Ostensibly a work of formal mathematics and philosophy, *LoF* became something of a cult classic, praised in the *Whole Earth Catalog*. Those who agree point to *LoF* as embodying an enigmatic "mathematics of consciousness," its algebraic symbolism capturing an (perhaps even *the*) implicit root of cognition: the ability to *distinguish*. *LoF* argues that the *pa* (primary algebra) reveals striking connections among logic, Boolean algebra, and arithmetic, and the philosophy of language and mind.

Banaschewski (1977)^[4] argues that the *pa* is nothing but new notation for Boolean algebra. Indeed, the two-element Boolean algebra **2** can be seen as the intended interpretation of the *pa*. Yet the notation of the *pa*:

- Fully exploits the duality characterizing not just Boolean algebras but all lattices;
- Highlights how syntactically distinct statements in logic and **2** can have identical semantics;
- Dramatically simplifies Boolean algebra calculations, and proofs in sentential and syllogistic logic.

Moreover, the syntax of the *pa* can be extended to formal systems other than **2** and sentential logic, resulting in *boundary mathematics* (see Related Work below).

LoF has influenced, among others, Heinz von Foerster, Louis Kauffman, Niklas Luhmann, Humberto Maturana, Francisco Varela and William Bricken. Some of these authors have modified the primary algebra in a variety of interesting ways.

LoF claimed that certain well-known mathematical conjectures of very long standing, such as the Four Color Theorem, Fermat's Last Theorem, and the Goldbach conjecture, are provable using extensions of the *pa*. Spencer-Brown eventually circulated a purported proof of the Four Color Theorem, but it met with skepticism.^[5]

(The Four Color Theorem and Fermat's Last Theorem were proved in 1976 and 1995, respectively, using methods owing nothing to *LoF*.)

The form (Chapter 1)

The symbol:



also called the *mark* or *cross*, is the essential feature of the Laws of Form. In Spencer-Brown's inimitable and enigmatic fashion, the Mark symbolizes the root of cognition, i.e., the dualistic Mark indicates the capability of differentiating a "this" from "everything else *but* this."

In *LoF*, a Cross denotes the drawing of a "distinction", and can be thought of as signifying the following, all at once:

- The act of drawing a boundary around something, thus separating it from everything else;
- That which becomes distinct from everything by drawing the boundary;
- Crossing from one side of the boundary to the other.

All three ways imply an *action* on the part of the cognitive entity (e.g., person) making the distinction. As *LoF* puts it:

"The first command:

- Draw a distinction

can well be expressed in such ways as:

- Let there be a distinction,
- Find a distinction,
- See a distinction,
- Describe a distinction,
- Define a distinction,

Or:

- Let a distinction be drawn." (*LoF*, Notes to chapter 2)

The counterpoint to the Marked state is the Unmarked state, which is simply nothing, the void, represented by a blank space. It is simply the absence of a Cross. No distinction has been made and nothing has been crossed. The Marked state and the void are the two primitive values of the Laws of Form.

The Cross can be seen as denoting the distinction between two states, one "considered as a symbol" and another not so considered. From this fact arises a curious resonance with some theories of consciousness and language. Paradoxically, the Form is at once Observer and Observed, and is also the creative act of making an observation. *LoF* (excluding back matter) closes with the words:

"...the first distinction, the Mark and the observer are not only interchangeable, but, in the form, identical."

C. S. Peirce came to a related insight in the 1890s; see Related Work.

The primary arithmetic (Chapter 4)

The syntax of the **primary arithmetic (PA)** goes as follows. There are just two atomic expressions:

- The empty Cross $\boxed{\quad}$;
- All or part of the blank page (the "void").

There are two inductive rules:

- A Cross $\boxed{\quad}$ may be written over any expression;
- Any two expressions may be concatenated.

The semantics of the primary arithmetic are perhaps nothing more than the sole explicit definition in *LoF*: *Distinctio nis perfect continence*.

Let the *unmarked state* be a synonym for the void. Let an empty Cross denote the *marked state*. To cross is to move from one of the unmarked or marked states to the other. We can now state the "arithmetical" axioms **A1** and **A2**, which ground the primary arithmetic (and hence all of the Laws of Form):

A1. The law of Calling. Calling twice from a state is indistinguishable from calling once. To make a distinction twice has the same effect as making it once. For example, saying "Let there be light" and then saying "Let there be light" again, is the same as saying it once. Formally:

$$\boxed{\quad} \boxed{\quad} = \boxed{\quad}$$

A2. The law of Crossing. After crossing from the unmarked to the marked state, crossing again ("recrossing") starting from the marked state returns one to the unmarked state. Hence recrossing annuls crossing. Formally:

$$\overline{\boxed{\quad}} =$$

In both **A1** and **A2**, the expression to the right of '=' has fewer symbols than the expression to the left of '='. This suggests that every primary arithmetic expression can, by repeated application of **A1** and **A2**, be *simplified* to one of two states: the marked or the unmarked state. This is indeed the case, and the result is the expression's *simplification*. The two fundamental metatheorems of the primary arithmetic state that:

- Every finite expression has a unique simplification. (**T3** in *LoF*);
- Starting from an initial marked or unmarked state, "complicating" an expression by a finite number of repeated application of **A1** and **A2** cannot yield an expression whose simplification differs from the initial state. (**T4** in *LoF*).

Thus the relation of *logical equivalence* partitions all primary arithmetic expressions into two equivalence classes: those that simplify to the Cross, and those that simplify to the void.

A1 and **A2** have loose analogs in the properties of series and parallel electrical circuits, and in other ways of diagramming processes, including flowcharting. **A1** corresponds to a parallel connection and **A2** to a series connection, with the understanding that making a distinction corresponds to changing how two points in a circuit are connected, and not simply to adding wiring.

The primary arithmetic is analogous to the following formal languages from mathematics and computer science:

- A Dyck language of order 1 with a null alphabet;
- The simplest context-free language in the Chomsky hierarchy;
- A rewrite system that is strongly normalizing and confluent.

The phrase *calculus of indications* in *LoF* is a synonym for "primary arithmetic".

The notion of canon

A concept peculiar to *LoF* is that of *canon*. While *LoF* does not define canon, the following two excerpts from the Notes to chpt. 2 are apt:

"The more important structures of command are sometimes called *canons*. They are the ways in which the guiding injunctions appear to group themselves in constellations, and are thus by no means independent of each other. A canon bears the distinction of being outside (i.e., describing) the system under construction, but a command to construct (e.g., 'draw a distinction'), even though it may be of central importance, is not a canon. A canon is an order, or set of orders, to permit or allow, but not to construct or create."

"...the primary form of mathematical communication is not description but injunction... Music is a similar art form, the composer does not even attempt to describe the set of sounds he has in mind, much less the set of feelings occasioned through them, but writes down a set of commands which, if they are obeyed by the performer, can result in a reproduction, to the listener, of the composer's original experience."

These excerpts relate to the distinction in metalogic between the object language, the formal language of the logical system under discussion, and the metalanguage, a language (often a natural language) distinct from the object language, employed to exposit and discuss the object language. The first quote seems to assert that the *canons* are part of the metalanguage. The second quote seems to assert that statements in the object language are essentially commands addressed to the reader by the author. Neither assertion holds in standard metalogic.

The primary algebra (Chapter 6)

Syntax

Given any valid primary arithmetic expression, insert into one or more locations any number of Latin letters bearing optional numerical subscripts; the result is a *pa* formula. Letters so employed in mathematics and logic are called variables. A *pa* variable indicates a location where one can write the primitive value $\overline{\square}$ or its complement $\overline{\overline{\square}}$. Multiple instances of the same variable denote multiple locations of the same primitive value.

Rules governing logical equivalence

The sign '=' may link two logically equivalent expressions; the result is an equation. By "logically equivalent" is meant that the two expressions have the same simplification. Logical equivalence is an equivalence relation over the set of *pa* formulas, governed by the rules **R1** and **R2**. Let *C* and *D* be formulae each containing at least one instance of the subformula *A*:

- **R1, Substitution of equals.** Replace *one or more* instances of *A* in *C* by *B*, resulting in *E*. If $A=B$, then $C=E$.
- **R2, Uniform replacement.** Replace *all* instances of *A* in *C* and *D* with *B*. *C* becomes *E* and *D* becomes *F*. If $C=D$, then $E=F$. Note that $A=B$ is not required.

R2 is employed very frequently in *pa* demonstrations (see below), almost always silently. These rules are routinely invoked in logic and most of mathematics, nearly always unconsciously.

The *pa* consists of equations, i.e., pairs of formulae linked by an infix '='. **R1** and **R2** enable transforming one equation into another. Hence the *pa* is an *equational* formal system, like the many algebraic structures, including Boolean algebra, that are varieties. Equational logic was common before *Principia Mathematica* (e.g., Peirce,^{1,2,3} Johnson 1892), and has present-day advocates (Gries and Schneider 1993).

Conventional mathematical logic consists of tautological formulae, signalled by a prefixed turnstile. To denote that the *pa* formula *A* is a tautology, simply write "*A* = $\overline{\square}$ ". If one replaces '=' in **R1** and **R2** with the biconditional, the resulting rules hold in conventional logic. However, conventional logic relies mainly on the rule modus ponens; thus

conventional logic is *ponential*. The equational-ponential dichotomy distills much of what distinguishes mathematical logic from the rest of mathematics.

Initials

An *initial* is a *pa* equation verifiable by a decision procedure and as such is *not* an axiom. *LoF* lays down the initials:

- $J1 : ((A)A) = .$

The absence of anything to the right of the " $=$ " above, is deliberate.

- $J2 : ((A)(B))C = ((AC)(BC)).$

J2 is the familiar distributive law of sentential logic and Boolean algebra.

Another set of initials, friendlier to calculations, is:

- $J0 : (())A = A.$
- $J1a : (A)A = ()$
- $C2 : A(AB) = A(B).$

It is thanks to **C2** that the *pa* is a lattice. By virtue of **J1a**, it is a complemented lattice whose upper bound is $()$. By **J0**, $(())$ is the corresponding lower bound and identity element. **J0** is also an algebraic version of **A2** and makes clear the sense in which $(())$ aliases with the blank page.

T13 in *LoF* generalizes **C2** as follows. Any *pa* (or sentential logic) formula B can be viewed as an ordered tree with *branches*. Then:

T13: A subformula A can be copied at will into any depth of B greater than that of A , as long as A and its copy are in the same branch of B . Also, given multiple instances of A in the same branch of B , all instances but the shallowest are redundant.

While a proof of T13 would require induction, the intuition underlying it should be clear.

C2 or its equivalent is named:

- "Generation" in *LoF*;
- "Exclusion" in Johnson (1892);
- "Pervasion" in the work of William Bricken;
- "Mimesis" in the entry logical nand.

Perhaps the first instance of an axiom or rule with the power of **C2** was the "Rule of (De)Iteration," combining T13 and $AA=A$, of C. S. Peirce's existential graphs.

LoF asserts that concatenation can be read as commuting and associating by default and hence need not be explicitly assumed or demonstrated. (Peirce made a similar assertion about his existential graphs.) Let a period be a temporary notation to establish grouping. That concatenation commutes and associates may then be demonstrated from the:

- Initial $AC.D=CD.A$ and the consequence $AA=A$ (Byrne 1946). This result holds for all lattices, because $AA=A$ is an easy consequence of the absorption law, which holds for all lattices;
- Initials $AC.D=AD.C$ and **J0**. Since **J0** holds only for lattices with a lower bound, this method holds only for bounded lattices (which include the *pa* and **2**). Commutativity is trivial; just set $A=(())$. Associativity: $AC.D = CA.D = CD.A = A.CD$.

Having demonstrated associativity, the period can be discarded.

The initials in Meguire (2011) are $AC.D=CD.A$, called **B1**; **B2**, J0 above; **B3**, J1a above; and **B4**, C2. By design, these initials are very similar to the axioms for an abelian group, **G1-G3** below.

Proof theory

The *pa* contains three kinds of proved assertions:

- *Consequence* is a *pa* equation verified by a *demonstration*. A demonstration consists of a sequence of *steps*, each step justified by an initial or a previously demonstrated consequence.
- *Theorem* is a statement in the metalanguage verified by a *proof*, i.e., an argument, formulated in the metalanguage, that is accepted by trained mathematicians and logicians.
- *Initial*, defined above. Demonstrations and proofs invoke an initial as if it were an axiom.

The distinction between consequence and theorem holds for all formal systems, including mathematics and logic, but is usually not made explicit. A demonstration or decision procedure can be carried out and verified by computer. The proof of a theorem cannot be.

Let A and B be *pa* formulas. A demonstration of $A=B$ may proceed in either of two ways:

- Modify A in steps until B is obtained, or vice versa;
- Simplify both $(A)B$ and $(B)A$ to $\boxed{\quad}$. This is known as a "calculation".

Once $A=B$ has been demonstrated, $A=B$ can be invoked to justify steps in subsequent demonstrations. *pa* demonstrations and calculations often require no more than **J1a**, **J2**, **C2**, and the consequences $(\Box A)=()$ (**C3** in *LoF*), $((A))=A$ (**C1**), and $AA=A$ (**C5**).

The consequence $((A)B)C = (AC)((B)C)$, **C7** in *LoF*, enables an algorithm, sketched in *LoF*'s proof of T14, that transforms an arbitrary *pa* formula to an equivalent formula whose depth does not exceed two. The result is a *normal form*, the *pa* analog of the conjunctive normal form. *LoF* (T14-15) proves the *pa* analog of the well-known Boolean algebra theorem that every formula has a normal form.

Let A be a subformula of some formula B . When paired with **C3**, **J1a** can be viewed as the closure condition for calculations: B is a tautology if and only if A and (A) both appear in depth 0 of B . A related condition appears in some versions of natural deduction. A demonstration by calculation is often little more than:

- Invoking T13 repeatedly to eliminate redundant subformulae;
- Erasing any subformulae having the form $((A)A)$.

The last step of a calculation always invokes **J1a**.

LoF includes elegant new proofs of the following standard metatheory:

- *Completeness*: all *pa* consequences are demonstrable from the initials (T17).
- *Independence*: **J1** cannot be demonstrated from **J2** and vice versa (T18).

That sentential logic is complete is taught in every first university course in mathematical logic. But university courses in Boolean algebra seldom mention the completeness of **2**.

Interpretations

If the Marked and Unmarked states are read as the Boolean values 1 and 0 (or **True** and **False**), the *pa* interprets **2** (or sentential logic). *LoF* shows how the *pa* can interpret the syllogism. Each of these interpretations is discussed in a subsection below. Extending the *pa* so that it could interpret standard first-order logic has yet to be done, but Peirce's *beta* existential graphs suggest that this extension is feasible.

Two-element Boolean algebra 2

The *pa* is an elegant minimalist notation for the two-element Boolean algebra **2**. Let:

- One of Boolean meet (\times) or join ($+$) interpret concatenation;
- The complement of A interpret \boxed{a}
- 0 (1) interpret the empty Mark if meet (join) interprets concatenation.

If meet (join) interprets AC , then join (meet) interprets $((A)(C))$. Hence the *pa* and **2** are isomorphic but for one detail: *pa* complementation can be nullary, in which case it denotes a primitive value. Modulo this detail, **2** is a model of the primary algebra. The primary arithmetic suggests the following arithmetic axiomatization of **2**: $1+1=1+0=0+1=1=\sim 0$, and $0+0=0=\sim 1$.

The set $B = \{\overline{\overline{\square}}, \overline{\square}\}$ is the Boolean domain or *carrier*. In the language of universal algebra, the *pa* is the algebraic structure $\langle B, --, (-), () \rangle$ of type $\langle 2, 1, 0 \rangle$. The expressive adequacy of the Sheffer stroke points to the *pa* also being a $\langle B, (-), () \rangle$ algebra of type $\langle 2, 0 \rangle$. In both cases, the identities are J1a, J0, C2, and $ACD=CDA$. Since the *pa* and **2** are isomorphic, **2** can be seen as a $\langle B, +, \neg, 1 \rangle$ algebra of type $\langle 2, 1, 0 \rangle$. This description of **2** is simpler than the conventional one, namely an $\langle B, +, \times, \neg, 1, 0 \rangle$ algebra of type $\langle 2, 2, 1, 0, 0 \rangle$.

Sentential logic

Let the blank page denote **True** or **False**, and let a Cross be read as **Not**. Then the primary arithmetic has the following sentential reading:

= **False**

$\overline{\square} = \text{True} = \text{not False}$

$\overline{\overline{\square}} = \text{Not True} = \text{False}$

The *pa* interprets sentential logic as follows. A letter represents any given sentential expression. Thus:

$\overline{a} \quad \text{interprets Not A}$

$a \quad b \quad \text{interprets A Or B}$

$\overline{a} \quad b \quad \text{interprets Not A Or B or If A Then B.}$

$\overline{a} \quad \overline{b} \quad \text{interprets Not (Not A Or Not B)} \\ \text{or Not (If A Then Not B)}$

or A And B.

$((A)B)(A(B)), ((A)(B))(AB)$ both interpret **A if and only if B or A is equivalent to B**.

Thus any expression in sentential logic has a *pa* translation. Equivalently, the *pa* interprets sentential logic. Given an assignment of every variable to the Marked or Unmarked states, this *pa* translation reduces to a PA expression, which can be simplified. Repeating this exercise for all possible assignments of the two primitive values to each variable, reveals whether the original expression is tautological or satisfiable. This is an example of a decision procedure, one more or less in the spirit of conventional truth tables. Given some *pa* formula containing N variables, this decision procedure requires simplifying 2^N PA formulae. For a less tedious decision procedure more in the spirit of Quine's "truth value analysis," see Meguire (2003).

Schwartz (1981) proved that the *pa* is equivalent -- syntactically, semantically, and proof theoretically—with the classical propositional calculus. Likewise, it can be shown that the *pa* is syntactically equivalent with expressions built up in the usual way from the classical truth values **true** and **false**, the logical connectives NOT, OR, and AND, and parentheses.

Interpreting the Unmarked State as **False** is wholly arbitrary; that state can equally well be read as **True**. All that is required is that the interpretation of concatenation change from OR to AND. IF A THEN B now translates as $(A(B))$ instead of $(A)B$. More generally, the *pa* is "self-dual," meaning that any *pa* formula has two sentential or Boolean readings, each the dual of the other. Another consequence of self-duality is the irrelevance of De Morgan's laws; those laws are built into the syntax of the *pa* from the outset.

The true nature of the distinction between the *pa* on the one hand, and **2** and sentential logic on the other, now emerges. In the latter formalisms, complementation/negation operating on "nothing" is not well-formed. But an empty Cross is a well-formed *pa* expression, denoting the Marked state, a primitive value. Hence a nonempty Cross is an operator, while an empty Cross is an operand because it denotes a primitive value. Thus the *pa* reveals that the heretofore distinct mathematical concepts of operator and operand are in fact merely different facets of a single fundamental action, the making of a distinction.

Syllogisms

Appendix 2 of *LoF* shows how to translate traditional syllogisms and sorites into the *pa*. A valid syllogism is simply one whose *pa* translation simplifies to an empty Cross. Let A^* denote a *literal*, i.e., either A or (A) , indifferently. Then all syllogisms that do not require that one or more terms be assumed nonempty are one of 24 possible permutations of a generalization of Barbara whose *pa* equivalent is $(A^*B)((B)C^*)A^*C^*$. These 24 possible permutations include the 19 syllogistic forms deemed valid in Aristotelian and medieval logic. This *pa* translation of syllogistic logic also suggests that the *pa* can interpret monadic and term logic, and that the *pa* has affinities to the Boolean term schemata of Quine (1982: Part II).

An example of calculation

The following calculation of Leibniz's nontrivial *Praeclarum Theorema* exemplifies the demonstrative power of the *pa*. Let C1 be $((A))=A$, and let OI mean that variables and subformulae have been reordered in a way that commutativity and associativity permit. Because the only commutative connective appearing in the *Theorema* is conjunction, it is simpler to translate the *Theorema* into the *pa* using the dual interpretation. The objective then becomes one of simplifying that translation to $(())$.

- $[(P \rightarrow R) \wedge (Q \rightarrow S)] \rightarrow [(P \wedge Q) \rightarrow (R \wedge S)]$. *Praeclarum Theorema*.
- $((P(R))(Q(S))((PQ(RS))))$. *pa* translation.
- $= ((\mathbf{P}(R))P(\mathbf{Q}(S))Q(RS))$. OI; C1.
- $= (((R))((S))PQ(RS))$. Invoke C2 2x to eliminate the bold letters in the previous expression; OI.
- $= (RSPQ(RS))$. C1,2x.
- $= ((RSPQ)RSPQ)$. C2; OI.
- $= (())$. J1. \square

Remarks:

- C1 (C2) is repeatedly invoked in a fairly mechanical way to eliminate nested parentheses (variable instances). This is the essence of the calculation method;
- A single invocation of J1 (or, in other contexts, J1a) terminates the calculation. This too is typical;
- Experienced users of the *pa* are free to invoke OI silently. OI aside, the demonstration requires a mere 7 steps.

A technical aside

Given some standard notions from mathematical logic and some suggestions in Bostock (1997: 83, fn 11, 12), $\{ \}$ and may be interpreted as the classical bivalent truth values. Let the extension[6] of an n -place atomic formula be the set of ordered n -tuples of individuals that satisfy it (i.e., for which it comes out true). Let a sentential variable be a 0-place atomic formula, whose extension is a classical truth value, by definition. An ordered 2-tuple is an ordered pair, whose standard (Kuratowski's definition) set theoretic definition is $\langle a,b \rangle = \{\{a\},\{a,b\}\}$, where a,b are individuals. Ordered n -tuples for any $n > 2$ may be obtained from ordered pairs by a well-known recursive construction. Dana Scott has remarked that the extension of a sentential variable can also be seen as the empty ordered pair (ordered 0-tuple), $\{\{\},\{\}\} = \{\{a\}\}$ because $\{a,a\}=\{a\}$ for all a . Hence has the interpretation **True**. Reading $\{\}$ as **False** follows naturally.

Relation to magmas

The *pa* embodies a point noted by Huntington in 1933: Boolean algebra requires, in addition to one unary operation, one, and not two, binary operations. Hence the seldom-noted fact that Boolean algebras are magmas. (Magmas were called groupoids until the latter term was appropriated by category theory.) To see this, note that the *pa* is a commutative:

- Semigroup because *pa* juxtaposition commutes and associates;
- Monoid with identity element (\emptyset) , by virtue of **J0**.

Groups also require a unary operation, called inverse, the group counterpart of Boolean complementation. Let (a) denote the inverse of a . Let (\emptyset) denote the group identity element. Then groups and the *pa* have the same signatures, namely they are both $\langle \neg, -, (\emptyset) \rangle$ algebras of type $\langle 2, 1, 0 \rangle$. Hence the *pa* is a boundary algebra. The axioms for an abelian group, in boundary notation, are:

- **G1.** $abc = acb$ (assuming association from the left);
- **G2.** $(\emptyset)a = a$;
- **G3.** $(a)a = (\emptyset)$.

From **G1** and **G2**, the commutativity and associativity of concatenation may be derived, as above. Note that **G3** and **J1a** are identical. **G2** and **J0** would be identical if $(\emptyset)=()$ replaced **A2**. This is the defining arithmetical identity of group theory, in boundary notation.

The *pa* differs from an abelian group in two ways:

- From **A2**, it follows that $(\emptyset) \neq ()$. If the *pa* were a group, $(\emptyset)=()$ would hold, and one of $(a)a=(\emptyset)$ or $a(\emptyset)=a$ would have to be a *pa* consequence. Note that (\emptyset) and (\emptyset) are mutual *pa* complements, as group theory requires, so that $((\emptyset)) = (\emptyset)$ is true of both group theory and the *pa*;
- **C2** most clearly demarcates the *pa* from other magmas, because **C2** enables demonstrating the absorption law that defines lattices, and the distributive law central to Boolean algebra.

Both **A2** and **C2** follow from *B*'s being an ordered set.

Equations of the second degree (Chapter 11)

Chapter 11 of *LoF* introduces *equations of the second degree*, composed of recursive formulae that can be seen as having "infinite" depth. Some recursive formulae simplify to the marked or unmarked state. Others "oscillate" indefinitely between the two states depending on whether a given depth is even or odd. Specifically, certain recursive formulae can be interpreted as oscillating between **true** and **false** over successive intervals of time, in which case a formula is deemed to have an "imaginary" truth value. Thus the flow of time may be introduced into the *pa*.

Turney (1986) shows how these recursive formulae can be interpreted via Alonzo Church's Restricted Recursive Arithmetic (RRA). Church introduced RRA in 1955 as an axiomatic formalization of finite automata. Turney (1986) presents a general method for translating equations of the second degree into Church's RRA, illustrating his method using the formulae **E1**, **E2**, and **E4** in chapter 11 of *LoF*. This translation into RRA sheds light on the names Spencer-Brown gave to **E1** and **E4**, namely "memory" and "counter". RRA thus formalizes and clarifies *LoF*'s notion of an imaginary truth value.

Resonances in religion, philosophy, and science

The mathematical and logical content of *LoF* is wholly consistent with a secular point of view. Nevertheless, *LoF*'s "first distinction", and the Notes to its chapter 12, bring to mind the following landmarks in religious belief, and in philosophical and scientific reasoning, presented in rough historical order:

- Vedic, Hindu and Buddhist: Related ideas can be noted in the ancient Vedic Upanishads, which form the monastic foundations of Hinduism and later Buddhism. As stated in the *Aitareya Upanishad* ("The Microcosm of Man"), the Supreme Atman manifests itself as the objective Universe from one side, and as the subjective individual from the other side. In this process, things which are *effects* of God's creation become *causes* of our perceptions, by a reversal of the process. In the *Svetasvatara Upanishad*, the core concept of Vedicism and Monism is "Thou art That."
- Taoism, (Chinese Traditional Religion): "...The Tao that can be told is not the eternal Tao; The name that can be named is not the eternal name. The nameless is the beginning of heaven and earth..." (*Tao Te Ching*).
- Zoroastrianism: "This I ask Thee, tell me truly, Ahura. What artist made light and darkness?" (*Gathas* 44.5)
- Judaism (from the Tanakh, called Old Testament by Christians): "In the beginning when God created the heavens and the earth, the earth was a formless *void*... Then God said, 'Let there be light'; and there was light. ...God *separated* the light from the darkness. God called the light Day, and the darkness he called Night.

"...And God said, 'Let there be a dome in the midst of the waters, and let it *separate* the waters from the waters.' So God made the dome and *separated* the waters that were under the dome from the waters that were above the dome.

"...And God said, 'Let the waters under the sky be gathered together into one place, and let the dry land appear.' ...God called the dry land Earth, and the waters that were gathered together he called Seas.

"...And God said, 'Let there be lights in the dome of the sky to *separate* the day from the night...' God made the two great lights... to *separate* the light from the darkness." (*Genesis* 1:1-18; Revised Standard Version, emphasis added).

"And the whole earth was of *one* language, and of *one* speech." (*Genesis* 11:1; emphasis added).

"I am; that is who I am." (*Exodus* 3:14)

- Confucianism: Confucius claimed that he sought "a unity all pervading" (*Analects* XV.3) and that there was "one single thread binding my way together." (*Ana.* IV.15). The *Analects* also contain the following remarkable passage on how the social, moral, and aesthetic orders are grounded in right language, grounded in turn in the ability to "rectify names," i.e., to make correct distinctions: "Zilu said, 'What would be master's priority?' The master replied, "Rectifying names! ...If names are not rectified then language will not flow. If language does not flow, then affairs cannot be completed. If affairs are not completed, ritual and music will not flourish. If ritual and music do not flourish, punishments and penalties will miss their mark. When punishments and penalties miss their mark, people lack the wherewithal to control hand and foot." (*Ana.* XIII.3)
- Heraclitus: Pre-socratic philosopher, credited with forming the idea of logos. "He who hears not me but the *logos* will say: *All is one*." Further: "I am as I am not."
- Parmenides: Argued that the every-day perception of reality of the physical world is mistaken, and that the reality of the world is 'One Being': an unchanging, ungenerated, indestructible whole.
- Plato: *Logos* is also a fundamental technical term in the Platonic worldview.
- Christianity: "In the Beginning was the Word, and the Word was with God, and the Word was God." (*John* 1:1). "Word" translates *logos* in the koine original. "If you do not believe *that I am*, you will die in your sins." (*John* 8:24). "The Father and I are *one*." (*John* 10:30). "That they all may be *one*; as thou, Father, art in me, and I in thee, that they may also be *one* in us: that the world may believe that thou has sent me." (*John* 17:21). (emphases added)

- Object relations theory, psychodynamics: The primary separation experienced by infants between self and other objects, distinguishing of reality from phantasy.
- Islamic philosophy distinguishes essence (*Dhat*) from attribute (*Sifat*), which are neither identical nor separate.
- Leibniz: "All creatures derive from God and from nothingness. Their self-being is of God, their nonbeing is of nothing. Numbers too show this in a wonderful way, and the essences of things are like numbers. No creature can be without nonbeing; otherwise it would be God... The only self-knowledge is to distinguish well between our self-being and our nonbeing... Within our selfbeing there lies an infinity, a footprint or reflection of the omniscience and omnipresence of God."^[7]
- Josiah Royce: "Without negation, there is no inference. Without inference, there is no order, in the strictly logical sense of the word. The fundamentally significant position of the idea of negation in determining and controlling our idea of the orderliness of both the natural and the spiritual order, becomes, in the light of all these considerations, as momentous as it is, in our ordinary popular views of this subject, neglected. ...From this point of view, negation appears as one of the most significant ideas that lie at the base of all the exact sciences. By virtue of the idea of negation we are able to define processes of inference-processes which, in their abstract form, the purely mathematical sciences illustrate, and which, in their natural expression, the laws of the physical world, as known to our inductive science, exemplify."

"When logically analyzed, order turns out to be something that would be inconceivable and incomprehensible to us unless we had the idea which is expressed by the term 'negation'. Thus it is that negation, which is always also something intensely positive, not only aids us in giving order to life, and in finding order in the world, but logically determines the very essence of order."^[8]

Returning to the Bible, the injunction "Let there be light" conveys:

- "... and there was light" — the light itself;
- "... called the light Day" — the manifestation of the light;
- "... morning and evening" — the boundaries of the light.

A Cross denotes a distinction made, and the absence of a Cross means that no distinction has been made. In the Biblical example, light is distinct from the void – the absence of light. The Cross and the Void are, of course, the two primitive values of the Laws of Form.

Related work

Gottfried Leibniz, in memoranda not published before the late 19th and early 20th centuries, invented Boolean logic. His notation was isomorphic to that of *LoF*: concatenation read as conjunction, and "non-(X)" read as the complement of X. Leibniz's pioneering role in algebraic logic was foreshadowed by Lewis (1918) and Rescher (1954). But a full appreciation of Leibniz's accomplishments had to await the work of Wolfgang Lenzen, published in the 1980s and reviewed in Lenzen (2004).^[9]

Charles Sanders Peirce (1839–1914) anticipated the *pa* in three veins of work:

1. Two papers he wrote in 1886 proposed a logical algebra employing but one symbol, the *streamer*, nearly identical to the Cross of *LoF*. The semantics of the streamer are identical to those of the Cross, except that Peirce never wrote a streamer with nothing under it. An excerpt from one of these papers was published in 1976,^[10] but they were not published in full until 1993.^[11]
2. In a 1902 encyclopedia article,^[12] Peirce notated Boolean algebra and sentential logic in the manner of this entry, except that he employed two styles of brackets, toggling between '(', ')' and '[', ']' with each increment in formula depth.
3. The syntax of his alpha existential graphs is merely concatenation, read as conjunction, and enclosure by ovals, read as negation.^[13] If *pa* concatenation is read as conjunction, then these graphs are isomorphic to the *pa* (Kauffman 2001).^[14]

Ironically, *LoF* cites vol. 4 of Peirce's *Collected Papers*, the source for the formalisms in (2) and (3) above. (1)-(3) were virtually unknown at the time when (1960s) and in the place where (UK) *LoF* was written. Peirce's semiotics, about which *LoF* is silent, may yet shed light on the philosophical aspects of *LoF*.

Kauffman (2001)^[14] discusses another notation similar to that of *LoF*, that of a 1917 article by Jean Nicod, who was a disciple of Bertrand Russell's.

The above formalisms are, like the *pa*, all instances of *boundary mathematics*, i.e., mathematics whose syntax is limited to letters and brackets (enclosing devices). A minimalist syntax of this nature is a "boundary notation." Boundary notation is free of infix, prefix, or postfix operator symbols. The very well known curly braces ('{', '}') of set theory can be seen as a boundary notation.

The work of Leibniz, Peirce, and Nicod is innocent of metatheory, as they wrote before Emil Post's landmark 1920 paper (which *LoF* cites), proving that sentential logic is complete, and before Hilbert and Lukasiewicz showed how to prove axiom independence using models.

Craig (1979) argued that the world, and how humans perceive and interact with that world, has a rich Boolean structure. Craig was an orthodox logician and an authority on algebraic logic.

Second-generation cognitive science emerged in the 1970s, after *LoF* was written. On cognitive science and its relevance to Boolean algebra, logic, and set theory, see Lakoff (1987) (see index entries under "Image schema examples: container") and Lakoff and Núñez (2001). Neither book cites *LoF*.

The biologists and cognitive scientists Humberto Maturana and his student Francisco Varela both discuss *LoF* in their writings, which identify "distinction" as the fundamental cognitive act. The Berkeley psychologist and cognitive scientist Eleanor Rosch has written extensively on the closely related notion of categorization.

The Multiple Form Logic^[15], by G.A. Stathis, "generalises [the primary algebra] into Multiple Truth Values" so as to be "more consistent with Experience." Multiple Form Logic, which is *not* a boundary formalism, employs two primitive binary operations: concatenation, read as Boolean OR, and infix "#", read as XOR. The primitive values are 0 and 1, and the corresponding arithmetic is $11=1$ and $1\#1=0$. The axioms are $1A=1$, $A\#X\#X = A$, and $A(X\#(AB)) = A(X\#B)$.

Other formal systems with possible affinities to the primary algebra include:

- Mereology which typically has a lattice structure very similar to that of Boolean algebra. For a few authors, mereology is simply a model of Boolean algebra and hence of the primary algebra as well.
- Mereotopology, which is inherently richer than Boolean algebra;
- The system of Whitehead (1934), whose fundamental primitive is "indication."

The primary arithmetic and algebra are a minimalist formalism for sentential logic and Boolean algebra. Other minimalist formalisms having the power of set theory include:

- The lambda calculus;
- Combinatory logic with two (**S** and **K**) or even one (**X**) primitive combinators;
- Mathematical logic done with merely three primitive notions: one connective, NAND (whose *pa* translation is (AB) or—dually $\neg(A)(B)$), universal quantification, and one binary atomic formula, denoting set membership. This is the system of Quine (1951).
- The *beta* existential graphs, with a single binary predicate denoting set membership. This has yet to be explored. The *alpha* graphs mentioned above are a special case of the *beta* graphs.

Notes

- [1] <http://www.canterbury.ac.nz/spark/Researcher.aspx?researcherid=84894>
- [2] Meguire, P. (2011) Boundary Algebra: A Simpler Approach to Basic Logic and Boolean Algebra. Saarbrücken: VDM Publishing Ltd. 168pp
- [3] Felix Lau: "Die Form der Paradoxie", 2005 Carl-Auer Verlag, ISBN 9783896703521
- [4] <http://projecteuclid.org/Dienst/UI/1.0/Summarize/euclid.ndjfl/1093888028?abstract=true>
- [5] For a sympathetic evaluation, see Kauffman (2001) (<http://www.arxiv.org/math.CO/0112266>).
- [6] http://toolserver.org/%7Edispenser/cgi-bin/dab_solver.py?page=Laws_of_Form&editintro=Template:Disambiguation_needed&editintro&client=Template:Dn
- [7] "On the True *Theologia Mystica*" in Loemker, Leroy, ed. and trans., 1969. *Leibniz: Philosophical Papers and Letters*. Reidel: 368.
- [8] "Order" in Hasting, J., ed., 1917. *Encyclopedia of Religion and Ethics*. Scribner's: 540. Reprinted in Robinson, D. S., ed., 1951, *Royce's Logical Essays*. Dubuque IA: Wm. C. Brown: 230-31.
- [9] <http://www.philosopie.uni-osnabrueck.de/Publikationen%20Lenzen/Lenzen%20Leibniz%20Logic.pdf>
- [10] "Qualitative Logic", MS 736 (c. 1886) in Eisele, Carolyn, ed. 1976. *The New Elements of Mathematics by Charles S. Peirce. Vol. 4, Mathematical Philosophy*. (The Hague) Mouton: 101-15.1
- [11] "Qualitative Logic", MS 582 (1886) in Kloesel, Christian et al., eds., 1993. *Writings of Charles S. Peirce: A Chronological Edition, Vol. 5, 1884-1886*. Indiana University Press: 323-71. "The Logic of Relatives: Qualitative and Quantitative", MS 584 (1886) in Kloesel, Christian et al., eds., 1993. *Writings of Charles S. Peirce: A Chronological Edition, Vol. 5, 1884-1886*. Indiana University Press: 372-78.
- [12] Reprinted in Peirce, C.S. (1933) *Collected Papers of Charles Sanders Peirce, Vol. 4*, Charles Hartshorne and Paul Weiss, eds. Harvard University Press. Paragraphs 378-383
- [13] The existential graphs are described at length in Peirce, C.S. (1933) *Collected Papers, Vol. 4*, Charles Hartshorne and Paul Weiss, eds. Harvard University Press. Paragraphs 347-529.
- [14] <http://www2.math.uic.edu/~kauffman/CHK.pdf>
- [15] <http://multiforms.netfirms.com>

References

- Editions of *Laws of Form*:
 - 1969. London: Allen & Unwin, hardcover.
 - 1972. Crown Publishers, hardcover: ISBN 0-517-52776-6
 - 1973. Bantam Books, paperback. ISBN 0-553-07782-1
 - 1979. E.P. Dutton, paperback. ISBN 0-525-47544-3
 - 1994. Portland OR: Cognizer Company, paperback. ISBN 0-9639899-0-1
 - 1997 German translation, titled *Gesetze der Form*. Lübeck: Bohmeier Verlag. ISBN 3-89094-321-7
- Bostock, David, 1997. *Intermediate Logic*. Oxford Univ. Press.
- Byrne, Lee, 1946, "Two Formulations of Boolean Algebra," *Bulletin of the American Mathematical Society*: 268-71.
- Craig, William (1979). "Boolean Logic and the Everyday Physical World". *Proceedings and Addresses of the American Philosophical Association* 52 (6): 751–78. doi: 10.2307/3131383 (<http://dx.doi.org/10.2307/3131383>). JSTOR 3131383 (<http://www.jstor.org/stable/3131383>).
- David Gries, and Schneider, F B, 1993. *A Logical Approach to Discrete Math*. Springer-Verlag.
- William Ernest Johnson, 1892, "The Logical Calculus," *Mind* 1 (n.s.): 3-30.
- Louis H. Kauffman (<http://www.math.uic.edu/~kauffman/>), 2001, "The Mathematics of C.S. Peirce (<http://www2.math.uic.edu/~kauffman/CHK.pdf>)", *Cybernetics and Human Knowing* 8: 79-110.
- -----, 2006, "Reformulating the Map Color Theorem. (<http://www.arxiv.org/math.CO/0112266>)"
- -----, 2006a. "Laws of Form - An Exploration in Mathematics and Foundations. (<http://www.math.uic.edu/~kauffman/Laws.pdf>)" Book draft (hence big).
- Lenzen, Wolfgang, 2004, "Leibniz's Logic (http://www.philosopie.uni-osnabrueck.de/Publikationen_Lenzen/Lenzen_Leibniz_Logic.pdf)" in Gabbay, D., and Woods, J., eds., *The Rise of Modern Logic: From Leibniz to Frege (Handbook of the History of Logic – Vol. 3)*. Amsterdam: Elsevier, 1-83.
- Lakoff, George, 1987. *Women, Fire, and Dangerous Things*. University of Chicago Press.

- ----- and Rafael E. Núñez, 2001. *Where Mathematics Comes From: How the Embodied Mind Brings Mathematics into Being*. Basic Books.
- Meguire, P. G., 2003, "Discovering Boundary Algebra: A Simplified Notation for Boolean Algebra and the Truth Functors," *International Journal of General Systems* 32: 25-87.
- -----, 2011. *Boundary Algebra: A Simpler Approach to Basic Logic and Boolean Algebra*. VDM Publishing Ltd. ISBN 978-3639367492. The source for much of this entry, including the notation which encloses in parentheses what *LoF* places under a cross. Steers clear of the more speculative aspects of *LoF*.
- Willard Quine, 1951. *Mathematical Logic*, 2nd ed. Harvard University Press.
- -----, 1982. *Methods of Logic*, 4th ed. Harvard University Press.
- Rescher, Nicholas (1954). "Leibniz's Interpretation of His Logical Calculi". *Journal of Symbolic Logic* 18: 1–13.
- Schwartz, Daniel G. (1981). "Isomorphisms of G. Spencer-Brown's *Laws of Form* and F. Varela's Calculus for Self-Reference". *International Journal of General Systems* 6 (4): 239–55. doi: 10.1080/03081078108934802 (<http://dx.doi.org/10.1080/03081078108934802>).
- Turney, P. D. (1986). "Laws of Form and Finite Automata". *International Journal of General Systems* 12 (4): 307–18. doi: 10.1080/03081078608934939 (<http://dx.doi.org/10.1080/03081078608934939>).
- A. N. Whitehead, 1934, "Indication, classes, number, validation," *Mind* 43 (n.s.): 281-97, 543. The corrigenda on p. 543 are numerous and important, and later reprints of this article do not incorporate them.

External links

- *Laws of Form* web site (<http://www.lawsofform.org/>), by Richard Shoup.
- Spencer-Brown's talks at Esalen, 1973. (<http://www.lawsofform.org/aum/session1.html>) Self-referential forms are introduced in the section entitled "Degree of Equations and the Theory of Types."
- Louis H. Kauffman, (<http://www.math.uic.edu/~kauffman/>) " Box Algebra, Boundary Mathematics, Logic, and Laws of Form. (<http://www.math.uic.edu/~kauffman/Arithmetic.htm>)"
- Kissel, Matthias, " A nonsystematic but easy to understand introduction to *Laws of Form*. (http://web.archive.org/web/20070310071916/http://de.geocities.com/matthias_kissel/gdf/LoF.html)"
- The Multiple Form Logic (<http://multiforms.netfirms.com>), by G.A. Stathis, owes much to the primary algebra.
- The Laws of Form Forum (<http://groups.yahoo.com/group/lawsofform>), where the primary algebra and related formalisms have been discussed since 2002.
- A meeting with G.S.B (<http://www.omath.org.il/112431/4CT>) by Moshe Klein

Logical graph

A **logical graph** is a special type of diagrammatic structure in any one of several systems of graphical syntax that Charles Sanders Peirce developed for logic.

In his papers on *qualitative logic*, *entitative graphs*, and *existential graphs*, Peirce developed several versions of a graphical formalism, or a graph-theoretic formal language, designed to be interpreted for logic.

In the century since Peirce initiated this line of development, a variety of formal systems have branched out from what is abstractly the same formal base of graph-theoretic structures.

External links

 Media related to Logical graphs at Wikimedia Commons

- Logical Graph ^[1] @ Commens Dictionary of Peirce's Terms ^[2]
- Existential Graphs ^[3], Jay Zeman, ed., U. of Florida. With 4 works by Peirce.
- Frithjof Dau's page of readings and links on existential graphs ^[4] includes lists of: books exclusively on existential graphs; books containing existential graphs; articles; and some links and downloadables.
- The literature of C.S. Peirce's Existential Graphs (via Internet Archive) ^[5], Xin-Wen Liu, Institute of Philosophy, Chinese Academy of Social Sciences, Beijing, PRC. A whole lot there.

References

[1] <http://www.helsinki.fi/science/commens/terms/graphlogi.html>

[2] <http://www.helsinki.fi/science/commens/dictionary.html>

[3] <http://www.existentialgraphs.com/>

[4] http://dr-dau.net/eg_readings.shtml

[5] <http://web.archive.org/web/20071217055005/http://philosophy.cass.cn/facu/liuxinwen/01.htm>

Examples of Boolean algebras

Boolean domain

In mathematics and abstract algebra, a **Boolean domain** is a set consisting of exactly two elements whose interpretations include *false* and *true*. In logic, mathematics and theoretical computer science, a Boolean domain is usually written as $\{0, 1\}$,^[1] $\{ \text{false}, \text{true} \}$, $\{\text{F}, \text{T}\}$,^[4] or $\{\perp, \top\}$.^[5]

The algebraic structure that naturally builds on a Boolean domain is the Boolean algebra with two elements. The initial object in the category of bounded lattices is a Boolean domain.

In computer science, a Boolean variable is a variable that takes values in some Boolean domain. Some programming languages feature reserved words or symbols for the elements of the Boolean domain, for example `false` and `true`. However, many programming languages do not have a Boolean datatype in the strict sense. In C or BASIC, for example, falsity is represented by the number 0 and truth is represented by the number 1 or -1 respectively, and all variables that can take these values can also take any other numerical values.

Generalizations

The Boolean domain $\{0, 1\}$ can be replaced by the unit interval $[0,1]$, in which case rather than only taking values 0 or 1, any value between and including 0 and 1 can be assumed. Algebraically, negation (NOT) is replaced with $1 - x$, conjunction (AND) is replaced with multiplication (xy), and disjunction (OR) is defined via De Morgan's law to be $1 - (1 - x)(1 - y)$.

Interpreting these values as logical truth values yields a multi-valued logic, which forms the basis for fuzzy logic and probabilistic logic. In these interpretations, a value is interpreted as the "degree" of truth – to what extent a proposition is true, or the probability that the proposition is true.

Notes

- [1] Dirk van Dalen, *Logic and Structure*. Springer (2004), page 15.
- [2] David Makinson, *Sets, Logic and Maths for Computing*. Springer (2008), page 13.
- [3] George S. Boolos and Richard C. Jeffrey, *Computability and Logic*. Cambridge University Press (1980), page 99.
- [4] Elliott Mendelson, *Introduction to Mathematical Logic (4th. ed.)*. Chapman & Hall/CRC (1997), page 11.
- [5] Eric C. R. Hehner, *A Practical Theory of Programming*. Springer (1993, 2010), page 3.

Boolean ring

In mathematics, a **Boolean ring** R is a ring for which $x^2 = x$ for all x in R ; that is, R consists only of idempotent elements.

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

Notations

There are at least four different and incompatible systems of notation for Boolean rings and algebras.

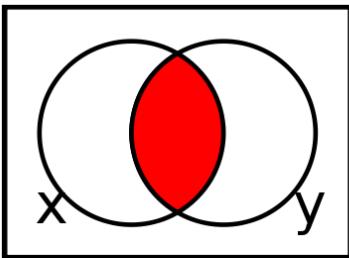
- In commutative algebra the standard notation is to use $x + y = (x \wedge \neg y) \vee (\neg x \wedge y)$ for the ring sum of x and y , and use $xy = x \wedge y$ for their product.
- In logic, a common notation is to use $x \wedge y$ for the meet (same as the ring product) and use $x \vee y$ for the join, given in terms of ring notation (given just above) by $x + y + xy$.
- In set theory and logic it is also common to use $x \cdot y$ for the meet, and $x + y$ for the join $x \vee y$. This use of $+$ is different from the use in ring theory.
- A rare convention is to use xy for the product and $x \oplus y$ for the ring sum, in an effort to avoid the ambiguity of $+$.

The old terminology was to use "Boolean ring" to mean a "Boolean ring possibly without an identity", and "Boolean algebra" to mean a Boolean ring with an identity. (This is the same as the old use of the terms "ring" and "algebra" in measure theory) (Also note that, when a Boolean ring has an identity, then a complement operation becomes definable on it, and a key characteristic of the modern definitions of both Boolean algebra and sigma-algebra is that they have complement operations.)

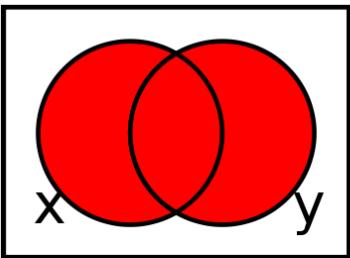
Examples

One example of a Boolean ring is the power set of any set X , where the addition in the ring is symmetric difference, and the multiplication is intersection. As another example, we can also consider the set of all finite or cofinite subsets of X , again with symmetric difference and intersection as operations. More generally with these operations any field of sets is a Boolean ring. By Stone's representation theorem every Boolean ring is isomorphic to a field of sets (treated as a ring with these operations).

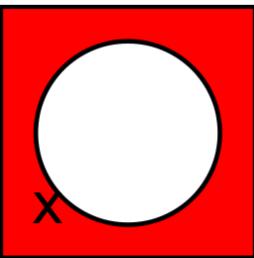
Relation to Boolean algebras



$x \wedge y$



$x \vee y$



$\neg x$

Venn diagrams for the Boolean operations of conjunction, disjunction, and complement

Since the join operation \vee in a Boolean algebra is often written additively, it makes sense in this context to denote ring addition by \oplus , a symbol that is often used to denote exclusive or.

Given a Boolean ring R , for x and y in R we can define

$$\begin{aligned}x \wedge y &= xy, \\x \vee y &= x \oplus y \oplus xy, \\\neg x &= 1 \oplus x.\end{aligned}$$

These operations then satisfy all of the axioms for meets, joins, and complements in a Boolean algebra. Thus every Boolean ring becomes a Boolean algebra. Similarly, every Boolean algebra becomes a Boolean ring thus:

$$\begin{aligned}xy &= x \wedge y, \\x \oplus y &= (x \vee y) \wedge \neg(x \wedge y).\end{aligned}$$

If a Boolean ring is translated into a Boolean algebra in this way, and then the Boolean algebra is translated into a ring, the result is the original ring. The analogous result holds beginning with a Boolean algebra.

A map between two Boolean rings is a ring homomorphism if and only if it is a homomorphism of the corresponding Boolean algebras. Furthermore, a subset of a Boolean ring is a ring ideal (prime ring ideal, maximal ring ideal) if and only if it is an order ideal (prime order ideal, maximal order ideal) of the Boolean algebra. The quotient ring of a Boolean ring modulo a ring ideal corresponds to the factor algebra of the corresponding Boolean algebra modulo the corresponding order ideal.

Properties of Boolean rings

Every Boolean ring R satisfies $x \oplus x = 0$ for all x in R , because we know

$$x \oplus x = (x \oplus x)^2 = x^2 \oplus x^2 \oplus x^2 \oplus x^2 = x \oplus x \oplus x \oplus x$$

and since $\langle R, \oplus \rangle$ is an abelian group, we can subtract $x \oplus x$ from both sides of this equation, which gives $x \oplus x = 0$. A similar proof shows that every Boolean ring is commutative:

$$x \oplus y = (x \oplus y)^2 = x^2 \oplus xy \oplus yx \oplus y^2 = x \oplus xy \oplus yx \oplus y$$

and this yields $xy \oplus yx = 0$, which means $xy = yx$ (using the first property above).

The property $x \oplus x = 0$ shows that any Boolean ring is an associative algebra over the field \mathbf{F}_2 with two elements, in just one way. In particular, any finite Boolean ring has as cardinality a power of two. Not every associative algebra with one over \mathbf{F}_2 is a Boolean ring: consider for instance the polynomial ring $\mathbf{F}_2[X]$.

The quotient ring R/I of any Boolean ring R modulo any ideal I is again a Boolean ring. Likewise, any subring of a Boolean ring is a Boolean ring.

Every prime ideal P in a Boolean ring R is maximal: the quotient ring R/P is an integral domain and also a Boolean ring, so it is isomorphic to the field \mathbf{F}_2 , which shows the maximality of P . Since maximal ideals are always prime, prime ideals and maximal ideals coincide in Boolean rings.

Boolean rings are von Neumann regular rings.

Boolean rings are absolutely flat: this means that every module over them is flat.

Every finitely generated ideal of a Boolean ring is principal (indeed, $(x,y) = (x+y+xy)$).

Unification in Boolean rings is decidable, that is, algorithms exist to solve arbitrary equations over Boolean rings.

Notes

References

- Atiyah, Michael Francis; Macdonald, I.G. (1969), *Introduction to Commutative Algebra*, Westview Press, ISBN 978-0-201-40751-8
- Fraleigh, John B. (1976), *A First Course In Abstract Algebra* (2nd ed.), Reading: Addison-Wesley, ISBN 0-201-01984-1
- Herstein, I. N. (1964), *Topics In Algebra*, Waltham: Blaisdell Publishing Company, ISBN 978-1114541016
- McCoy, Neal H. (1968), *Introduction To Modern Algebra, Revised Edition*, Boston: Allyn and Bacon, LCCN 68-15225 (<http://lccn.loc.gov/68-15225>)
- Ryabukhin, Yu.M. (2001), "Boolean_ring" (http://www.encyclopediaofmath.org/index.php?title=Boolean_ring), in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

Goodman–Nguyen–van Fraassen algebra

A **Goodman–Nguyen–van Fraassen algebra** is a type of conditional event algebra (CEA) that embeds the standard Boolean algebra of unconditional events in a larger algebra which is itself Boolean. The goal (as with all CEAs) is to equate the conditional probability $P(A \cap B) / P(A)$ with the probability of a conditional event, $P(A \rightarrow B)$ for more than just trivial choices of A , B , and P .

Construction of the algebra

Given set Ω , which is the set of possible outcomes, and set F of subsets of Ω —so that F is the set of possible events—consider an infinite Cartesian product of the form $E_1 \times E_2 \times \dots \times E_n \times \Omega \times \Omega \times \Omega \times \dots$, where E_1, E_2, \dots, E_n are members of F . Such a product specifies the set of all infinite sequences whose first element is in E_1 , whose second element is in E_2 , ..., and whose n th element is in E_n , and all of whose elements are in Ω . Note that one such product is the one where $E_1 = E_2 = \dots = E_n = \Omega$, i.e., the set $\Omega \times \Omega \times \Omega \times \Omega \times \dots$. Designate this set as $\hat{\Omega}$; it is the set of all infinite sequences whose elements are in Ω .

A new Boolean algebra is now formed, whose elements are subsets of $\hat{\Omega}$. To begin with, any event which was formerly represented by subset A of Ω is now represented by $\hat{A} = A \times \Omega \times \Omega \times \Omega \times \dots$.

Additionally, however, for events A and B , let the conditional event $A \rightarrow B$ be represented as the following infinite union of disjoint sets:

$$\begin{aligned} & [(A \cap B) \times \Omega \times \Omega \times \Omega \times \dots] \cup \\ & [A' \times (A \cap B) \times \Omega \times \Omega \times \Omega \times \dots] \cup \\ & [A' \times A' \times (A \cap B) \times \Omega \times \Omega \times \Omega \times \dots] \cup \dots \end{aligned}$$

The motivation for this representation of conditional events will be explained shortly. Note that the construction can be iterated; A and B can themselves be conditional events.

Intuitively, unconditional event A ought to be representable as conditional event $\Omega \rightarrow A$. And indeed: because $\Omega \cap A = A$ and $\Omega' = \emptyset$, the infinite union representing $\Omega \rightarrow A$ reduces to $A \times \Omega \times \Omega \times \Omega \times \dots$.

Let \hat{F} now be a set of subsets of $\hat{\Omega}$, which contains representations of all events in F and is otherwise just large enough to be closed under construction of conditional events and under the familiar Boolean operations. \hat{F} is a Boolean algebra of conditional events which contains a Boolean algebra corresponding to the algebra of ordinary events.

Definition of the extended probability function

Corresponding to the newly constructed logical objects, called conditional events, is a new definition of a probability function, \hat{P} , based on a standard probability function P :

$$\hat{P}(E_1 \times E_2 \times \dots \times E_n \times \Omega \times \Omega \times \Omega \times \dots) = P(E_1) \cdot P(E_2) \cdot \dots \cdot P(E_n) \cdot P(\Omega) \cdot P(\Omega) \cdot P(\Omega) \cdot \dots = P(E_1) \cdot P(E_2) \cdot \dots \cdot P(E_n), \text{ since } P(\Omega) = 1.$$

It follows from the definition of \hat{P} that $\hat{P}(\hat{A}) = P(A)$. Thus $\hat{P} = P$ over the domain of P .

$P(A \rightarrow B) = P(B|A)$

Now comes the insight which motivates all of the preceding work. For P , the original probability function, $P(A') = 1 - P(A)$, and therefore $P(B|A) = P(A \cap B) / P(A)$ can be rewritten as $P(A \cap B) / [1 - P(A')]$. The factor $1 / [1 - P(A')]$, however, can in turn be represented by its Maclaurin series expansion, $1 + P(A') + P(A')^2 + \dots$. Therefore, $P(B|A) = P(A \cap B) + P(A')P(A \cap B) + P(A')^2P(A \cap B) + \dots$

The right side of the equation is exactly the expression for the probability \hat{P} of $A \rightarrow B$, just defined as a union of carefully chosen disjoint sets. Thus that union can be taken to represent the conditional event $A \rightarrow B$, such that $\hat{P}(A \rightarrow B) = P(B|A)$ for any choice of A , B , and P . But since $\hat{P} = P$ over the domain of P , the hat notation is optional. So long as the context is understood (i.e., conditional event algebra), one can write $P(A \rightarrow B) = P(B|A)$, with P now being the extended probability function.

References

- Bamber, Donald, I. R. Goodman, and H. T. Nguyen. 2004. "Deduction from Conditional Knowledge." *Soft Computing* 8: 247–255.
- Goodman, I. R., R. P. S. Mahler, and H. T. Nguyen. 1999. "What is conditional event algebra and why should you care?" *SPIE Proceedings*, Vol 3720.

Interior algebra

In abstract algebra, an **interior algebra** is a certain type of algebraic structure that encodes the idea of the topological interior of a set. Interior algebras are to topology and the modal logic **S4** what Boolean algebras are to set theory and ordinary propositional logic. Interior algebras form a variety of modal algebras.

Definition

An **interior algebra** is an algebraic structure with the signature

$$\llbracket S, \cdot, +, ', 0, 1 \rrbracket$$

where

$$\llbracket S, \cdot, +, ', 0, 1 \rrbracket$$

is a Boolean algebra and postfix I designates a unary operator, the **interior operator**, satisfying the identities:

1. $x^I \leq x$
2. $x^{II} = x^I$
3. $(xy)^I = x^Iy^I$
4. $1^I = 1$

x^I is called the **interior** of x .

The dual of the interior operator is the **closure operator** C defined by $x^C = ((x')^I)'$. x^C is called the **closure** of x . By the principle of duality, the closure operator satisfies the identities:

1. $x^C \geq x$
2. $x^{CC} = x^C$
3. $(x + y)^C = x^C + y^C$
4. $0^C = 0$

If the closure operator is taken as primitive, the interior operator can be defined as $x^I = ((x')^C)'$. Thus the theory of interior algebras may be formulated using the closure operator instead of the interior operator, in which case one considers **closure algebras** of the form $\llbracket S, \cdot, +, ', 0, 1, ^C \rrbracket$, where $\llbracket S, \cdot, +, ', 0, 1 \rrbracket$ is again a Boolean algebra and C satisfies the above identities for the closure operator. Closure and interior algebras form dual pairs, and are paradigmatic instances of "Boolean algebras with operators." The early literature on this subject (mainly Polish topology) invoked closure operators, but the interior operator formulation eventually became the norm.

Open and closed elements

Elements of an interior algebra satisfying the condition $x^I = x$ are called **open**. The complements of open elements are called **closed** and are characterized by the condition $x^C = x$. An interior of an element is always open and the closure of an element is always closed. Interiors of closed elements are called **regular open** and closures of open elements are called **regular closed**. Elements which are both open and closed are called **clopen**. 0 and 1 are clopen.

An interior algebra is called **Boolean** if all its elements are open (and hence clopen). Boolean interior algebras can be identified with ordinary Boolean algebras as their interior and closure operators provide no meaningful additional structure. A special case is the class of **trivial** interior algebras which are the single element interior algebras characterized by the identity $0 = 1$.

Morphisms of interior algebras

Homomorphisms

Interior algebras, by virtue of being algebraic structures, have homomorphisms. Given two interior algebras A and B , a map $f : A \rightarrow B$ is an **interior algebra homomorphism** if and only if f is a homomorphism between the underlying Boolean algebras of A and B , that also preserves interiors and closures. Hence:

- $f(x^I) = f(x)^I$;
- $f(x^C) = f(x)^C$.

Topomorphisms

Topomorphisms are another important, and more general, class of morphisms between interior algebras. A map $f : A \rightarrow B$ is a topomorphism if and only if f is a homomorphism between the Boolean algebras underlying A and B , that also preserves the open and closed elements of A . Hence:

- If x is open in A , then $f(x)$ is open in B ;
- If x is closed in A , then $f(x)$ is closed in B .

Every interior algebra homomorphism is a topomorphism, but not every topomorphism is an interior algebra homomorphism.

Relationships to other areas of mathematics

Topology

Given a topological space $X = \langle X, T \rangle$ one can form the power set Boolean algebra of X :

$$\langle P(X), \cap, \cup, ', \emptyset, X \rangle$$

and extend it to an interior algebra

$$A(X) = \langle P(X), \cap, \cup, ', \emptyset, X, I \rangle,$$

where I is the usual topological interior operator. For all $S \subseteq X$ it is defined by

$$S^I = \cup \{O : O \subseteq S \text{ and } O \text{ is open in } X\}$$

For all $S \subseteq X$ the corresponding closure operator is given by

$$S^C = \cap \{C : S \subseteq C \text{ and } C \text{ is closed in } X\}$$

S^I is the largest open subset of S and S^C is the smallest closed superset of S in X . The open, closed, regular open, regular closed and clopen elements of the interior algebra $A(X)$ are just the open, closed, regular open, regular closed and clopen subsets of X respectively in the usual topological sense.

Every complete atomic interior algebra is isomorphic to an interior algebra of the form $A(X)$ for some topological space X . Moreover every interior algebra can be embedded in such an interior algebra giving a representation of an interior algebra as a **topological field of sets**. The properties of the structure $A(X)$ are the very motivation for the definition of interior algebras. Because of this intimate connection with topology, interior algebras have also been called **topo-Boolean algebras** or **topological Boolean algebras**.

Given a continuous map between two topological spaces

$$f : X \rightarrow Y$$

we can define a complete topomorphism

$$A(f) : A(Y) \rightarrow A(X)$$

by

$$A(f)(S) = f^{-1}[S]$$

for all subsets S of Y . Every complete topomorphism between two complete atomic interior algebras can be derived in this way. If **Top** is the category of topological spaces and continuous maps and **Cit** is the category of complete atomic interior algebras and complete topomorphisms then **Top** and **Cit** are dually isomorphic and $A : \mathbf{Top} \rightarrow \mathbf{Cit}$ is a contravariant functor that is a dual isomorphism of categories. $A(f)$ is a homomorphism if and only if f is a continuous open map.

Under this dual isomorphism of categories many natural topological properties correspond to algebraic properties, in particular connectedness properties correspond to irreducibility properties:

- X is empty if and only if $A(X)$ is trivial
- X is indiscrete if and only if $A(X)$ is simple
- X is discrete if and only if $A(X)$ is Boolean
- X is almost discrete if and only if $A(X)$ is semisimple
- X is finitely generated (Alexandrov) if and only if $A(X)$ is **operator complete** i.e. its interior and closure operators distribute over arbitrary meets and joins respectively
- X is connected if and only if $A(X)$ is directly indecomposable
- X is ultraconnected if and only if $A(X)$ is finitely subdirectly irreducible
- X is compact ultra-connected if and only if $A(X)$ is subdirectly irreducible

Generalized topology

The modern formulation of topological spaces in terms of topologies of open subsets, motivates an alternative formulation of interior algebras: A **generalized topological space** is an algebraic structure of the form

$$\llbracket B, \cdot, +, ', 0, 1, T \rrbracket$$

where $\llbracket B, \cdot, +, ', 0, 1 \rrbracket$ is a Boolean algebra as usual, and T is a unary relation on B (subset of B) such that:

1. $0, 1 \in T$
2. T is closed under arbitrary joins (i.e. if a join of an arbitrary subset of T exists then it will be in T)
3. T is closed under finite meets
4. For every element b of B , the join $\sum \{a \in T : a \leq b\}$ exists

T is said to be a **generalized topology** in the Boolean algebra.

Given an interior algebra its open elements form a generalized topology. Conversely given a generalized topological space

$$\llbracket B, \cdot, +, ', 0, 1, T \rrbracket$$

we can define an interior operator on B by $b^I = \sum \{a \in T : a \leq b\}$ thereby producing an interior algebra whose open elements are precisely T . Thus generalized topological spaces are equivalent to interior algebras.

Considering interior algebras to be generalized topological spaces, topomorphisms are then the standard homomorphisms of Boolean algebras with added relations, so that standard results from universal algebra apply.

Neighbourhood functions and neighbourhood lattices

The topological concept of neighbourhoods can be generalized to interior algebras: An element y of an interior algebra is said to be a **neighbourhood** of an element x if $x \leq y^I$. The set of neighbourhoods of x is denoted by $N(x)$ and forms a filter. This leads to another formulation of interior algebras:

A **neighbourhood function** on a Boolean algebra is a mapping N from its underlying set B to its set of filters, such that:

1. For all $x \in B$, $\max\{y \in B : x \in N(y)\}$ exists
2. For all $x, y \in B$, $x \in N(y)$ if and only if there is a $z \in B$ such that $y \leq z \leq x$ and $z \in N(z)$.

The mapping N of elements of an interior algebra to their filters of neighbourhoods is a neighbourhood function on the underlying Boolean algebra of the interior algebra. Moreover, given a neighbourhood function N on a Boolean algebra with underlying set B , we can define an interior operator by $x^I = \max\{y \in B : x \in N(y)\}$ thereby obtaining an interior algebra. $N(x)$ will then be precisely the filter of neighbourhoods of x in this interior algebra. Thus interior algebras are equivalent to Boolean algebras with specified neighbourhood functions.

In terms of neighbourhood functions, the open elements are precisely those elements x such that $x \in N(x)$. In terms of open elements $x \in N(y)$ if and only if there is an open element z such that $y \leq z \leq x$.

Neighbourhood functions may be defined more generally on (meet)-semilattices producing the structures known as neighbourhood (semi)lattices. Interior algebras may thus be viewed as precisely the **Boolean neighbourhood lattices** i.e. those neighbourhood lattices whose underlying semilattice forms a Boolean algebra.

Modal logic

Given a theory (set of formal sentences) M in the modal logic **S4**, we can form its Lindenbaum-Tarski algebra:

$$L(M) = \llbracket M / \sim, \wedge, \vee, \neg, F, T, \Box \rrbracket$$

where \sim is the equivalence relation on sentences in M given by $p \sim q$ if and only if p and q are logically equivalent in M , and M / \sim is the set of equivalence classes under this relation. Then $L(M)$ is an interior algebra. The interior operator in this case corresponds to the modal operator \Box (**necessarily**), while the closure operator corresponds to \Diamond (**possibly**). This construction is a special case of a more general result for modal algebras and modal logic.

The open elements of $L(M)$ correspond to sentences that are only true if they are **necessarily** true, while the closed elements correspond to those that are only false if they are **necessarily** false.

Because of their relation to **S4**, interior algebras are sometimes called **S4 algebras** or **Lewis algebras**, after the logician C. I. Lewis, who first proposed the modal logics **S4** and **S5**.

Preorders

Since interior algebras are (normal) Boolean algebras with operators, they can be represented by fields of sets on appropriate relational structures. In particular, since they are modal algebras, they can be represented as fields of sets on a set with a single binary relation, called a modal frame. The modal frames corresponding to interior algebras are precisely the preordered sets. Preordered sets (also called *S4-frames*) provide the Kripke semantics of the modal logic **S4**, and the connection between interior algebras and preorders is deeply related to their connection with modal logic.

Given a preordered set $X = \llbracket X, \ll \rrbracket$ we can construct an interior algebra

$$B(X) = \llbracket P(X), \cap, \cup, ', \emptyset, X, I \rrbracket$$

from the power set Boolean algebra of X where the interior operator I is given by

$$S^I = \{x \in X : \text{for all } y \in X, x \ll y \text{ implies } y \in S\} \text{ for all } S \subseteq X.$$

The corresponding closure operator is given by

$$S^C = \{x \in X : \text{there exists a } y \in S \text{ with } x \ll y\} \text{ for all } S \subseteq X.$$

S^I is the set of all *worlds* inaccessible from *worlds* outside S , and S^C is the set of all *worlds* accessible from some *world* in S . Every interior algebra can be embedded in an interior algebra of the form $\mathbf{B}(X)$ for some preordered set X giving the above mentioned representation as a field of sets (a **preorder field**).

This construction and representation theorem is a special case of the more general result for modal algebras and modal frames. In this regard, interior algebras are particularly interesting because of their connection to topology. The construction provides the preordered set X with a topology, the Alexandrov topology, producing a topological space $\mathbf{T}(X)$ whose open sets are:

$$\{O \subseteq X : \text{for all } x \in O \text{ and all } y \in X, x \ll y \text{ implies } y \in O\}.$$

The corresponding closed sets are:

$$\{C \subseteq X : \text{for all } x \in C \text{ and all } y \in X, y \ll x \text{ implies } y \in C\}.$$

In other words, the open sets are the ones whose *worlds* are inaccessible from outside (the **up-sets**), and the closed sets are the ones for which every outside *world* is inaccessible from inside (the **down-sets**). Moreover, $\mathbf{B}(X) = \mathbf{A}(\mathbf{T}(X))$.

Monadic Boolean algebras

Any monadic Boolean algebra can be considered to be an interior algebra where the interior operator is the universal quantifier and the closure operator is the existential quantifier. The monadic Boolean algebras are then precisely the variety of interior algebras satisfying the identity $x^{IC} = x^I$. In other words they are precisely the interior algebras in which every open element is closed or equivalently, in which every closed element is open. Moreover, such interior algebras are precisely the semisimple interior algebras. They are also the interior algebras corresponding to the modal logic **S5**, and so have also been called **S5 algebras**.

In the relationship between preordered sets and interior algebras they correspond to the case where the preorder is an equivalence relation, reflecting the fact that such preordered sets provide the Kripke semantics for **S5**. This also reflects the relationship between the monadic logic of quantification (for which monadic Boolean algebras provide an algebraic description) and **S5** where the modal operators \Box (**necessarily**) and \Diamond (**possibly**) can be interpreted in the Kripke semantics using monadic universal and existential quantification, respectively, without reference to an accessibility relation.

Heyting algebras

The open elements of an interior algebra form a Heyting algebra and the closed elements form a dual Heyting algebra. The regular open elements and regular closed elements correspond to the pseudo-complemented elements and dual pseudo-complemented elements of these algebras respectively and thus form Boolean algebras. The clopen elements correspond to the complemented elements and form a common subalgebra of these Boolean algebras as well as of the interior algebra itself. Every Heyting algebra can be represented as the open elements of an interior algebra.

Heyting algebras play the same role for intuitionistic logic that interior algebras play for the modal logic **S4** and Boolean algebras play for propositional logic. The relation between Heyting algebras and interior algebras reflects the relationship between intuitionistic logic and **S4**, in which one can interpret theories of intuitionistic logic as **S4** theories closed under necessity.

Derivative algebras

Given an interior algebra A , the closure operator obeys the axioms of the derivative operator, D . Hence we can form a derivative algebra $D(A)$ with the same underlying Boolean algebra as A by using the closure operator as a derivative operator.

Thus interior algebras are derivative algebras. From this perspective, they are precisely the variety of derivative algebras satisfying the identity $x^D \geq x$. Derivative algebras provide the appropriate algebraic semantics for the modal logic **WK4**. Hence derivative algebras stand to topological derived sets and **WK4** as interior/closure algebras stand to topological interiors/closures and **S4**.

Given a derivative algebra V with derivative operator D , we can form an interior algebra $I(V)$ with the same underlying Boolean algebra as V , with interior and closure operators defined by $x^I = x \cdot x^D$ and $x^C = x + x^D$, respectively. Thus every derivative algebra can be regarded as an interior algebra. Moreover given an interior algebra A , we have $I(D(A)) = A$. However, $D(I(V)) = V$ does *not* necessarily hold for every derivative algebra V .

Metamathematics

Grzegorczyk proved the elementary theory of closure algebras undecidable.^[1]

Notes

[1] Andrzej Grzegorczyk (1951) "Undecidability of some topological theories," *Fundamenta Mathematicae* 38: 137-52.

References

- Blok, W.A., 1976, *Varieties of interior algebras*, Ph.D. thesis, University of Amsterdam.
- Esakia, L., 2004, "Intuitionistic logic and modality via topology," *Annals of Pure and Applied Logic* 127: 155-70.
- McKinsey, J.C.C. and Alfred Tarski, 1944, "The Algebra of Topology," *Annals of Mathematics* 45: 141-91.
- Naturman, C.A., 1991, *Interior Algebras and Topology*, Ph.D. thesis, University of Cape Town Department of Mathematics.

Lindenbaum–Tarski algebra

In mathematical logic, the **Lindenbaum–Tarski algebra** (or **Lindenbaum algebra**) of a logical theory T consists of the equivalence classes of sentences of the theory (i.e., the quotient, under the equivalence relation \sim defined such that $p \sim q$ exactly when p and q are provably equivalent in T). That is, two sentences are equivalent if the theory T proves that each implies the other. The Lindenbaum–Tarski algebra is thus the quotient algebra obtained by factoring the boolean algebra of formulas by this congruence relation.

The algebra is named for logicians Adolf Lindenbaum and Alfred Tarski. It was first introduced by Tarski in 1935 as a device to establish correspondence between classical propositional calculus and Boolean algebras. The Lindenbaum–Tarski algebra is considered the origin of the modern algebraic logic.^[1]

Operations

The operations in a Lindenbaum–Tarski algebra A are inherited from those in the underlying theory T . These typically include conjunction and disjunction, which are well-defined on the equivalence classes. When negation is also present in T , then A is a Boolean algebra, provided the logic is classical. If the theory is propositional and its set of logical connectives is functionally complete, the Lindenbaum–Tarski algebra is the free Boolean algebra generated by the set of propositional variables.

Related algebras

Heyting algebras and interior algebras are the Lindenbaum–Tarski algebras for intuitionistic logic and the modal logic **S4**, respectively.

A logic for which Tarski's method is applicable, is called *algebraizable*. There are however a number of logics where this is not the case, for instance the modal logics **S1**, **S2**, or **S3**, which lack the rule of necessitation ($\vdash\varphi$ implying $\vdash\Box\varphi$), so \sim (defined above) is not a congruence (because $\vdash\varphi\rightarrow\psi$ does not imply $\vdash\Box\varphi\rightarrow\Box\psi$). Another type of logics where Tarski's method is inapplicable are relevance logics, because given two theorems an implication from one to the other may not itself be a theorem in a relevance logic. The study of the algebraization process (and notion) as topic of interest by itself, not necessarily by Tarski's method, has led to the development of abstract algebraic logic.

References

[1] ; here: pages 1-2

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

Relation algebra

In mathematics and abstract algebra, a **relation algebra** is a residuated Boolean algebra expanded with an involution called **converse**, a unary operation. The motivating example of a relation algebra is the algebra 2^{X^2} of all binary relations on a set X , that is, subsets of the cartesian square X^2 , with $R \bullet S$ interpreted as the usual composition of binary relations R and S , and with the converse of R interpreted as the inverse relation.

Relation algebra emerged in the 19th century work of Augustus De Morgan and Charles Peirce, which culminated in the algebraic logic of Ernst Schröder. The equational form of relation algebra treated here was developed by Alfred Tarski and his students, starting in the 1940s. Tarski and Givant (1987) applied relation algebra to a variable-free treatment of axiomatic set theory, with the implication that mathematics founded on set theory could itself be conducted without variables.

Definition

A **relation algebra** $(L, \wedge, \vee, \neg, 0, 1, \bullet, \mathbf{I}, \circ)$ is an algebraic structure equipped with the Boolean operations of conjunction $x \wedge y$, disjunction $x \vee y$, and negation $\neg x$, the Boolean constants 0 and 1, the relational operations of composition $x \bullet y$ and converse x° , and the relational constant \mathbf{I} , such that these operations and constants satisfy certain equations constituting an axiomatization of relation algebras. A relation algebra is to a system of binary relations on a set containing the empty (0), complete (1), and identity (\mathbf{I}) relations and closed under these five operations as a group is to a system of permutations of a set containing the identity permutation and closed under composition and inverse.

Following Jónsson and Tsinakis (1993) it is convenient to define additional operations $x \triangleleft y = x \bullet y^\circ$, and, dually, $x \triangleright y = x^\circ \bullet y$. Jónsson and Tsinakis showed that $\mathbf{I} \triangleleft x = x \triangleright \mathbf{I}$, and that both were equal to x° . Hence a relation algebra can equally well be defined as an algebraic structure $(L, \wedge, \vee, \neg, 0, 1, \bullet, \mathbf{I}, \triangleleft, \triangleright)$. The advantage of this signature over the usual one that a relation algebra can then be defined in full simply as a residuated Boolean algebra for which $\mathbf{I} \triangleleft x$ is an involution, that is, $\mathbf{I} \triangleleft (\mathbf{I} \triangleleft x) = x$. The latter condition can be thought of as the relational counterpart of the equation $1/(1/x) = x$ for ordinary arithmetic reciprocal, and some authors use reciprocal as a synonym for converse.

Since residuated Boolean algebras are axiomatized with finitely many identities, so are relation algebras. Hence the latter form a variety, the variety **RA** of relation algebras. Expanding the above definition as equations yields the following finite axiomatization.

Axioms

The axioms **B1-B10** below are adapted from Givant (2006: 283), and were first set out by Tarski in 1948.^[1]

L is a Boolean algebra under binary disjunction, \vee , and unary complementation (\neg) :

$$\mathbf{B1}: A \vee B = B \vee A$$

$$\mathbf{B2}: A \vee (B \vee C) = (A \vee B) \vee C$$

$$\mathbf{B3}: (\neg A \vee B) \neg \vee (\neg A \vee \neg B) \neg = A$$

This axiomatization of Boolean algebra is due to Huntington (1933). Note that the meet of the implied Boolean algebra is *not* the \bullet operator (even though it distributes over \vee like a meet does), nor is the 1 of the Boolean algebra the \mathbf{I} constant.

L is a monoid under binary composition (\bullet) and nullary identity \mathbf{I} :

$$\mathbf{B4}: A \bullet (B \bullet C) = (A \bullet B) \bullet C$$

$$\mathbf{B5}: A \bullet \mathbf{I} = A$$

Unary converse $(\cdot)^\sim$ is an involution with respect to composition:

$$\mathbf{B6}: A^{\sim\sim} = A$$

$$\mathbf{B7}: (A \bullet B)^\sim = B^\sim \bullet A^\sim$$

Converse and composition distribute over disjunction:

$$\mathbf{B8}: (A \vee B)^\sim = A^\sim \vee B^\sim$$

$$\mathbf{B9}: (A \vee B) \bullet C = (A \bullet C) \vee (B \bullet C)$$

B10 is Tarski's equational form of the fact, discovered by Augustus De Morgan, that $A \bullet B \leq C^- \leftrightarrow A^\sim \bullet C \leq B^- \leftrightarrow C \bullet B^\sim \leq A^-$.

$$\mathbf{B10}: (A^\sim \bullet (A \bullet B)^-) \vee B^- = B^-$$

These axioms are ZFC theorems; for the purely Boolean **B1-B3**, this fact is trivial. After each of the following axioms is shown the number of the corresponding theorem in chpt. 3 of Suppes (1960), an exposition of ZFC: **B4** 27, **B5** 45, **B6** 14, **B7** 26, **B8** 16, **B9** 23.

Expressing properties of binary relations in RA

The following table shows how many of the usual properties of binary relations can be expressed as succinct **RA** equalities or inequalities. Below, an inequality of the form $A \leq B$ is shorthand for the Boolean equation $A \vee B = B$.

The most complete set of results of this nature is chpt. C of Carnap (1958), where the notation is rather distant from that of this entry. Chpt. 3.2 of Suppes (1960) contains fewer results, presented as ZFC theorems and using a notation that more resembles that of this entry. Neither Carnap nor Suppes formulated their results using the **RA** of this entry, or in an equational manner.

R is	If and only if:
Functional	$R^\sim \bullet R \leq \mathbf{I}$
Total or Connected	$\mathbf{I} \leq R \bullet R^\sim$ (R^\sim is surjective)
Function	functional and total.
Injective	$R \bullet R^\sim \leq \mathbf{I}$ (R^\sim is functional)
Surjective	$\mathbf{I} \leq R^\sim \bullet R$ (R^\sim is total)
Injection	$R^\sim \bullet R = R \bullet R^\sim = \mathbf{I}$ (Injective surjective function)
Reflexive	$\mathbf{I} \leq R$
Coreflexive	$\mathbf{R} \leq I$
Irreflexive	$R \wedge \mathbf{I} = 0$
Transitive	$R \bullet R \leq R$
Preorder	R is reflexive and transitive.
Antisymmetric	$R \wedge R^\sim \leq \mathbf{I}$
Partial order	R is an antisymmetric preorder.
Total order	R is a total partial order.
Strict partial order	R is transitive and irreflexive.
Strict total order	R is a total strict partial order.
Symmetric	$R^\sim = R$
Equivalence	$R \bullet R^\sim = R$. R is a symmetric preorder.

Asymmetric	$R \neq R^\sim$
Dense	$R \wedge \mathbf{I}^\sim \leq (R \wedge \mathbf{I}^\sim) \bullet (R \wedge \mathbf{I}^\sim)$.

Expressive power

The metamathematics of **RA** are discussed at length in Tarski and Givant (1987), and more briefly in Givant (2006).

RA consists entirely of equations manipulated using nothing more than uniform replacement and the substitution of equals for equals. Both rules are wholly familiar from school mathematics and from abstract algebra generally. Hence **RA** proofs are carried out in a manner familiar to all mathematicians, unlike the case in mathematical logic generally.

RA can express any (and up to logical equivalence, exactly the) first-order logic (FOL) formulas containing no more than three variables. (A given variable can be quantified multiple times and hence quantifiers can be nested arbitrarily deeply by "reusing" variables.) Surprisingly, this fragment of FOL suffices to express Peano arithmetic and almost all axiomatic set theories ever proposed. Hence **RA** is, in effect, a way of algebraizing nearly all mathematics, while dispensing with FOL and its connectives, quantifiers, turnstiles, and modus ponens. Because **RA** can express Peano arithmetic and set theory, Gödel's incompleteness theorems apply to it; **RA** is incomplete, incompletable, and undecidable.^[citation needed] (N.B. The Boolean algebra fragment of **RA** is complete and decidable.)

The **representable relation algebras**, forming the class **RRA**, are those relation algebras isomorphic to some relation algebra consisting of binary relations on some set, and closed under the intended interpretation of the **RA** operations. It is easily shown, e.g. using the method of pseudoelementary classes, that **RRA** is a quasivariety, that is, axiomatizable by a universal Horn theory. In 1950, Roger Lyndon proved the existence of equations holding in **RRA** that did not hold in **RA**. Hence the variety generated by **RRA** is a proper subvariety of the variety **RA**. In 1955, Alfred Tarski showed that **RRA** is itself a variety. In 1964, Donald Monk showed that **RRA** has no finite axiomatization, unlike **RA** which is finitely axiomatized by definition.

Q-Relation Algebras

An **RA** is a Q-Relation Algebra (**QRA**) if, in addition to **B1-B10**, there exist some A and B such that (Tarski and Givant 1987: §8.4):

$$\mathbf{Q0}: A^\sim \bullet A \leq \mathbf{I}$$

$$\mathbf{Q1}: B^\sim \bullet B \leq \mathbf{I}$$

$$\mathbf{Q2}: A^\sim \bullet B = \mathbf{1}$$

Essentially these axioms imply that the universe has a (non-surjective) pairing relation whose projections are A and B . It is a theorem that every **QRA** is a **RRA** (Proof by Maddux, see Tarski & Givant 1987: 8.4(iii)).

Every **QRA** is representable (Tarski and Givant 1987). That not every relation algebra is representable is a fundamental way **RA** differs from **QRA** and Boolean algebras which, by Stone's representation theorem for Boolean algebras, are always representable as sets of subsets of some set, closed under union, intersection, and complement.

Examples

1. Any Boolean algebra can be turned into a **RA** by interpreting conjunction as composition (the monoid multiplication \bullet), i.e. $x \bullet y$ is defined as $x \wedge y$. This interpretation requires that converse interpret identity ($\tilde{y} = y$), and that both residuals $y \setminus x$ and x/y interpret the conditional $y \rightarrow x$ (i.e., $\neg y \vee x$).
2. The motivating example of a relation algebra depends on the definition of a binary relation R on a set X as any subset $R \subseteq X^2$, where X^2 is the Cartesian square of X . The power set 2^{X^2} consisting of all binary relations on X is a Boolean algebra. While 2^{X^2} can be made a relation algebra by taking $R \bullet S = R \wedge S$, as per example (1) above, the standard interpretation of \bullet is instead $x(R \bullet S)z = \exists y.xRySz$. That is, the ordered pair (x,z) belongs to the relation $R \bullet S$ just when there exists $y \in X$ such that $(x,y) \in R$ and $(y,z) \in S$. This interpretation uniquely determines $R \setminus S$ as consisting of all pairs (y,z) such that for all $x \in X$, if xRy then xSz . Dually, S/R consists of all pairs (x,y) such that for all $z \in X$, if yRz then xSz . The translation $\tilde{y} = \neg(y \setminus \mathbf{I})$ then establishes the converse R^\sim of R as consisting of all pairs (y,x) such that $(x,y) \in R$.
3. An important generalization of the previous example is the power set 2^E where $E \subseteq X^2$ is any equivalence relation on the set X . This is a generalization because X^2 is itself an equivalence relation, namely the complete relation consisting of all pairs. While 2^E is not a subalgebra of 2^{X^2} when $E \neq X^2$ (since in that case it does not contain the relation X^2 , the top element 1 being E instead of X^2), it is nevertheless turned into a relation algebra using the same definitions of the operations. Its importance resides in the definition of a *representable relation algebra* as any relation algebra isomorphic to a subalgebra of the relation algebra 2^E for some equivalence relation E on some set. The previous section says more about the relevant metamathematics.
4. If group sum or product interprets composition, group inverse interprets converse, group identity interprets **I**, and if R is a one to one correspondence, so that $R^\sim \bullet R = R \bullet R^\sim = \mathbf{I}$,^[2] then L is a group as well as a monoid. **B4-B7** become well-known theorems of group theory, so that **RA** becomes a proper extension of group theory as well as of Boolean algebra.

Historical remarks

DeMorgan founded **RA** in 1860, but C. S. Peirce took it much further and became fascinated with its philosophical power. The work of DeMorgan and Peirce came to be known mainly in the extended and definitive form Ernst Schröder gave it in Vol. 3 of his *Vorlesungen* (1890–1905). *Principia Mathematica* drew strongly on Schröder's **RA**, but acknowledged him only as the inventor of the notation. In 1912, Alwin Korselt proved that a particular formula in which the quantifiers were nested 4 deep had no **RA** equivalent.^[3] This fact led to a loss of interest in **RA** until Tarski (1941) began writing about it. His students have continued to develop **RA** down to the present day. Tarski returned to **RA** in the 1970s with the help of Steven Givant; this collaboration resulted in the monograph by Tarski and Givant (1987), the definitive reference for this subject. For more on the history of **RA**, see Maddux (1991, 2006).

Software

- RelMICS / Relational Methods in Computer Science [4] maintained by Wolfram Kahl [5]
- Carsten Sinz: ARA / An Automatic Theorem Prover for Relation Algebras [6]

Footnotes

- [1] Alfred Tarski (1948) "Abstract: Representation Problems for Relation Algebras," *Bulletin of the AMS* 54: 80.
- [2] Tarski, A. (1941), p. 87.
- [3] Korselt did not publish his finding. It was first published in Leopold Löwenheim (1915) "Über Möglichkeiten im Relativkalkül," *Mathematische Annalen* 76: 447–470. Translated as "On possibilities in the calculus of relatives" in Jean van Heijenoort, 1967. *A Source Book in Mathematical Logic, 1879–1931*. Harvard Univ. Press: 228–251.
- [4] <http://relmics.mcmaster.ca/html/index.html>
- [5] <http://www.cas.mcmaster.ca/~kahl/>
- [6] <http://www-sr.informatik.uni-tuebingen.de/~sinz/ARA/>

References

- Rudolf Carnap (1958) *Introduction to Symbolic Logic and its Applications*. Dover Publications.
- Givant, Steven (2006). "The calculus of relations as a foundation for mathematics". *Journal of Automated Reasoning* 37: 277–322. doi: 10.1007/s10817-006-9062-x (<http://dx.doi.org/10.1007/s10817-006-9062-x>).
- Halmos, P. R., 1960. *Naive Set Theory*. Van Nostrand.
- Leon Henkin, Alfred Tarski, and Monk, J. D., 1971. *Cylindric Algebras, Part 1*, and 1985, *Part 2*. North Holland.
- Hirsch R., and Hodkinson, I., 2002, *Relation Algebra by Games* (http://www.elsevier.com/wps/find/bookdescription.cws_home/625473/description#description), vol. 147 in *Studies in Logic and the Foundations of Mathematics*. Elsevier Science.
- Jónsson, Bjarni; Tsinakis, Constantine (1993). "Relation algebras as residuated Boolean algebras". *Algebra Universalis* 30: 469–78. doi: 10.1007/BF01195378 (<http://dx.doi.org/10.1007/BF01195378>).
- Maddux, Roger (1991). "The Origin of Relation Algebras in the Development and Axiomatization of the Calculus of Relations" (<http://orion.math.iastate.edu/maddux/papers/Maddux1991.pdf>). *Studia Logica* 50 (3–4): 421–455. doi: 10.1007/BF00370681 (<http://dx.doi.org/10.1007/BF00370681>).
- -----, 2006. *Relation Algebras*, vol. 150 in *Studies in Logic and the Foundations of Mathematics*. Elsevier Science.
- Patrick Suppes, 1960. *Axiomatic Set Theory*. Van Nostrand. Dover reprint, 1972. Chpt. 3.
- Gunther Schmidt, 2010. *Relational Mathematics*. Cambridge University Press.
- Tarski, Alfred (1941). "On the calculus of relations" (<http://www.jstor.org/stable/2268577>). *Journal of Symbolic Logic* 6: 73–89.
- -----, and Givant, Steven, 1987. *A Formalization of Set Theory without Variables*. Providence RI: American Mathematical Society.

External links

- Yohji AKAMA, Yasuo Kawahara, and Hitoshi Furusawa, " Constructing Allegory from Relation Algebra and Representation Theorems. (<http://nicosia.is.s.u-tokyo.ac.jp/pub/staff/akama/repr.ps>)"
- Richard Bird, Oege de Moor, Paul Hoogendoijk, " Generic Programming with Relations and Functors. (<http://citeseer.ist.psu.edu/bird99generic.html>)"
- R.P. de Freitas and Viana, " A Completeness Result for Relation Algebra with Binders. (<http://www.cos.ufrj.br/~naborges/fv02.ps>)"
- Peter Jipsen (<http://www1.chapman.edu/~jipsen/>):
 - Relation algebras (http://math.chapman.edu/structuresold/files/Relation_algebras.pdf). In Mathematical structures. (<http://math.chapman.edu/cgi-bin/structures>) If there are problems with LaTeX, see an old HTML version here. (http://math.chapman.edu/cgi-bin/structures.pl?Relation_algebras)
 - " Foundations of Relations and Kleene Algebra. (<http://math.chapman.edu/~jipsen/talks/RelMiCS2006/JipsenRAKAtutorial.pdf>)"
 - " Computer Aided Investigations of Relation Algebras. (<http://www1.chapman.edu/~jipsen/dissertation/>)"
 - " A Gentzen System And Decidability For Residuated Lattices." (<http://citeseer.ist.psu.edu/337149.html>)
- Vaughan Pratt:
 - " Origins of the Calculus of Binary Relations. (<http://boole.stanford.edu/pub/ocbr.pdf>)" A historical treatment.
 - " The Second Calculus of Binary Relations. (<http://boole.stanford.edu/pub/scbr.pdf>)"
- Priss, Uta:
 - " An FCA interpretation of Relation Algebra. (<http://www.upriss.org.uk/papers/fcaic06.pdf>)"
 - " Relation Algebra and FCA (<http://www.upriss.org.uk/fca/relalg.html>)" Links to publications and software
- Kahl, Wolfram (<http://www.cas.mcmaster.ca/~kahl/>), and Schmidt, Gunther, (<http://ist.unibw-muenchen.de/People/schmidt/>) " Exploring (Finite) Relation Algebras Using Tools Written in Haskell. (<http://relmics.mcmaster.ca/~kahl/Publications/TR/2000-02/>)" See homepage (<http://relmics.mcmaster.ca/tools/RATH/index.html>) of the whole project.

Residuated Boolean algebra

In mathematics, a residuated Boolean algebra is a residuated lattice whose lattice structure is that of a Boolean algebra. Examples include Boolean algebras with the monoid taken to be conjunction, the set of all formal languages over a given alphabet Σ under concatenation, the set of all binary relations on a given set X under relational composition, and more generally the power set of any equivalence relation, again under relational composition. The original application was to relation algebras as a finitely axiomatized generalization of the binary relation example, but there exist interesting examples of residuated Boolean algebras that are not relation algebras, such as the language example.

Definition

A **residuated Boolean algebra** is an algebraic structure $(L, \wedge, \vee, \neg, 0, 1, \bullet, \mathbf{I}, \setminus, /)$ such that

- (i) $(L, \wedge, \vee, \bullet, \mathbf{I}, \setminus, /)$ is a residuated lattice, and
- (ii) $(L, \wedge, \vee, \neg, 0, 1)$ is a Boolean algebra.

An equivalent signature better suited to the relation algebra application is $(L, \wedge, \vee, \neg, 0, 1, \bullet, \mathbf{I}, \triangleright, \triangleleft)$ where the unary operations $x\setminus$ and $x\triangleright$ are intertranslatable in the manner of De Morgan's laws via

$$x\setminus y = \neg(x\triangleright \neg y), \quad x\triangleright y = \neg(x\setminus \neg y), \quad \text{and dually } /y \text{ and } \triangleleft y \text{ as}$$

$$x/y = \neg(\neg x\triangleleft y), \quad x\triangleleft y = \neg(\neg x/y),$$

with the residuation axioms in the residuated lattice article reorganized accordingly (replacing z by $\neg z$) to read

$$(x\triangleright z)\wedge y = 0 \iff (x\bullet y)\wedge z = 0 \iff (z\triangleleft y)\wedge x = 0$$

This De Morgan dual reformulation is motivated and discussed in more detail in the section below on conjugacy.

Since residuated lattices and Boolean algebras are each definable with finitely many equations, so are residuated Boolean algebras, whence they form a finitely axiomatizable variety.

Examples

1. Any Boolean algebra, with the monoid multiplication \bullet taken to be conjunction and both residuals taken to be material implication $x\rightarrow y$. Of the remaining 15 binary Boolean operations that might be considered in place of conjunction for the monoid multiplication, only five meet the monotonicity requirement, namely 0, 1, x , y , and $x\vee y$. Setting $y = z = 0$ in the residuation axiom $y \leq x\setminus z \iff x\bullet y \leq z$, we have $0 \leq x\setminus 0 \iff x\bullet 0 \leq 0$, which is falsified by taking $x = 1$ when $x\bullet y = 1$, x , or $x\vee y$. The dual argument for z/y rules out $x\bullet y = y$. This just leaves $x\bullet y = 0$ (a constant binary operation independent of x and y), which satisfies almost all the axioms when the residuals are both taken to be the constant operation $x/y = x\setminus y = 1$. The axiom it fails is $x\bullet \mathbf{I} = x = \mathbf{I}\bullet x$, for want of a suitable value for \mathbf{I} . Hence conjunction is the only binary Boolean operation making the monoid multiplication that of a residuated Boolean algebra.
2. The power set 2^{X^2} made a Boolean algebra as usual with \cap , \cup and complement relative to X^2 , and made a monoid with relational composition. The monoid unit \mathbf{I} is the identity relation $\{(x,x)|x \in X\}$. The right residual $R\setminus S$ is defined by $x(R\setminus S)y$ if and only if for all z in X , zRx implies zSy . Dually the left residual S/R is defined by $y(S/R)x$ if and only if for all z in X , xRz implies ySz .
3. The power set 2^{Σ^*} made a Boolean algebra as for example 2, but with language concatenation for the monoid. Here the set Σ is used as an alphabet while Σ^* denotes the set of all finite (including empty) words over that alphabet. The concatenation LM of languages L and M consists of all words uv such that $u \in L$ and $v \in M$. The monoid unit is the language $\{\varepsilon\}$ consisting of just the empty word ε . The right residual $M\setminus L$ consists of all words w over Σ such that $Mw \subseteq L$. The left residual L/M is the same with wM in place of Mw .

Conjugacy

The De Morgan duals \triangleright and \triangleleft of residuation arise as follows. Among residuated lattices, Boolean algebras are special by virtue of having a complementation operation \neg . This permits an alternative expression of the three inequalities

$$y \leq x \setminus z \Leftrightarrow x \cdot y \leq z \Leftrightarrow x \leq z \setminus y$$

in the axiomatization of the two residuals in terms of disjointness, via the equivalence $x \leq y \Leftrightarrow x \wedge \neg y = 0$. Abbreviating $x \wedge y = 0$ to $x \# y$ as the expression of their disjointness, and substituting $\neg z$ for z in the axioms, they become with a little Boolean manipulation

$$\neg(x \setminus \neg z) \# y \Leftrightarrow x \cdot y \# z \Leftrightarrow \neg(\neg z / y) \# x$$

Now $\neg(x \setminus \neg z)$ is reminiscent of De Morgan duality, suggesting that $x \setminus$ be thought of as a unary operation f , defined by $f(y) = x \setminus y$, that has a De Morgan dual $\neg f(\neg y)$, analogous to $\forall x \varphi(x) = \neg \exists x \neg \varphi(x)$. Denoting this dual operation as $x \triangleright$, we define $x \triangleright z$ as $\neg(x \setminus \neg z)$. Similarly we define another operation $z \triangleleft y$ as $\neg(\neg z / y)$. By analogy with $x \setminus$ as the residual operation associated with the operation $x \cdot$, we refer to $x \triangleright$ as the conjugate operation, or simply **conjugate**, of $x \cdot$. Likewise $\triangleleft y$ is the **conjugate** of $\cdot y$. Unlike residuals, conjugacy is an equivalence relation between operations: if f is the conjugate of g then g is also the conjugate of f , i.e. the conjugate of the conjugate of f is f . Another advantage of conjugacy is that it becomes unnecessary to speak of right and left conjugates, that distinction now being inherited from the difference between $x \cdot$ and $\cdot x$, which have as their respective conjugates $x \triangleright$ and $\triangleleft x$. (But this advantage accrues also to residuals when $x \setminus$ is taken to be the residual operation to $x \cdot$.)

All this yields (along with the Boolean algebra and monoid axioms) the following equivalent axiomatization of a residuated Boolean algebra.

$$y \# x \triangleright z \Leftrightarrow x \cdot y \# z \Leftrightarrow x \# z \triangleleft y$$

With this signature it remains the case that this axiomatization can be expressed as finitely many equations.

Converse

In examples 2 and 3 it can be shown that $x \triangleright \mathbf{I} = \mathbf{I} \triangleleft x$. In example 2 both sides equal the converse x^\sim of x , while in example 3 both sides are \mathbf{I} when x contains the empty word and 0 otherwise. In the former case $x^{\sim\sim} = x$. This is impossible for the latter because $x \triangleright \mathbf{I}$ retains hardly any information about x . Hence in example 2 we can substitute x^\sim for x in $x \triangleright \mathbf{I} = x^\sim = \mathbf{I} \triangleleft x$ and cancel (soundly) to give

$$x^\sim \triangleright \mathbf{I} = x = \mathbf{I} \triangleleft x^\sim.$$

$x^{\sim\sim} = x$ can be proved from these two equations. Tarski's notion of a **relation algebra** can be defined as a residuated Boolean algebra having an operation x^\sim satisfying these two equations.

The cancellation step in the above is not possible for example 3, which therefore is not a relation algebra, x^\sim being uniquely determined as $x \triangleright \mathbf{I}$.

Consequences of this axiomatization of converse include $x^{\sim\sim} = x$, $\neg(x^\sim) = (\neg x)^\sim$, $(x \vee y)^\sim = x^\sim \vee y^\sim$, and $(x \cdot y)^\sim = y^\sim \cdot x^\sim$.

References

- Bjarni Jónsson and Constantine Tsinakis, Relation algebras as residuated Boolean algebras, *Algebra Universalis*, 30 (1993) 469-478.
- Peter Jipsen, *Computer aided investigations of relation algebras*^[1], Ph.D. Thesis, Vanderbilt University, May 1992.

References

[1] <http://www1.chapman.edu/~jipsen/dissertation/>

Robbins algebra

In abstract algebra, a **Robbins algebra** is an algebra containing a single binary operation, usually denoted by \vee , and a single unary operation usually denoted by \neg . These operations satisfy the following axioms:

For all elements a, b , and c :

1. Associativity: $a \vee (b \vee c) = (a \vee b) \vee c$
2. Commutativity: $a \vee b = b \vee a$
3. *Robbins equation*: $\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a$

For many years, it was conjectured, but unproven, that all Robbins algebras are Boolean algebras. This was proved in 1996, so the term "Robbins algebra" is now simply a synonym for "Boolean algebra".

History

In 1933, Edward Huntington proposed a new set of axioms for Boolean algebras, consisting of (1) and (2) above, plus:

- *Huntington's equation*: $\neg(\neg a \vee b) \vee \neg(\neg a \vee \neg b) = a$.

From these axioms, Huntington derived the usual axioms of Boolean algebra.

Very soon thereafter, Herbert Robbins posed the "Robbins conjecture", namely that the Huntington equation could be replaced with what came to be called the Robbins equation, and the result would still be Boolean algebra. \vee would interpret Boolean join and \neg Boolean complement. Boolean meet and the constants 0 and 1 are easily defined from the Robbins algebra primitives. Pending verification of the conjecture, the system of Robbins was called "Robbins algebra."

Verifying the Robbins conjecture required proving Huntington's equation, or some other axiomatization of a Boolean algebra, as theorems of a Robbins algebra. Huntington, Robbins, Alfred Tarski, and others worked on the problem, but failed to find a proof or counterexample.

William McCune proved the conjecture in 1996, using the automated theorem prover EQP. For a complete proof of the Robbins conjecture in one consistent notation and following McCune closely, see Mann (2003). Dahn (1998) simplified McCune's machine proof.

References

- Dahn, B. I. (1998) Abstract to "Robbins Algebras Are Boolean: A Revision of McCune's Computer-Generated Solution of Robbins Problem,"^[1] *Journal of Algebra* 208(2): 526–32.
- Mann, Allen (2003) "A Complete Proof of the Robbins Conjecture."^[2]
- William McCune, "Robbins Algebras Are Boolean,"^[3] With links to proofs and other papers.

References

- [1] http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6WH2-45S4RJV-7&_user=10&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000050221&_version=1&_urlVersion=0&_userid=10&md5=cafdb746bc7e14ffe040ff04ec4e42c2
- [2] http://math.colgate.edu/~amann/MA/robbins_complete.pdf
- [3] <http://www.cs.unm.edu/~mccune/papers/robbins/>

Sigma-algebra

In mathematical analysis, a **σ -algebra** (also **sigma-algebra**, **σ -field**, **sigma-field**) on a set is a collection of subsets satisfying certain properties. The main use of σ -algebras is in the definition of measures; specifically, the collection of those subsets for which a given measure is defined is a σ -algebra. This concept is important in mathematical analysis as the foundation for Lebesgue integration, and in probability theory, where it is interpreted as the collection of events which can be assigned probabilities.

The definition is that a σ -algebra over a set X is a nonempty collection Σ of subsets of X that is closed under the complement and countable unions of its members and contains X itself.

That is, a σ -algebra is an algebra of sets, completed to include countably infinite operations. The pair (X, Σ) is also a field of sets, called a **measurable space**.

If $X = \{a, b, c, d\}$, one possible sigma algebra on X is $\Sigma = \{\emptyset, \{a, b\}, \{c, d\}, \{a, b, c, d\}\}$, where \emptyset is the empty set. A more useful example is the set of subsets of the real line formed by starting with all open intervals and adding in all countable unions, countable intersections, and relative complements and continuing this process until the relevant closure properties are achieved (a construction known as the Borel σ -algebra [see Borel set]).

Motivation

A measure on X is a function which assigns a non-negative real number to subsets of X ; this can be thought of as making precise a notion of "size" or "volume" for sets. We want the size of the union of disjoint sets to be the sum of their individual sizes, even for an infinite sequence of disjoint sets.

One would like to assign a size to *every* subset of X , but in many natural settings, this is not possible. For example the axiom of choice implies that when the size under consideration is the ordinary notion of length for subsets of the real line, then there exist sets for which no size exists, for example, the Vitali sets. For this reason, one considers instead a smaller collection of privileged subsets of X . These subsets will be called the measurable sets. They are closed under operations that one would expect for measurable sets, that is, the complement of a measurable set is a measurable set and the countable union of measurable sets is a measurable set. Non-empty collections of sets with these properties are called σ -algebras.

The collection of subsets of X that form the σ -algebra is usually denoted by Σ , the capital Greek letter sigma. The pair (X, Σ) is an algebra of sets and also a field of sets, called a measurable space. If the subsets of X in Σ correspond to numbers in elementary algebra, then the two set operations union (symbol \cup) and intersection (\cap) correspond to addition and multiplication. The collection of sets Σ is completed to include countably infinite operations.

Definition and properties

Let X be some set, and let 2^X represent its power set. Then a subset $\Sigma \subset 2^X$ is called a **σ -algebra** if it satisfies the following three properties:

1. Σ is *non-empty*: There is at least one $A \subset X$ in Σ .
2. Σ is *closed under complementation*: If A is in Σ , then so is its complement, $X \setminus A$.
3. Σ is *closed under countable unions*: If A_1, A_2, A_3, \dots are in Σ , then so is $A = A_1 \cup A_2 \cup A_3 \cup \dots$.

From these axioms, it follows that the σ -algebra is also closed under countable intersections (by applying De Morgan's laws).

It also follows that the set X itself and the empty set are both in Σ , since by (1) Σ is non-empty, so some particular $A \in \Sigma$ may be chosen, and by (2), $X \setminus A$ is also in Σ . By (3) $A \cup (X \setminus A) = X$ is in Σ . And finally, since X is in Σ , (2) asserts that its complement, the empty set, is also in Σ .

Elements of the σ -algebra are called measurable sets. An ordered pair (X, Σ) , where X is a set and Σ is a σ -algebra over X , is called a **measurable space**. A function between two measurable spaces is called a measurable function if the preimage of every measurable set is measurable. The collection of measurable spaces forms a category, with the measurable functions as morphisms. Measures are defined as certain types of functions from a σ -algebra to $[0, \infty]$.

Relation to σ -ring

A σ -algebra Σ is just a σ -ring that contains the universal set X . A σ -ring need not be a σ -algebra, as for example measurable subsets of zero Lebesgue measure in the real line are a σ -ring, but not a σ -algebra since the real line has infinite measure and thus cannot be obtained by their countable union. If, instead of zero measure, one takes measurable subsets of finite Lebesgue measure, those are a ring but not a σ -ring, since the real line can be obtained by their countable union yet its measure is not finite.

Typographic note

σ -algebras are sometimes denoted using calligraphic capital letters, or the Fraktur typeface. Thus (X, Σ) may be denoted as (X, \mathcal{F}) or (X, \mathfrak{F}) . This is handy to avoid situations where the letter Σ may be confused for the summation operator.

σ -algebra generated by a family of sets

Let F be an arbitrary family of subsets of X . Then there exists a unique smallest σ -algebra which contains every set in F (even though F may or may not itself be a σ -algebra). This σ -algebra is denoted $\sigma(F)$ and called **the σ -algebra generated by F** .

To see that such a σ -algebra always exists, let

$$\Phi = \{ E \subset 2^X : E \text{ is a } \sigma\text{-algebra which contains } F \}.$$

The σ -algebra generated by F will therefore be the smallest element in Φ . Indeed, such a smallest element exists: First, Φ is not empty because the power set 2^X is in Φ . Consequently, let σ^* denote the intersection of all elements in Φ . This intersection will be nonempty because the arbitrary intersection of σ -algebras is a σ -algebra. Because each element in Φ contains F , the intersection σ^* will also contain F . Moreover, because each element in Φ is a σ -algebra, the intersection σ^* will also be a σ algebra (observe that if every element in Φ has the three properties of a σ -algebra, then the intersection of Φ will as well). Hence, because σ^* is a σ -algebra which contains F , σ^* is in Φ , and because it is the intersection of all sets in Φ , σ^* is indeed the *smallest* set in Φ by definition, which in turn implies that $\sigma^* = \sigma(F)$, the σ -algebra generated by F .

For a simple example, consider the set $X = \{1, 2, 3\}$. Then the σ -algebra generated by the single subset $\{1\}$ is $\sigma(\{1\}) = \{\emptyset, \{1\}, \{2, 3\}, \{1, 2, 3\}\}$. By an abuse of notation, when a collection of subsets contains only one member, A , one

may write $\sigma(A)$ instead of $\sigma(\{A\})$; in the prior example $\sigma(1)$ instead of $\sigma(\{1\})$.

σ -algebra generated by a function

If f is a function from a set X to a set Y and B is a σ -algebra of subsets of Y , then the **σ -algebra generated by the function f** , denoted by $\sigma(f)$, is the collection of all inverse images $f^{-1}(S)$ of the sets S in B .

Clearly, a function f from a set X to a set Y is measurable with respect to a σ -algebra Σ of subsets of X if and only if $\sigma(f)$ is a subset of Σ .

One common situation, and understood by default if B is not specified explicitly, is when Y is a metric or topological space and B are the Borel sets on Y .

Examples

Let X be any set, then the following are σ -algebras over X :

- The family consisting only of the empty set and the set X , called the minimal or **trivial σ -algebra** over X .
- The power set of X , called the **discrete σ -algebra**.
- The collection of subsets of X which are countable or whose complements are countable (which is distinct from the power set of X if and only if X is uncountable). This is the σ -algebra generated by the singletons of X .
- If $\{\Sigma_\lambda\}$ is a family of σ -algebras over X indexed by λ then the intersection of all Σ_λ 's is a σ -algebra over X .

Examples for generated algebras

An important example is the Borel algebra over any topological space: the σ -algebra generated by the open sets (or, equivalently, by the closed sets). Note that this σ -algebra is not, in general, the whole power set. For a non-trivial example, see the Vitali set.

On the Euclidean space \mathbf{R}^n , another σ -algebra is of importance: that of all Lebesgue measurable sets. This σ -algebra contains more sets than the Borel σ -algebra on \mathbf{R}^n and is preferred in integration theory, as it gives a complete measure space.

References

External links

- Hazewinkel, Michiel, ed. (2001), "Algebra of sets" (<http://www.encyclopediaofmath.org/index.php?title=p/a011400>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Sigma Algebra from PlanetMath.

Topological Boolean algebra

- In abstract algebra and mathematical logic, **topological Boolean algebra** is one of the many names that have been used for an interior algebra in the literature.
- In **topological algebra** — the study of topological spaces with algebraic structure, a **topological Boolean algebra** is a Boolean algebra endowed with a topological structure in which the operations of complement, join, and meet are continuous functions.

DISAMBIG

Two-element Boolean algebra

In mathematics and abstract algebra, the **two-element Boolean algebra** is the Boolean algebra whose *underlying set* (or universe or *carrier*) B is the Boolean domain. The elements of the Boolean domain are 1 and 0 by convention, so that $B = \{0, 1\}$. Paul Halmos's name for this algebra "2" has some following in the literature, and will be employed here.

Definition

B is a partially ordered set and the elements of B are also its bounds.

An operation of arity n is a mapping from B^n to B . Boolean algebra consists of two binary operations and unary complementation. The binary operations have been named and notated in various ways. Here they are called 'sum' and 'product', and notated by infix '+' and '.', respectively. Sum and product commute and associate, as in the usual algebra of real numbers. As for the order of operations, brackets are decisive if present. Otherwise '.' precedes '+'. Hence $A \cdot B + C$ is parsed as $(A \cdot B) + C$ and not as $A \cdot (B + C)$. Complementation is denoted by writing an overbar over its argument. The numerical analog of the complement of X is $1 - X$. In the language of universal algebra, a Boolean algebra is a $\langle B, +, ., \bar{}, 1, 0 \rangle$ algebra of type $\langle 2, 2, 1, 0, 0 \rangle$.

Either one-to-one correspondence between $\{0,1\}$ and $\{\text{True}, \text{False}\}$ yields classical bivalent logic in equational form, with complementation read as NOT. If 1 is read as *True*, '+' is read as OR, and '.' as AND, and vice versa if 1 is read as *False*.

Some basic identities

2 can be seen as grounded in the following trivial "Boolean" arithmetic:

- $1+1 = 1+0 = 0+1 = 1$
- $0 \cdot 0 = 0 \cdot 1 = 1 \cdot 0 = 0$
- $1 \cdot 1 = 1$
- $0+0 = 0$
- $\bar{1} = 0$
- $\bar{0} = 1$.

Note that:

- '+' and '.' work exactly as in numerical arithmetic, except that $1+1=1$. '+' and '.' are derived by analogy from numerical arithmetic; simply set any nonzero number to 1.
- Swapping 0 and 1, and '+' and '.' preserves truth; this is the essence of the duality pervading all Boolean algebras.

This Boolean arithmetic suffices to verify any equation of 2, including the axioms, by examining every possible assignment of 0s and 1s to each variable (see decision procedure).

The following equations may now be verified:

- $A + A = A$
- $A \cdot A = A$
- $A + 0 = A$
- $A + 1 = 1$
- $A \cdot 0 = 0$
- $\overline{\overline{A}} = A$.

Each of '+' and '.' distributes over the other:

- $A \cdot (B + C) = A \cdot B + A \cdot C$;
- $A + (B \cdot C) = (A + B) \cdot (A + C)$.

That '.' distributes over '+' agrees with elementary algebra, but not '+' over '.'. For this and other reasons, a sum of products (leading to a NAND synthesis) is more commonly employed than a product of sums (leading to a NOR synthesis).

Each of '+' and '.' can be defined in terms of the other and complementation:

- $A \cdot B = \overline{\overline{A} + \overline{B}}$
- $A + B = \overline{\overline{A} \cdot \overline{B}}$.

We only need one binary operation, and concatenation suffices to denote it. Hence concatenation and overbar suffice to notate **2**. This notation is also that of Quine's Boolean term schemata. Letting (X) denote the complement of X and " $()$ " denote either 0 or 1 yields the syntax of the primary algebra.

A *basis* for **2** is a set of equations, called axioms, from which all of the above equations (and more) can be derived. There are many known bases for all Boolean algebras and hence for **2**. An elegant basis notated using only concatenation and overbar is:

1. $ABC = BCA$ (Concatenation commutes, associates)
2. $\overline{AA} = 1$ (**2** is a complemented lattice, with an upper bound of 1)
3. $A0 = A$ (0 is the lower bound).
4. $\overline{A}\overline{AB} = A\overline{B}$ (**2** is a distributive lattice)

If $0=1$, (1)-(3) are the axioms for an abelian group.

(1) only serves to prove that concatenation commutes and associates. First assume that (1) associates from either the left or the right, then prove commutativity. Then prove association from the other direction. Associativity is simply association from the left and right combined.

This basis makes for an easy approach to proof, called calculation, that proceeds by simplifying expressions to 0 or 1, by invoking axioms (2)–(4), and the elementary identities $AA = A$, $\overline{\overline{A}} = A$, $1 + A = 1$, and the distributive law.

Metatheory

De Morgan's theorem states that if one does the following, in the given order, to any Boolean function:

- Complement every variable;
- Swap + and . operators (taking care to add brackets to ensure the order of operations remains the same);
- Complement the result,

the result is logically equivalent to what you started with. Repeated application of De Morgan's theorem to parts of a function can be used to drive all complements down to the individual variables.

A powerful and nontrivial metatheorem states that any theorem of **2** holds for all Boolean algebras.^[1] Conversely, an identity that holds for an arbitrary nontrivial Boolean algebra also holds in **2**. Hence all the mathematical content of Boolean algebra is captured by **2**. This theorem is useful because any equation in **2** can be verified by a decision

procedure. Logicians refer to this fact as "**2** is decidable". All known decision procedures require a number of steps that is an exponential function of the number of variables N appearing in the equation to be verified. Whether there exists a decision procedure whose steps are a polynomial function of N falls under the P = NP conjecture.

Footnotes

- [1] Givant, S., and Halmos, P. (2009) *Introduction to Boolean Algebras* (<http://dx.doi.org/10.1007/978-0-387-68436-9>), Springer Verlag.
Theorem 9.

References

Many elementary texts on Boolean algebra were published in the early years of the computer era. Perhaps the best of the lot, and one still in print, is:

- Mendelson, Elliot, 1970. *Schaum's Outline of Boolean Algebra*. McGraw-Hill.

The following items reveal how the two-element Boolean algebra is mathematically nontrivial.

- Stanford Encyclopedia of Philosophy: " The Mathematics of Boolean Algebra, (<http://plato.stanford.edu/entries/boolalg-math/>)" by J. Donald Monk.
- Burris, Stanley N., and H.P. Sankappanavar, H. P., 1981. *A Course in Universal Algebra*. (<http://www.thoralf.uwaterloo.ca/htdocs/ualg.html>) Springer-Verlag. ISBN 3-540-90578-2.

Extensions and generalizations

Complete Boolean algebra

In mathematics, a **complete Boolean algebra** is a Boolean algebra in which every subset has a supremum (least upper bound). Complete Boolean algebras are used to construct Boolean-valued models of set theory in the theory of forcing. Every Boolean algebra A has an essentially unique completion, which is a complete Boolean algebra containing A such that every element is the supremum of some subset of A . As a partially ordered set, this completion of A is the Dedekind-MacNeille completion.

More generally, if κ is a cardinal then a Boolean algebra is called **κ -complete** if every subset of cardinality less than κ has a supremum.

Examples

- Every finite Boolean algebra is complete.
- The algebra of subsets of a given set is a complete Boolean algebra.
- The regular open sets of any topological space form a complete Boolean algebra. This example is of particular importance because every forcing poset can be considered as a topological space (a base for the topology consisting of sets that are the set of all elements less than or equal to a given element). The corresponding regular open algebra can be used to form Boolean-valued models which are then equivalent to generic extensions by the given forcing poset.
- The algebra of all measurable subsets of a σ -finite measure space, modulo null sets, is a complete Boolean algebra.
- The algebra of all measurable subsets of a measure space is a \aleph_1 -complete Boolean algebra, but is not usually complete.
- The algebra of all subsets of an infinite set that are finite or have finite complement is a Boolean algebra but is not complete.
- The Boolean algebra of all Baire sets modulo meager sets in a topological space with a countable base is complete.
- Another example of a Boolean algebra that is not complete is the Boolean algebra $P(\omega)$ of all sets of natural numbers, quotiented out by the ideal Fin of finite subsets. The resulting object, denoted $P(\omega)/Fin$, consists of all equivalence classes of sets of naturals, where the relevant equivalence relation is that two sets of naturals are equivalent if their symmetric difference is finite. The Boolean operations are defined analogously, for example, if A and B are two equivalence classes in $P(\omega)/Fin$, we define $A \wedge B$ to be the equivalence class of $a \cap b$, where a and b are some (any) elements of A and B respectively.

Now let a_0, a_1, \dots be pairwise disjoint infinite sets of naturals, and let A_0, A_1, \dots be their corresponding equivalence classes in $P(\omega)/Fin$. Then given any upper bound X of A_0, A_1, \dots in $P(\omega)/Fin$, we can find a *lesser* upper bound, by removing from a representative for X one element of each a_n . Therefore the A_n have no supremum.

- A Boolean algebra is complete if and only if its Stone space of prime ideals is extremally disconnected.

Properties of complete Boolean algebras

- Sikorski's extension theorem states that

if A is a subalgebra of a Boolean algebra B , then any homomorphism from A to a complete Boolean algebra C can be extended to a morphism from B to C .

- Every subset of a complete Boolean algebra has a supremum, by definition; it follows that every subset also has an infimum (greatest lower bound).
- For a complete boolean algebra both infinite distributive laws hold.
- For a complete boolean algebra infinite de-Morgan's laws hold.

The completion of a Boolean algebra

The completion of a Boolean algebra can be defined in several equivalent ways:

- The completion of A is (up to isomorphism) the unique complete Boolean algebra B containing A such that A is dense in B ; this means that for every nonzero element of B there is a smaller non-zero element of A .
- The completion of A is (up to isomorphism) the unique complete Boolean algebra B containing A such that every element of B is the supremum of some subset of A .

The completion of a Boolean algebra A can be constructed in several ways:

- The completion is the Boolean algebra of regular open sets in the Stone space of prime ideals of A . Each element x of A corresponds to the open set of prime ideals not containing x (which open and closed, and therefore regular).
- The completion is the Boolean algebra of regular cuts of A . Here a *cut* is a subset U of A^+ (the non-zero elements of A) such that if q is in U and $p \leq q$ then p is in U , and is called *regular* if whenever p is not in U there is some $r \leq p$ such that U has no elements $\leq r$. Each element p of A corresponds to the cut of elements $\leq p$.

If A is a metric space and B its completion then any isometry from A to a complete metric space C can be extended to a unique isometry from B to C . The analogous statement for complete Boolean algebras is not true: a homomorphism from a Boolean algebra A to a complete Boolean algebra C cannot necessarily be extended to a (supremum preserving) homomorphism of complete Boolean algebras from the completion B of A to C . (By Sikorski's extension theorem it can be extended to a homomorphism of Boolean algebras from B to C , but this will not in general be a homomorphism of complete Boolean algebras; in other words, it need not preserve suprema.)

Free κ -complete Boolean algebras

Unless the Axiom of Choice is relaxed, free complete boolean algebras generated by a set do not exist (unless the set is finite). More precisely, for any cardinal κ , there is a complete Boolean algebra of cardinality 2^κ greater than κ that is generated as a complete Boolean algebra by a countable subset; for example the Boolean algebra of regular open sets in the product space κ^ω , where κ has the discrete topology. A countable generating set consists of all sets $a_{m,n}$ for m, n integers, consisting of the elements $x \in \kappa^\omega$ such that $x(m) < x(n)$. (This boolean algebra is called a collapsing algebra, because forcing with it collapses the cardinal κ onto ω .)

In particular the forgetful functor from complete Boolean algebras to sets has no left adjoint, even though it is continuous and the category of Boolean algebras is small-complete. This shows that the "solution set condition" in Freyd's adjoint functor theorem is necessary.

Given a set X , one can form the free Boolean algebra A generated by this set and then take its completion B . However B is not a "free" complete Boolean algebra generated by X (unless X is finite or AC is omitted), because a function from X to a free Boolean algebra C cannot in general be extended to a (supremum-preserving) morphism of Boolean algebras from B to C .

On the other hand, for any fixed cardinal κ , there is a free (or universal) κ -complete Boolean algebra generated by any given set.

References

- Johnstone, Peter T. (1982), *Stone spaces*, Cambridge University Press, ISBN 0-521-33779-8
- Koppelberg, Sabine (1989), Monk, J. Donald; Bonnet, Robert, eds., *Handbook of Boolean algebras 1*, Amsterdam: North-Holland Publishing Co., pp. xx+312, ISBN 0-444-70261-X, MR 0991565 (<http://www.ams.org/mathscinet-getitem?mr=0991565>)
- Monk, J. Donald; Bonnet, Robert, eds. (1989), *Handbook of Boolean algebras 2*, Amsterdam: North-Holland Publishing Co., ISBN 0-444-87152-7, MR 0991595 (<http://www.ams.org/mathscinet-getitem?mr=0991595>)
- Monk, J. Donald; Bonnet, Robert, eds. (1989), *Handbook of Boolean algebras 3*, Amsterdam: North-Holland Publishing Co., ISBN 0-444-87153-5, MR 0991607 (<http://www.ams.org/mathscinet-getitem?mr=0991607>)
- Vladimirov, D.A. (2001), "Boolean algebra" (<http://www.encyclopediaofmath.org/index.php?title=b/b016920>), in Hazewinkel, Michiel, *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4

Derivative algebra (abstract algebra)

In abstract algebra, a **derivative algebra** is an algebraic structure of the signature

$$\langle A, \cdot, +, ', 0, 1, {}^D \rangle$$

where

$$\langle A, \cdot, +, ', 0, 1 \rangle$$

is a Boolean algebra and D is a unary operator, the **derivative operator**, satisfying the identities:

1. $0^D = 0$
2. $x^{DD} \leq x + x^D$
3. $(x + y)^D = x^D + y^D$.

x^D is called the **derivative** of x . Derivative algebras provide an algebraic abstraction of the **derived set** operator in topology. They also play the same role for the modal logic $wK4 = K + p \wedge \Box p \rightarrow \Box \Box p$ that Boolean algebras play for ordinary propositional logic.

References

- Esakia, L., *Intuitionistic logic and modality via topology*, Annals of Pure and Applied Logic, 127 (2004) 155-170
- McKinsey, J.C.C. and Tarski, A., *The Algebra of Topology*, Annals of Mathematics, 45 (1944) 141-191

First-order logic

First-order logic is a formal system used in mathematics, philosophy, linguistics, and computer science. It is also known as **first-order predicate calculus**, the **lower predicate calculus**, **quantification theory**, and predicate logic. First-order logic is distinguished from propositional logic by its use of quantified variables.

A theory about some topic is usually first-order logic together with a specified domain of discourse over which the quantified variables range, finitely many functions which map from that domain into it, finitely many predicates defined on that domain, and a recursive set of axioms which are believed to hold for those things. Sometimes "theory" is understood in a more formal sense, which is just a set of sentences in first-order logic.

The adjective "first-order" distinguishes first-order logic from higher-order logic in which there are predicates having predicates or functions as arguments, or in which one or both of predicate quantifiers or function quantifiers are permitted. In first-order theories, predicates are often associated with sets. In interpreted higher-order theories, predicates may be interpreted as sets of sets.

There are many deductive systems for first-order logic that are sound (all provable statements are true) and complete (all true statements are provable). Although the logical consequence relation is only semidecidable, much progress has been made in automated theorem proving in first-order logic. First-order logic also satisfies several metalogical theorems that make it amenable to analysis in proof theory, such as the Löwenheim–Skolem theorem and the compactness theorem.

First-order logic is the standard for the formalization of mathematics into axioms and is studied in the foundations of mathematics. Mathematical theories, such as number theory and set theory, have been formalized into first-order axiom schemata such as Peano arithmetic and Zermelo–Fraenkel set theory (ZF) respectively.

No first-order theory, however, has the strength to describe fully and categorically structures with an infinite domain, such as the natural numbers or the real line. Categorical axiom systems for these structures can be obtained in stronger logics such as second-order logic.

For a history of first-order logic and how it came to dominate formal logic, see José Ferreirós (2001).

Introduction

While propositional logic deals with simple declarative propositions, first-order logic additionally covers predicates and quantification.

A predicate takes an entity or entities in the domain of discourse as input and outputs either True or False. Consider the two sentences "Socrates is a philosopher" and "Plato is a philosopher". In propositional logic, these sentences are viewed as being unrelated and are denoted, for example, by p and q . However, the predicate "is a philosopher" occurs in both sentences which have a common structure of " a is a philosopher". The variable a is instantiated as "Socrates" in first sentence and is instantiated as "Plato" in the second sentence. The use of predicates, such as "is a philosopher" in this example, distinguishes first-order logic from propositional logic.

Predicates can be compared. Consider, for example, the first-order formula "if a is a philosopher, then a is a scholar". This formula is a conditional statement with "a is a philosopher" as hypothesis and "a is a scholar" as conclusion. The truth of this formula depends on which object is denoted by a , and on the interpretations of the predicates "is a philosopher" and "is a scholar".

Variables can be quantified over. The variable a in the previous formula can be quantified over, for instance, in the first-order sentence "For every a , if a is a philosopher, then a is a scholar". The universal quantifier "for every" in this sentence expresses the idea that the claim "if a is a philosopher, then a is a scholar" holds for all choices of a .

The negation of the above sentence "For every a , if a is a philosopher, then a is a scholar" is logically equivalent to the sentence "There exists a such that a is a philosopher and a is not a scholar". The existential quantifier "there

"exists" expresses the idea that the claim " a is a philosopher and a is not a scholar" holds for *some* choice of a .

The predicates "is a philosopher" and "is a scholar" each take a single variable. Predicates can take several variables. In the first-order sentence "Socrates is the teacher of Plato", the predicate "is the teacher of" takes two variables.

To interpret a first-order formula, one specifies what each predicate means and the entities that can instantiate the predicated variables. These entities form the domain of discourse or universe, which is usually required to be a nonempty set. Given that the interpretation with the domain of discourse as consisting of all human beings and the predicate "is a philosopher" understood as "have written the Republic", the sentence "There exist a such that a is a philosopher" is seen as being true, as witnessed by Plato.

Syntax

There are two key parts of first order logic. The syntax determines which collections of symbols are legal expressions in first-order logic, while the semantics determine the meanings behind these expressions.

Alphabet

Unlike natural languages, such as English, the language of first-order logic is completely formal, so that it can be mechanically determined whether a given expression is legal. There are two key types of legal expressions: **terms**, which intuitively represent objects, and **formulas**, which intuitively express predicates that can be true or false. The terms and formulas of first-order logic are strings of **symbols** which together form the **alphabet** of the language. As with all formal languages, the nature of the symbols themselves is outside the scope of formal logic; they are often regarded simply as letters and punctuation symbols.

It is common to divide the symbols of the alphabet into **logical symbols**, which always have the same meaning, and **non-logical symbols**, whose meaning varies by interpretation. For example, the logical symbol \wedge always represents "and"; it is never interpreted as "or". On the other hand, a non-logical predicate symbol such as $\text{Phil}(x)$ could be interpreted to mean " x is a philosopher", " x is a man named Philip", or any other unary predicate, depending on the interpretation at hand.

Logical symbols

There are several logical symbols in the alphabet, which vary by author but usually include:

- The quantifier symbols \forall and \exists
- The logical connectives: \wedge for conjunction, \vee for disjunction, \rightarrow for implication, \leftrightarrow for biconditional, \neg for negation. Occasionally other logical connective symbols are included. Some authors use \rightarrow , or Cpq , instead of \rightarrow , and \leftrightarrow , or Epq , instead of \leftrightarrow , especially in contexts where \rightarrow is used for other purposes. Moreover, the horseshoe \supset may replace \rightarrow ; the triple-bar \equiv may replace \leftrightarrow , and a tilde (\sim), Np , or Fpq , may replace \neg ; \parallel , or Apq may replace \vee ; and $\&$, Kpq , or the middle dot, \cdot , may replace \wedge , especially if these symbols are not available for technical reasons. (Note: the aforementioned symbols Cpq , Epq , Np , Apq , and Kpq are used in Polish notation.)
- Parentheses, brackets, and other punctuation symbols. The choice of such symbols varies depending on context.
- An infinite set of **variables**, often denoted by lowercase letters at the end of the alphabet x, y, z, \dots . Subscripts are often used to distinguish variables: x_0, x_1, x_2, \dots
- An **equality symbol** (sometimes, **identity symbol**) $=$; see the section on equality below.

It should be noted that not all of these symbols are required – only one of the quantifiers, negation and conjunction, variables, brackets and equality suffice. There are numerous minor variations that may define additional logical symbols:

- Sometimes the truth constants T , Vpq , or \top , for "true" and F , Opq , or \perp , for "false" are included. Without any such logical operators of valence 0, these two constants can only be expressed using quantifiers.

- Sometimes additional logical connectives are included, such as the Sheffer stroke, Dpq (NAND), and exclusive or, Jpq .

Non-logical symbols

The non-logical symbols represent predicates (relations), functions and constants on the domain of discourse. It used to be standard practice to use a fixed, infinite set of non-logical symbols for all purposes. A more recent practice is to use different non-logical symbols according to the application one has in mind. Therefore it has become necessary to name the set of all non-logical symbols used in a particular application. This choice is made via a **signature**.^[1]

The traditional approach is to have only one, infinite, set of non-logical symbols (one signature) for all applications. Consequently, under the traditional approach there is only one language of first-order logic.^[2] This approach is still common, especially in philosophically oriented books.

1. For every integer $n \geq 0$ there is a collection of **n -ary**, or **n -place**, **predicate symbols**. Because they represent relations between n elements, they are also called **relation symbols**. For each arity n we have an infinite supply of them:

$$P_0^n, P_1^n, P_2^n, P_3^n, \dots$$

2. For every integer $n \geq 0$ there are infinitely many **n -ary function symbols**:

$$f_0^n, f_1^n, f_2^n, f_3^n, \dots$$

In contemporary mathematical logic, the signature varies by application. Typical signatures in mathematics are $\{1, \times\}$ or just $\{\times\}$ for groups, or $\{0, 1, +, \times, <\}$ for ordered fields. There are no restrictions on the number of non-logical symbols. The signature can be empty, finite, or infinite, even uncountable. Uncountable signatures occur for example in modern proofs of the Löwenheim-Skolem theorem.

In this approach, every non-logical symbol is of one of the following types.

1. A **predicate symbol** (or **relation symbol**) with some **valence** (or **arity**, number of arguments) greater than or equal to 0. These which are often denoted by uppercase letters P, Q, R, \dots .
 - Relations of valence 0 can be identified with propositional variables. For example, P , which can stand for any statement.
 - For example, $P(x)$ is a predicate variable of valence 1. One possible interpretation is "x is a man".
 - $Q(x,y)$ is a predicate variable of valence 2. Possible interpretations include "x is greater than y" and "x is the father of y".
2. A **function symbol**, with some valence greater than or equal to 0. These are often denoted by lowercase letters f, g, h, \dots .
 - Examples: $f(x)$ may be interpreted as for "the father of x". In arithmetic, it may stand for " $-x$ ". In set theory, it may stand for "the power set of x". In arithmetic, $g(x,y)$ may stand for " $x+y$ ". In set theory, it may stand for "the union of x and y".
 - Function symbols of valence 0 are called **constant symbols**, and are often denoted by lowercase letters at the beginning of the alphabet a, b, c, \dots . The symbol a may stand for Socrates. In arithmetic, it may stand for 0. In set theory, such a constant may stand for the empty set.

The traditional approach can be recovered in the modern approach by simply specifying the "custom" signature to consist of the traditional sequences of non-logical symbols.

Formation rules

The formation rules define the terms and formulas of first order logic. When terms and formulas are represented as strings of symbols, these rules can be used to write a formal grammar for terms and formulas. These rules are generally context-free (each production has a single symbol on the left side), except that the set of symbols may be allowed to be infinite and there may be many start symbols, for example the variables in the case of terms.

Terms

The set of **terms** is inductively defined by the following rules:

1. **Variables.** Any variable is a term.
2. **Functions.** Any expression $f(t_1, \dots, t_n)$ of n arguments (where each argument t_i is a term and f is a function symbol of valence n) is a term. In particular, symbols denoting individual constants are 0-ary function symbols, and are thus terms.

Only expressions which can be obtained by finitely many applications of rules 1 and 2 are terms. For example, no expression involving a predicate symbol is a term.

Formulas

The set of **formulas** (also called **well-formed formulas**^[3] or **wffs**) is inductively defined by the following rules:

1. **Predicate symbols.** If P is an n -ary predicate symbol and t_1, \dots, t_n are terms then $P(t_1, \dots, t_n)$ is a formula.
2. **Equality.** If the equality symbol is considered part of logic, and t_1 and t_2 are terms, then $t_1 = t_2$ is a formula.
3. **Negation.** If φ is a formula, then $\neg\varphi$ is a formula.
4. **Binary connectives.** If φ and ψ are formulas, then $(\varphi \rightarrow \psi)$ is a formula. Similar rules apply to other binary logical connectives.
5. **Quantifiers.** If φ is a formula and x is a variable, then $\forall x\varphi$ and $\exists x\varphi$ are formulas.

Only expressions which can be obtained by finitely many applications of rules 1–5 are formulas. The formulas obtained from the first two rules are said to be **atomic formulas**.

For example,

$$\forall x\forall y(P(f(x)) \rightarrow \neg(P(x) \rightarrow Q(f(y), x, z)))$$

is a formula, if f is a unary function symbol, P a unary predicate symbol, and Q a ternary predicate symbol. On the other hand, $\forall x x \rightarrow$ is not a formula, although it is a string of symbols from the alphabet.

The role of the parentheses in the definition is to ensure that any formula can only be obtained in one way by following the inductive definition (in other words, there is a unique parse tree for each formula). This property is known as **unique readability** of formulas. There are many conventions for where parentheses are used in formulas. For example, some authors use colons or full stops instead of parentheses, or change the places in which parentheses are inserted. Each author's particular definition must be accompanied by a proof of unique readability.

This definition of a formula does not support defining an if-then-else function `ite(c, a, b)`, where "c" is a condition expressed as a formula, that would return "a" if c is true, and "b" if it is false. This is because both predicates and functions can only accept terms as parameters, but the first parameter is a formula. Some languages built on first-order logic, such as SMT-LIB 2.0, add this.^[4]

Notational conventions

For convenience, conventions have been developed about the precedence of the logical operators, to avoid the need to write parentheses in some cases. These rules are similar to the order of operations in arithmetic. A common convention is:

- \neg is evaluated first
- \wedge and \vee are evaluated next
- Quantifiers are evaluated next
- \rightarrow is evaluated last.

Moreover, extra punctuation not required by the definition may be inserted to make formulas easier to read. Thus the formula

$$(\neg \forall x P(x) \rightarrow \exists x \neg P(x))$$

might be written as

$$(\neg [\forall x P(x)]) \rightarrow \exists x [\neg P(x)].$$

In some fields, it is common to use infix notation for binary relations and functions, instead of the prefix notation defined above. For example, in arithmetic, one typically writes "2 + 2 = 4" instead of "=+(2,2,4)". It is common to regard formulas in infix notation as abbreviations for the corresponding formulas in prefix notation.

The definitions above use infix notation for binary connectives such as \rightarrow . A less common convention is Polish notation, in which one writes \rightarrow , \wedge , and so on in front of their arguments rather than between them. This convention allows all punctuation symbols to be discarded. Polish notation is compact and elegant, but rarely used in practice because it is hard for humans to read it. In Polish notation, the formula

$$\forall x \forall y (P(f(x)) \rightarrow \neg(P(x) \rightarrow Q(f(y), x, z)))$$

becomes " $\forall x \forall y \rightarrow P f x \neg \rightarrow P x Q f y x z$ ".

Free and bound variables

In a formula, a variable may occur **free** or **bound**. Intuitively, a variable is free in a formula if it is not quantified: in $\forall y P(x, y)$, variable x is free while y is bound. The free and bound variables of a formula are defined inductively as follows.

1. **Atomic formulas.** If φ is an atomic formula then x is free in φ if and only if x occurs in φ . Moreover, there are no bound variables in any atomic formula.
2. **Negation.** x is free in $\neg \varphi$ if and only if x is free in φ . x is bound in $\neg \varphi$ if and only if x is bound in φ .
3. **Binary connectives.** x is free in $(\varphi \rightarrow \psi)$ if and only if x is free in either φ or ψ . x is bound in $(\varphi \rightarrow \psi)$ if and only if x is bound in either φ or ψ . The same rule applies to any other binary connective in place of \rightarrow .
4. **Quantifiers.** x is free in $\forall y \varphi$ if and only if x is free in φ and x is a different symbol from y . Also, x is bound in $\forall y \varphi$ if and only if x is y or x is bound in φ . The same rule holds with \exists in place of \forall .

For example, in $\forall x \forall y (P(x) \rightarrow Q(x, f(x), z))$, x and y are bound variables, z is a free variable, and w is neither because it does not occur in the formula.

Freeness and boundness can be also specialized to specific occurrences of variables in a formula. For example, in $P(x) \rightarrow \forall x Q(x)$, the first occurrence of x is free while the second is bound. In other words, the x in $P(x)$ is free while the x in $\forall x Q(x)$ is bound.

A formula in first-order logic with no free variables is called a **first-order sentence**. These are the formulas that will have well-defined truth values under an interpretation. For example, whether a formula such as $\text{Phil}(x)$ is true must depend on what x represents. But the sentence $\exists x \text{ Phil}(x)$ will be either true or false in a given interpretation.

Examples

Abelian groups

In mathematics the language of ordered abelian groups has one constant symbol 0, one unary function symbol $-$, one binary function symbol $+$, and one binary relation symbol \leq . Then:

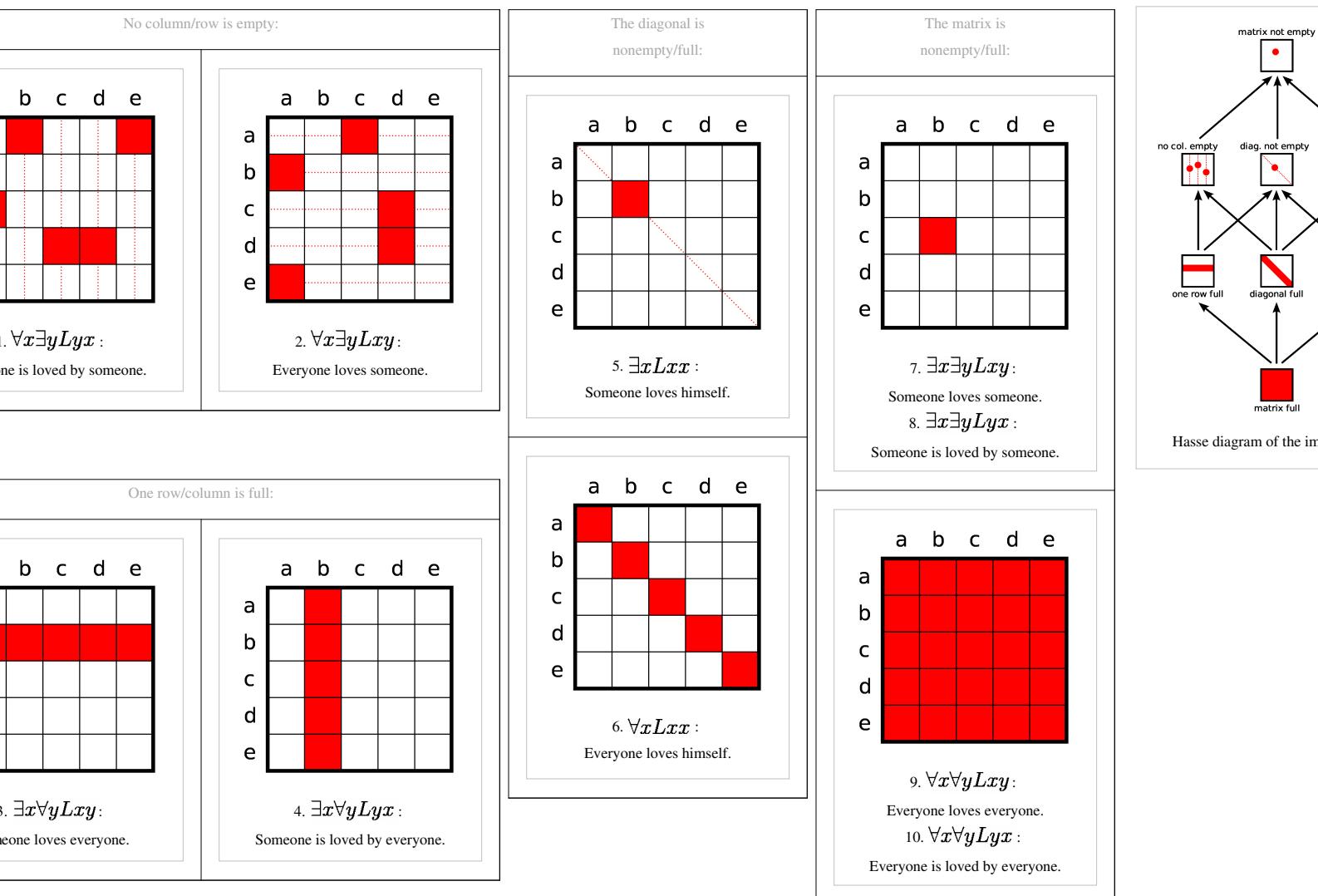
- The expressions $+(x, y)$ and $+(x, +(y, -(z)))$ are **terms**. These are usually written as $x + y$ and $x + y - z$.
- The expressions $+(x, y) = 0$ and $\leq(+(+x, +(y, -(z))), +(x, y))$ are **atomic formulas**.

These are usually written as $x + y = 0$ and $x + y - z \leq x + y$.

- The expression $(\forall x \forall y \leq(+(+x, y), z) \rightarrow \forall x \forall y (+(+x, y) = 0))$ is a **formula**, which is usually written as $\forall x \forall y (x + y \leq z) \rightarrow \forall x \forall y (x + y = 0)$.

Loving relation

There are 10 different formulas with 8 different meanings, that use the *loving* relation Lxy (" x loves y ") and the quantifiers \forall and \exists :



The logical matrices represent the formulas for the case that there are five individuals that can love (vertical axis) and be loved (horizontal axis). Except for the sentences 9 and 10, they are examples. E.g. the matrix representing sentence 5 stands for "b loves himself"; the matrix representing sentences 7 and 8 stands for "c loves b."

It's important and instructive to distinguish sentence 1, $\forall x \exists y Lxy$, and 3, $\exists x \forall y Lxy$: In both cases everyone is loved; but in the first case everyone is loved by someone, in the second case everyone is loved by the same person. Some sentences imply each other — e.g. if 3 is true also 1 is true, but not vice versa. (See Hasse diagram)

Semantics

An interpretation of a first-order language assigns a denotation to all non-logical constants in that language. It also determines a domain of discourse that specifies the range of the quantifiers. The result is that each term is assigned an object that it represents, and each sentence is assigned a truth value. In this way, an interpretation provides semantic meaning to the terms and formulas of the language. The study of the interpretations of formal languages is called formal semantics. What follows is a description of the standard or Tarskian semantics for first-order logic. (It is also possible to define game semantics for first-order logic, but aside from requiring the axiom of choice, game semantics agree with Tarskian semantics for first-order logic, so game semantics will not be elaborated herein.)

The domain of discourse D is a nonempty set of "objects" of some kind. Intuitively, a first-order formula is a statement about these objects; for example, $\exists x P(x)$ states the existence of an object x such that the predicate P is true where referred to it. The domain of discourse is the set of considered objects. For example, one can take D to be the set of integer numbers.

The interpretation of a function symbol is a function. For example, if the domain of discourse consists of integers, a function symbol f of arity 2 can be interpreted as the function that gives the sum of its arguments. In other words, the symbol f is associated with the function $I(f)$ which, in this interpretation, is addition.

The interpretation of a constant symbol is a function from the one-element set D^0 to D , which can be simply identified with an object in D . For example, an interpretation may assign the value $I(c) = 10$ to the constant symbol c .

The interpretation of an n -ary predicate symbol is a set of n -tuples of elements of the domain of discourse. This means that, given an interpretation, a predicate symbol, and n elements of the domain of discourse, one can tell whether the predicate is true of those elements according to the given interpretation. For example, an interpretation $I(P)$ of a binary predicate symbol P may be the set of pairs of integers such that the first one is less than the second. According to this interpretation, the predicate P would be true if its first argument is less than the second.

First-order structures

The most common way of specifying an interpretation (especially in mathematics) is to specify a **structure** (also called a **model**; see below). The structure consists of a nonempty set D that forms the domain of discourse and an interpretation I of the non-logical terms of the signature. This interpretation is itself a function:

- Each function symbol f of arity n is assigned a function $I(f)$ from D^n to D . In particular, each constant symbol of the signature is assigned an individual in the domain of discourse.
- Each predicate symbol P of arity n is assigned a relation $I(P)$ over D^n or, equivalently, a function from D^n to $\{\text{true}, \text{false}\}$. Thus each predicate symbol is interpreted by a Boolean-valued function on D .

Evaluation of truth values

A formula evaluates to true or false given an interpretation, and a **variable assignment** μ that associates an element of the domain of discourse with each variable. The reason that a variable assignment is required is to give meanings to formulas with free variables, such as $y = x$. The truth value of this formula changes depending on whether x and y denote the same individual.

First, the variable assignment μ can be extended to all terms of the language, with the result that each term maps to a single element of the domain of discourse. The following rules are used to make this assignment:

1. **Variables.** Each variable x evaluates to $\mu(x)$
2. **Functions.** Given terms t_1, \dots, t_n that have been evaluated to elements d_1, \dots, d_n of the domain of discourse, and a n -ary function symbol f , the term $f(t_1, \dots, t_n)$ evaluates to $(I(f))(d_1, \dots, d_n)$.

Next, each formula is assigned a truth value. The inductive definition used to make this assignment is called the T-schema.

1. **Atomic formulas (1).** A formula $P(t_1, \dots, t_n)$ is associated the value true or false depending on whether $\langle v_1, \dots, v_n \rangle \in I(P)$, where v_1, \dots, v_n are the evaluation of the terms t_1, \dots, t_n and $I(P)$ is the interpretation of P , which by assumption is a subset of D^n .
2. **Atomic formulas (2).** A formula $t_1 = t_2$ is assigned true if t_1 and t_2 evaluate to the same object of the domain of discourse (see the section on equality below).
3. **Logical connectives.** A formula in the form $\neg\phi$, $\phi \rightarrow \psi$, etc. is evaluated according to the truth table for the connective in question, as in propositional logic.
4. **Existential quantifiers.** A formula $\exists x\phi(x)$ is true according to M and μ if there exists an evaluation μ' of the variables that only differs from μ regarding the evaluation of x and such that ϕ is true according to the interpretation M and the variable assignment μ' . This formal definition captures the idea that $\exists x\phi(x)$ is true if and only if there is a way to choose a value for x such that $\phi(x)$ is satisfied.
5. **Universal quantifiers.** A formula $\forall x\phi(x)$ is true according to M and μ if $\phi(x)$ is true for every pair composed by the interpretation M and some variable assignment μ' that differs from μ only on the value of x . This captures the idea that $\forall x\phi(x)$ is true if every possible choice of a value for x causes $\phi(x)$ to be true.

If a formula does not contain free variables, and so is a sentence, then the initial variable assignment does not affect its truth value. In other words, a sentence is true according to M and μ if and only if it is true according to M and every other variable assignment μ' .

There is a second common approach to defining truth values that does not rely on variable assignment functions. Instead, given an interpretation M , one first adds to the signature a collection of constant symbols, one for each element of the domain of discourse in M ; say that for each d in the domain the constant symbol c_d is fixed. The interpretation is extended so that each new constant symbol is assigned to its corresponding element of the domain. One now defines truth for quantified formulas syntactically, as follows:

1. **Existential quantifiers (alternate).** A formula $\exists x\phi(x)$ is true according to M if there is some d in the domain of discourse such that $\phi(c_d)$ holds. Here $\phi(c_d)$ is the result of substituting c_d for every free occurrence of x in ϕ .
2. **Universal quantifiers (alternate).** A formula $\forall x\phi(x)$ is true according to M if, for every d in the domain of discourse, $\phi(c_d)$ is true according to M .

This alternate approach gives exactly the same truth values to all sentences as the approach via variable assignments.

Validity, satisfiability, and logical consequence

If a sentence φ evaluates to True under a given interpretation M , one says that M **satisfies** φ ; this is denoted $M \models \varphi$. A sentence is **satisfiable** if there is some interpretation under which it is true.

Satisfiability of formulas with free variables is more complicated, because an interpretation on its own does not determine the truth value of such a formula. The most common convention is that a formula with free variables is said to be satisfied by an interpretation if the formula remains true regardless which individuals from the domain of discourse are assigned to its free variables. This has the same effect as saying that a formula is satisfied if and only if its universal closure is satisfied.

A formula is **logically valid** (or simply **valid**) if it is true in every interpretation. These formulas play a role similar to tautologies in propositional logic.

A formula φ is a **logical consequence** of a formula ψ if every interpretation that makes ψ true also makes φ true. In this case one says that φ is logically implied by ψ .

Algebraizations

An alternate approach to the semantics of first-order logic proceeds via abstract algebra. This approach generalizes the Lindenbaum–Tarski algebras of propositional logic. There are three ways of eliminating quantified variables from first-order logic that do not involve replacing quantifiers with other variable binding term operators:

- Cylindric algebra, by Alfred Tarski and his coworkers;
- Polyadic algebra, by Paul Halmos;
- Predicate functor logic, mainly due to Willard Quine.

These algebras are all lattices that properly extend the two-element Boolean algebra.

Tarski and Givant (1987) showed that the fragment of first-order logic that has no atomic sentence lying in the scope of more than three quantifiers has the same expressive power as relation algebra. This fragment is of great interest because it suffices for Peano arithmetic and most axiomatic set theory, including the canonical ZFC. They also prove that first-order logic with a primitive ordered pair is equivalent to a relation algebra with two ordered pair projection functions.

First-order theories, models, and elementary classes

A **first-order theory** consists of a set of axioms in a particular first-order signature. The set of axioms is often finite or recursively enumerable, in which case the theory is called **effective**. Some authors require theories to also include all logical consequences of the axioms.

A first-order structure that satisfies all sentences in a given theory is said to be a **model** of the theory. An **elementary class** is the set of all structures satisfying a particular theory. These classes are a main subject of study in model theory.

Many theories have an intended interpretation, a certain model that is kept in mind when studying the theory. For example, the intended interpretation of Peano arithmetic consists of the usual natural numbers with their usual operations. However, the Löwenheim–Skolem theorem shows that most first-order theories will also have other, nonstandard models.

A theory is **consistent** if it is not possible to prove a contradiction from the axioms of the theory. A theory is **complete** if, for every formula in its signature, either that formula or its negation is a logical consequence of the axioms of the theory. Gödel's incompleteness theorem shows that effective first-order theories that include a sufficient portion of the theory of the natural numbers can never be both consistent and complete.

Empty domains

The definition above requires that the domain of discourse of any interpretation must be a nonempty set. There are settings, such as inclusive logic, where empty domains are permitted. Moreover, if a class of algebraic structures includes an empty structure (for example, there is an empty poset), that class can only be an elementary class in first-order logic if empty domains are permitted or the empty structure is removed from the class.

There are several difficulties with empty domains, however:

- Many common rules of inference are only valid when the domain of discourse is required to be nonempty. One example is the rule stating that $\phi \vee \exists x\psi$ implies $\exists x(\phi \vee \psi)$ when x is not a free variable in ϕ . This rule, which is used to put formulas into prenex normal form, is sound in nonempty domains, but unsound if the empty domain is permitted.
- The definition of truth in an interpretation that uses a variable assignment function cannot work with empty domains, because there are no variable assignment functions whose range is empty. (Similarly, one cannot assign interpretations to constant symbols.) This truth definition requires that one must select a variable assignment function (μ above) before truth values for even atomic formulas can be defined. Then the truth value of a sentence is defined to be its truth value under any variable assignment, and it is proved that this truth value does not depend on which assignment is chosen. This technique does not work if there are no assignment functions at all; it must be changed to accommodate empty domains.

Thus, when the empty domain is permitted, it must often be treated as a special case. Most authors, however, simply exclude the empty domain by definition.

Deductive systems

A **deductive system** is used to demonstrate, on a purely syntactic basis, that one formula is a logical consequence of another formula. There are many such systems for first-order logic, including Hilbert-style deductive systems, natural deduction, the sequent calculus, the tableaux method, and resolution. These share the common property that a deduction is a finite syntactic object; the format of this object, and the way it is constructed, vary widely. These finite deductions themselves are often called **derivations** in proof theory. They are also often called proofs, but are completely formalized unlike natural-language mathematical proofs.

A deductive system is **sound** if any formula that can be derived in the system is logically valid. Conversely, a deductive system is **complete** if every logically valid formula is derivable. All of the systems discussed in this article are both sound and complete. They also share the property that it is possible to effectively verify that a purportedly valid deduction is actually a deduction; such deduction systems are called **effective**.

A key property of deductive systems is that they are purely syntactic, so that derivations can be verified without considering any interpretation. Thus a sound argument is correct in every possible interpretation of the language, regardless whether that interpretation is about mathematics, economics, or some other area.

In general, logical consequence in first-order logic is only semidecidable: if a sentence A logically implies a sentence B then this can be discovered (for example, by searching for a proof until one is found, using some effective, sound, complete proof system). However, if A does not logically imply B, this does not mean that A logically implies the negation of B. There is no effective procedure that, given formulas A and B, always correctly decides whether A logically implies B.

Rules of inference

A **rule of inference** states that, given a particular formula (or set of formulas) with a certain property as a hypothesis, another specific formula (or set of formulas) can be derived as a conclusion. The rule is sound (or truth-preserving) if it preserves validity in the sense that whenever any interpretation satisfies the hypothesis, that interpretation also satisfies the conclusion.

For example, one common rule of inference is the **rule of substitution**. If t is a term and φ is a formula possibly containing the variable x , then $\varphi[t/x]$ (often denoted $\varphi[x/t]$) is the result of replacing all free instances of x by t in φ . The substitution rule states that for any φ and any term t , one can conclude $\varphi[t/x]$ from φ provided that no free variable of t becomes bound during the substitution process. (If some free variable of t becomes bound, then to substitute t for x it is first necessary to change the bound variables of φ to differ from the free variables of t .)

To see why the restriction on bound variables is necessary, consider the logically valid formula φ given by $\exists x(x = y)$, in the signature of $(0, 1, +, \times, =)$ of arithmetic. If t is the term " $x + 1$ ", the formula $\varphi[t/y]$ is $\exists x(x = x + 1)$, which will be false in many interpretations. The problem is that the free variable x of t became bound during the substitution. The intended replacement can be obtained by renaming the bound variable x of φ to something else, say z , so that the formula after substitution is $\exists z(z = x + 1)$, which is again logically valid. The substitution rule demonstrates several common aspects of rules of inference. It is entirely syntactical; one can tell whether it was correctly applied without appeal to any interpretation. It has (syntactically-defined) limitations on when it can be applied, which must be respected to preserve the correctness of derivations. Moreover, as is often the case, these limitations are necessary because of interactions between free and bound variables that occur during syntactic manipulations of the formulas involved in the inference rule.

Hilbert-style systems and natural deduction

A deduction in a Hilbert-style deductive system is a list of formulas, each of which is a **logical axiom**, a hypothesis that has been assumed for the derivation at hand, or follows from previous formulas via a rule of inference. The logical axioms consist of several axiom schemes of logically valid formulas; these encompass a significant amount of propositional logic. The rules of inference enable the manipulation of quantifiers. Typical Hilbert-style systems have a small number of rules of inference, along with several infinite schemes of logical axioms. It is common to have only modus ponens and universal generalization as rules of inference.

Natural deduction systems resemble Hilbert-style systems in that a deduction is a finite list of formulas. However, natural deduction systems have no logical axioms; they compensate by adding additional rules of inference that can be used to manipulate the logical connectives in formulas in the proof.

Sequent calculus

The sequent calculus was developed to study the properties of natural deduction systems. Instead of working with one formula at a time, it uses **sequents**, which are expressions of the form

$$A_1, \dots, A_n \vdash B_1, \dots, B_k,$$

where $A_1, \dots, A_n, B_1, \dots, B_k$ are formulas and the turnstile symbol \vdash is used as punctuation to separate the two halves. Intuitively, a sequent expresses the idea that $(A_1 \wedge \dots \wedge A_n) \text{ implies } (B_1 \vee \dots \vee B_k)$.

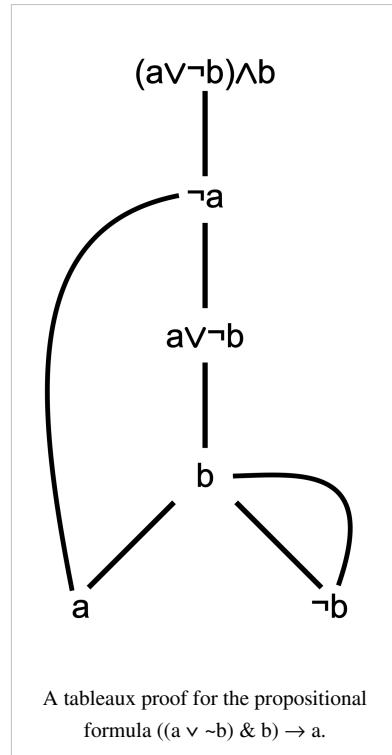
Tableaux method

Unlike the methods just described, the derivations in the tableaux method are not lists of formulas. Instead, a derivation is a tree of formulas. To show that a formula A is provable, the tableaux method attempts to demonstrate that the negation of A is unsatisfiable. The tree of the derivation has $\neg A$ at its root; the tree branches in a way that reflects the structure of the formula. For example, to show that $C \vee D$ is unsatisfiable requires showing that C and D are each unsatisfiable; this corresponds to a branching point in the tree with parent $C \vee D$ and children C and D .

Resolution

The resolution rule is a single rule of inference that, together with unification, is sound and complete for first-order logic. As with the tableaux method, a formula is proved by showing that the negation of the formula is unsatisfiable. Resolution is commonly used in automated theorem proving.

The resolution method works only with formulas that are disjunctions of atomic formulas; arbitrary formulas must first be converted to this form through Skolemization. The resolution rule states that from the hypotheses $A_1 \vee \dots \vee A_k \vee C$ and $B_1 \vee \dots \vee B_l \vee \neg C$, the conclusion $A_1 \vee \dots \vee A_k \vee B_1 \vee \dots \vee B_l$ can be obtained.



Provable identities

The following sentences can be called "identities" because the main connective in each is the biconditional.

$$\begin{aligned} \neg \forall x P(x) &\Leftrightarrow \exists x \neg P(x) \\ \neg \exists x P(x) &\Leftrightarrow \forall x \neg P(x) \\ \forall x \forall y P(x, y) &\Leftrightarrow \forall y \forall x P(x, y) \\ \exists x \exists y P(x, y) &\Leftrightarrow \exists y \exists x P(x, y) \\ \forall x P(x) \wedge \forall x Q(x) &\Leftrightarrow \forall x (P(x) \wedge Q(x)) \\ \exists x P(x) \vee \exists x Q(x) &\Leftrightarrow \exists x (P(x) \vee Q(x)) \\ P \wedge \exists x Q(x) &\Leftrightarrow \exists x (P \wedge Q(x)) \text{ (where } x \text{ must not occur free in } P\text{)} \\ P \vee \forall x Q(x) &\Leftrightarrow \forall x (P \vee Q(x)) \text{ (where } x \text{ must not occur free in } P\text{)} \end{aligned}$$

Equality and its axioms

There are several different conventions for using equality (or identity) in first-order logic. The most common convention, known as **first-order logic with equality**, includes the equality symbol as a primitive logical symbol which is always interpreted as the real equality relation between members of the domain of discourse, such that the "two" given members are the same member. This approach also adds certain axioms about equality to the deductive system employed. These equality axioms are:

1. **Reflexivity.** For each variable x , $x = x$.
 2. **Substitution for functions.** For all variables x and y , and any function symbol f ,
- $$x = y \rightarrow f(\dots, x, \dots) = f(\dots, y, \dots).$$
3. **Substitution for formulas.** For any variables x and y and any formula $\varphi(x)$, if φ' is obtained by replacing any number of free occurrences of x in φ with y , such that these remain free occurrences of y , then

$$x = y \rightarrow (\varphi \rightarrow \varphi').$$

These are axiom schemes, each of which specifies an infinite set of axioms. The third scheme is known as **Leibniz's law**, "the principle of substitutivity", "the indiscernibility of identicals", or "the replacement property". The second scheme, involving the function symbol f , is (equivalent to) a special case of the third scheme, using the formula

$$x = y \rightarrow (f(\dots, x, \dots) = z \rightarrow f(\dots, y, \dots) = z).$$

Many other properties of equality are consequences of the axioms above, for example:

1. **Symmetry.** If $x = y$ then $y = x$.
2. **Transitivity.** If $x = y$ and $y = z$ then $x = z$.

First-order logic without equality

An alternate approach considers the equality relation to be a non-logical symbol. This convention is known as **first-order logic without equality**. If an equality relation is included in the signature, the axioms of equality must now be added to the theories under consideration, if desired, instead of being considered rules of logic. The main difference between this method and first-order logic with equality is that an interpretation may now interpret two distinct individuals as "equal" (although, by Leibniz's law, these will satisfy exactly the same formulas under any interpretation). That is, the equality relation may now be interpreted by an arbitrary equivalence relation on the domain of discourse that is congruent with respect to the functions and relations of the interpretation.

When this second convention is followed, the term **normal model** is used to refer to an interpretation where no distinct individuals a and b satisfy $a = b$. In first-order logic with equality, only normal models are considered, and so there is no term for a model other than a normal model. When first-order logic without equality is studied, it is necessary to amend the statements of results such as the Löwenheim–Skolem theorem so that only normal models are considered.

First-order logic without equality is often employed in the context of second-order arithmetic and other higher-order theories of arithmetic, where the equality relation between sets of natural numbers is usually omitted.

Defining equality within a theory

If a theory has a binary formula $A(x,y)$ which satisfies reflexivity and Leibniz's law, the theory is said to have equality, or to be a theory with equality. The theory may not have all instances of the above schemes as axioms, but rather as derivable theorems. For example, in theories with no function symbols and a finite number of relations, it is possible to define equality in terms of the relations, by defining the two terms s and t to be equal if any relation is unchanged by changing s to t in any argument.

Some theories allow other *ad hoc* definitions of equality:

- In the theory of partial orders with one relation symbol \leq , one could define $s = t$ to be an abbreviation for $s \leq t \wedge t \leq s$.
- In set theory with one relation \in , one may define $s = t$ to be an abbreviation for $\forall x(s \in x \leftrightarrow t \in x) \wedge \forall x(x \in s \leftrightarrow x \in t)$. This definition of equality then automatically satisfies the axioms for equality. In this case, one should replace the usual axiom of extensionality, $\forall x \forall y [\forall z(z \in x \leftrightarrow z \in y) \Rightarrow x = y]$, by $\forall x \forall y [\forall z(z \in x \leftrightarrow z \in y) \Rightarrow \forall z(x \in z \leftrightarrow y \in z)]$, i.e. if x and y have the same elements, then they belong to the same sets.

Metalogical properties

One motivation for the use of first-order logic, rather than higher-order logic, is that first-order logic has many metalogical properties that stronger logics do not have. These results concern general properties of first-order logic itself, rather than properties of individual theories. They provide fundamental tools for the construction of models of first-order theories.

Completeness and undecidability

Gödel's completeness theorem, proved by Kurt Gödel in 1929, establishes that there are sound, complete, effective deductive systems for first-order logic, and thus the first-order logical consequence relation is captured by finite provability. Naively, the statement that a formula φ logically implies a formula ψ depends on every model of φ ; these models will in general be of arbitrarily large cardinality, and so logical consequence cannot be effectively verified by checking every model. However, it is possible to enumerate all finite derivations and search for a derivation of ψ from φ . If ψ is logically implied by φ , such a derivation will eventually be found. Thus first-order logical consequence is semidecidable: it is possible to make an effective enumeration of all pairs of sentences (φ, ψ) such that ψ is a logical consequence of φ .

Unlike propositional logic, first-order logic is undecidable (although semidecidable), provided that the language has at least one predicate of arity at least 2 (other than equality). This means that there is no decision procedure that determines whether arbitrary formulas are logically valid. This result was established independently by Alonzo Church and Alan Turing in 1936 and 1937, respectively, giving a negative answer to the Entscheidungsproblem posed by David Hilbert in 1928. Their proofs demonstrate a connection between the unsolvability of the decision problem for first-order logic and the unsolvability of the halting problem.

There are systems weaker than full first-order logic for which the logical consequence relation is decidable. These include propositional logic and monadic predicate logic, which is first-order logic restricted to unary predicate symbols and no function symbols. The Bernays–Schönfinkel class of first-order formulas is also decidable. Decidable subsets of first-order logic are also studied in the framework of description logics.

The Löwenheim–Skolem theorem

The Löwenheim–Skolem theorem shows that if a first-order theory of cardinality λ has any infinite model then it has models of every infinite cardinality greater than or equal to λ . One of the earliest results in model theory, it implies that it is not possible to characterize countability or uncountability in a first-order language. That is, there is no first-order formula $\varphi(x)$ such that an arbitrary structure M satisfies φ if and only if the domain of discourse of M is countable (or, in the second case, uncountable).

The Löwenheim–Skolem theorem implies that infinite structures cannot be categorically axiomatized in first-order logic. For example, there is no first-order theory whose only model is the real line: any first-order theory with an infinite model also has a model of cardinality larger than the continuum. Since the real line is infinite, any theory satisfied by the real line is also satisfied by some nonstandard models. When the Löwenheim–Skolem theorem is applied to first-order set theories, the nonintuitive consequences are known as Skolem's paradox.

The compactness theorem

The compactness theorem states that a set of first-order sentences has a model if and only if every finite subset of it has a model. This implies that if a formula is a logical consequence of an infinite set of first-order axioms, then it is a logical consequence of some finite number of those axioms. This theorem was proved first by Kurt Gödel as a consequence of the completeness theorem, but many additional proofs have been obtained over time. It is a central tool in model theory, providing a fundamental method for constructing models.

The compactness theorem has a limiting effect on which collections of first-order structures are elementary classes. For example, the compactness theorem implies that any theory that has arbitrarily large finite models has an infinite model. Thus the class of all finite graphs is not an elementary class (the same holds for many other algebraic structures).

There are also more subtle limitations of first-order logic that are implied by the compactness theorem. For example, in computer science, many situations can be modeled as a directed graph of states (nodes) and connections (directed edges). Validating such a system may require showing that no "bad" state can be reached from any "good" state. Thus one seeks to determine if the good and bad states are in different connected components of the graph. However, the compactness theorem can be used to show that connected graphs are not an elementary class in first-order logic, and there is no formula $\varphi(x,y)$ of first-order logic, in the signature of graphs, that expresses the idea that there is a path from x to y . Connectedness can be expressed in second-order logic, however, but not with only existential set quantifiers, as Σ_1^1 also enjoys compactness.

Lindström's theorem

Per Lindström showed that the metalogical properties just discussed actually characterize first-order logic in the sense that no stronger logic can also have those properties (Ebbinghaus and Flum 1994, Chapter XIII). Lindström defined a class of abstract logical systems, and a rigorous definition of the relative strength of a member of this class. He established two theorems for systems of this type:

- A logical system satisfying Lindström's definition that contains first-order logic and satisfies both the Löwenheim–Skolem theorem and the compactness theorem must be equivalent to first-order logic.
- A logical system satisfying Lindström's definition that has a semidecidable logical consequence relation and satisfies the Löwenheim–Skolem theorem must be equivalent to first-order logic.

Limitations

Although first-order logic is sufficient for formalizing much of mathematics, and is commonly used in computer science and other fields, it has certain limitations. These include limitations on its expressiveness and limitations of the fragments of natural languages that it can describe.

For instance, first-order logic is undecidable, meaning a sound, complete and terminating decision algorithm is impossible. This has led to the study of interesting decidable fragments such as C_2 , first-order logic with two variables and the counting quantifiers $\exists^{\geq n}$ and $\exists^{\leq n}$ (these quantifiers are, respectively, "there exists at least n " and "there exists at most n ") (Horrocks 2010).

Expressiveness

The Löwenheim–Skolem theorem shows that if a first-order theory has any infinite model, then it has infinite models of every cardinality. In particular, no first-order theory with an infinite model can be categorical. Thus there is no first-order theory whose only model has the set of natural numbers as its domain, or whose only model has the set of real numbers as its domain. Many extensions of first-order logic, including infinitary logics and higher-order logics, are more expressive in the sense that they do permit categorical axiomatizations of the natural numbers or real numbers. This expressiveness comes at a metalogical cost, however: by Lindström's theorem, the compactness theorem and the downward Löwenheim–Skolem theorem cannot hold in any logic stronger than first-order.

Formalizing natural languages

First-order logic is able to formalize many simple quantifier constructions in natural language, such as "every person who lives in Perth lives in Australia". But there are many more complicated features of natural language that cannot be expressed in (single-sorted) first-order logic. "Any logical system which is appropriate as an instrument for the analysis of natural language needs a much richer structure than first-order predicate logic" (Gamut 1991, p. 75).

Type	Example	Comment
Quantification over properties	If John is self-satisfied, then there is at least one thing he has in common with Peter	Requires a quantifier over predicates, which cannot be implemented in single-sorted first-order logic: $Zj \rightarrow \exists X(Xj \wedge Xp)$
Quantification over properties	Santa Claus has all the attributes of a sadist	Requires quantifiers over predicates, which cannot be implemented in single-sorted first-order logic: $\forall X(\forall x(Sx \rightarrow Xx) \rightarrow Xs)$
Predicate adverbial	John is walking quickly	Cannot be analysed as $Wj \wedge Qj$; predicate adverbials are not the same kind of thing as second-order predicates such as colour
Relative adjective	Jumbo is a small elephant	Cannot be analysed as $Sj \wedge Ej$; predicate adjectives are not the same kind of thing as second-order predicates such as colour
Predicate adverbial modifier	John is walking very quickly	-
Relative adjective modifier	Jumbo is terribly small	An expression such as "terribly", when applied to a relative adjective such as "small", results in a new composite relative adjective "terribly small"
Prepositions	Mary is sitting next to John	The preposition "next to" when applied to "John" results in the predicate adverbial "next to John"

Restrictions, extensions, and variations

There are many variations of first-order logic. Some of these are inessential in the sense that they merely change notation without affecting the semantics. Others change the expressive power more significantly, by extending the semantics through additional quantifiers or other new logical symbols. For example, infinitary logics permit formulas of infinite size, and modal logics add symbols for possibility and necessity.

Restricted languages

First-order logic can be studied in languages with fewer logical symbols than were described above.

- Because $\exists x\phi(x)$ can be expressed as $\neg\forall x\neg\phi(x)$, and $\forall x\phi(x)$ can be expressed as $\neg\exists x\neg\phi(x)$, either of the two quantifiers \exists and \forall can be dropped.
- Since $\phi \vee \psi$ can be expressed as $\neg(\neg\phi \wedge \neg\psi)$ and $\phi \wedge \psi$ can be expressed as $\neg(\neg\phi \vee \neg\psi)$, either \vee or \wedge can be dropped. In other words, it is sufficient to have \neg and \vee , or \neg and \wedge , as the only logical connectives.
- Similarly, it is sufficient to have only \neg and \rightarrow as logical connectives, or to have only the Sheffer stroke (NAND) or the Peirce arrow (NOR) operator.
- It is possible to entirely avoid function symbols and constant symbols, rewriting them via predicate symbols in an appropriate way. For example, instead of using a constant symbol 0 one may use a predicate $0(x)$ (interpreted as $x = 0$), and replace every predicate such as $P(0, y)$ with $\forall x (0(x) \rightarrow P(x, y))$. A function such as $f(x_1, x_2, \dots, x_n)$ will similarly be replaced by a predicate $F(x_1, x_2, \dots, x_n, y)$ interpreted as $y = f(x_1, x_2, \dots, x_n)$. This change requires adding additional axioms to the theory at hand, so that interpretations of the predicate symbols used have the correct semantics.

Restrictions such as these are useful as a technique to reduce the number of inference rules or axiom schemes in deductive systems, which leads to shorter proofs of metalogical results. The cost of the restrictions is that it becomes more difficult to express natural-language statements in the formal system at hand, because the logical connectives

used in the natural language statements must be replaced by their (longer) definitions in terms of the restricted collection of logical connectives. Similarly, derivations in the limited systems may be longer than derivations in systems that include additional connectives. There is thus a trade-off between the ease of working within the formal system and the ease of proving results about the formal system.

It is also possible to restrict the arities of function symbols and predicate symbols, in sufficiently expressive theories. One can in principle dispense entirely with functions of arity greater than 2 and predicates of arity greater than 1 in theories that include a pairing function. This is a function of arity 2 that takes pairs of elements of the domain and returns an ordered pair containing them. It is also sufficient to have two predicate symbols of arity 2 that define projection functions from an ordered pair to its components. In either case it is necessary that the natural axioms for a pairing function and its projections are satisfied.

Many-sorted logic

Ordinary first-order interpretations have a single domain of discourse over which all quantifiers range. **Many-sorted first-order logic** allows variables to have different **sorts**, which have different domains. This is also called **typed first-order logic**, and the sorts called **types** (as in data type), but it is not the same as first-order type theory. Many-sorted first-order logic is often used in the study of second-order arithmetic.

When there are only finitely many sorts in a theory, many-sorted first-order logic can be reduced to single-sorted first-order logic. One introduces into the single-sorted theory a unary predicate symbol for each sort in the many-sorted theory, and adds an axiom saying that these unary predicates partition the domain of discourse. For example, if there are two sorts, one adds predicate symbols $P_1(x)$ and $P_2(x)$ and the axiom

$$\forall x(P_1(x) \vee P_2(x)) \wedge \neg \exists x(P_1(x) \wedge P_2(x)).$$

Then the elements satisfying P_1 are thought of as elements of the first sort, and elements satisfying P_2 as elements of the second sort. One can quantify over each sort by using the corresponding predicate symbol to limit the range of quantification. For example, to say there is an element of the first sort satisfying formula $\phi(x)$, one writes

$$\exists x(P_1(x) \wedge \phi(x)).$$

Additional quantifiers

Additional quantifiers can be added to first-order logic.

- Sometimes it is useful to say that " $P(x)$ holds for exactly one x ", which can be expressed as $\exists!x P(x)$. This notation, called uniqueness quantification, may be taken to abbreviate a formula such as $\exists x (P(x) \wedge \forall y (P(y) \rightarrow (x = y)))$.
- **First-order logic with extra quantifiers** has new quantifiers Qx, \dots , with meanings such as "there are many x such that ...". Also see branching quantifiers and the plural quantifiers of George Boolos and others.
- **Bounded quantifiers** are often used in the study of set theory or arithmetic.

Infinitary logics

Infinitary logic allows infinitely long sentences. For example, one may allow a conjunction or disjunction of infinitely many formulas, or quantification over infinitely many variables. Infinitely long sentences arise in areas of mathematics including topology and model theory.

Infinitary logic generalizes first-order logic to allow formulas of infinite length. The most common way in which formulas can become infinite is through infinite conjunctions and disjunctions. However, it is also possible to admit generalized signatures in which function and relation symbols are allowed to have infinite arities, or in which quantifiers can bind infinitely many variables. Because an infinite formula cannot be represented by a finite string, it is necessary to choose some other representation of formulas; the usual representation in this context is a tree. Thus formulas are, essentially, identified with their parse trees, rather than with the strings being parsed.

The most commonly studied infinitary logics are denoted $L_{\alpha\beta}$, where α and β are each either cardinal numbers or the symbol ∞ . In this notation, ordinary first-order logic is $L_{\omega\omega}$. In the logic $L_{\infty\omega}$, arbitrary conjunctions or disjunctions are allowed when building formulas, and there is an unlimited supply of variables. More generally, the logic that permits conjunctions or disjunctions with less than κ constituents is known as $L_{\kappa\omega}$. For example, $L_{\omega}1\omega$ permits countable conjunctions and disjunctions.

The set of free variables in a formula of $L_{\kappa\omega}$ can have any cardinality strictly less than κ , yet only finitely many of them can be in the scope of any quantifier when a formula appears as a subformula of another.^[5] In other infinitary logics, a subformula may be in the scope of infinitely many quantifiers. For example, in $L_{\kappa\infty}$, a single universal or existential quantifier may bind arbitrarily many variables simultaneously. Similarly, the logic $L_{\kappa\lambda}$ permits simultaneous quantification over fewer than λ variables, as well as conjunctions and disjunctions of size less than κ .

Non-classical and modal logics

- **Intuitionistic first-order logic** uses intuitionistic rather than classical propositional calculus; for example, $\neg\neg\varphi$ need not be equivalent to φ .
- First-order **modal logic** allows one to describe other possible worlds as well as this contingently true world which we inhabit. In some versions, the set of possible worlds varies depending on which possible world one inhabits. Modal logic has extra *modal operators* with meanings which can be characterized informally as, for example "it is necessary that φ " (true in all possible worlds) and "it is possible that φ " (true in some possible world). With standard first-order logic we have a single domain and each predicate is assigned one extension. With first-order modal logic we have a *domain function* that assigns each possible world its own domain, so that each predicate gets an extension only relative to these possible worlds. This allows us to model cases where, for example, Alex is a Philosopher, but might have been a Mathematician, and might not have existed at all. In the first possible world $P(a)$ is true, in the second $P(a)$ is false, and in the third possible world there is no a in the domain at all.
- **first-order fuzzy logics** are first-order extensions of propositional fuzzy logics rather than classical propositional calculus.

Higher-order logics

The characteristic feature of first-order logic is that individuals can be quantified, but not predicates. Thus

$$\exists a(\text{Phil}(a))$$

is a legal first-order formula, but

$$\exists \text{Phil}(\text{Phil}(a))$$

is not, in most formalizations of first-order logic. Second-order logic extends first-order logic by adding the latter type of quantification. Other higher-order logics allow quantification over even higher types than second-order logic permits. These higher types include relations between relations, functions from relations to relations between relations, and other higher-type objects. Thus the "first" in first-order logic describes the type of objects that can be quantified.

Unlike first-order logic, for which only one semantics is studied, there are several possible semantics for second-order logic. The most commonly employed semantics for second-order and higher-order logic is known as **full semantics**. The combination of additional quantifiers and the full semantics for these quantifiers makes higher-order logic stronger than first-order logic. In particular, the (semantic) logical consequence relation for second-order and higher-order logic is not semidecidable; there is no effective deduction system for second-order logic that is sound and complete under full semantics.

Second-order logic with full semantics is more expressive than first-order logic. For example, it is possible to create axiom systems in second-order logic that uniquely characterize the natural numbers and the real line. The cost of this expressiveness is that second-order and higher-order logics have fewer attractive metalogical properties than

first-order logic. For example, the Löwenheim–Skolem theorem and compactness theorem of first-order logic become false when generalized to higher-order logics with full semantics.

Automated theorem proving and formal methods

Automated theorem proving refers to the development of computer programs that search and find derivations (formal proofs) of mathematical theorems. Finding derivations is a difficult task because the search space can be very large; an exhaustive search of every possible derivation is theoretically possible but computationally infeasible for many systems of interest in mathematics. Thus complicated heuristic functions are developed to attempt to find a derivation in less time than a blind search.

The related area of automated proof verification uses computer programs to check that human-created proofs are correct. Unlike complicated automated theorem provers, verification systems may be small enough that their correctness can be checked both by hand and through automated software verification. This validation of the proof verifier is needed to give confidence that any derivation labeled as "correct" is actually correct.

Some proof verifiers, such as Metamath, insist on having a complete derivation as input. Others, such as Mizar and Isabelle, take a well-formatted proof sketch (which may still be very long and detailed) and fill in the missing pieces by doing simple proof searches or applying known decision procedures: the resulting derivation is then verified by a small, core "kernel". Many such systems are primarily intended for interactive use by human mathematicians: these are known as proof assistants. They may also use formal logics that are stronger than first-order logic, such as type theory. Because a full derivation of any nontrivial result in a first-order deductive system will be extremely long for a human to write,^[6] results are often formalized as a series of lemmas, for which derivations can be constructed separately.

Automated theorem provers are also used to implement formal verification in computer science. In this setting, theorem provers are used to verify the correctness of programs and of hardware such as processors with respect to a formal specification. Because such analysis is time-consuming and thus expensive, it is usually reserved for projects in which a malfunction would have grave human or financial consequences.

Notes

- [1] The word *language* is sometimes used as a synonym for signature, but this can be confusing because "language" can also refer to the set of formulas.
- [2] More precisely, there is only one language of each variant of one-sorted first-order logic: with or without equality, with or without functions, with or without propositional variables,
- [3] Some authors who use the term "well-formed formula" use "formula" to mean any string of symbols from the alphabet. However, most authors in mathematical logic use "formula" to mean "well-formed formula" and have no term for non-well-formed formulas. In every context, it is only the well-formed formulas that are of interest.
- [4] The SMT-LIB Standard: Version 2.0, by Clark Barrett, Aaron Stump, and Cesare Tinelli. <http://goedel.cs.uiowa.edu/smtlib/>
- [5] Some authors only admit formulas with finitely many free variables in L_{κ_0} , and more generally only formulas with $< \lambda$ free variables in $L_{\kappa\lambda}$.
- [6] Avigad *et al.* (2007) discuss the process of formally verifying a proof of the prime number theorem. The formalized proof required approximately 30,000 lines of input to the Isabelle proof verifier.

References

- Andrews, Peter B. (2002); *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*, 2nd ed., Berlin: Kluwer Academic Publishers. Available from Springer.
- Avigad, Jeremy; Donnelly, Kevin; Gray, David; and Raff, Paul (2007); "A formally verified proof of the prime number theorem", *ACM Transactions on Computational Logic*, vol. 9 no. 1 doi: 10.1145/1297658.1297660 (<http://dx.doi.org/10.1145/1297658.1297660>)
- Barwise, Jon (1977); "An Introduction to First-Order Logic", in Barwise, Jon, ed. (1982). *Handbook of Mathematical Logic*. Studies in Logic and the Foundations of Mathematics. Amsterdam, NL: North-Holland.

ISBN 978-0-444-86388-1.

- Barwise, Jon; and Etchemendy, John (2000); *Language Proof and Logic*, Stanford, CA: CSLI Publications (Distributed by the University of Chicago Press)
- Bocheński, Józef Maria (2007); *A Précis of Mathematical Logic*, Dordrecht, NL: D. Reidel, translated from the French and German editions by Otto Bird
- Ferreirós, José (2001); *The Road to Modern Logic — An Interpretation* (<http://jstor.org/stable/2687794>), Bulletin of Symbolic Logic, Volume 7, Issue 4, 2001, pp. 441–484, DOI 10.2307/2687794, JStor ([http://links.jstor.org/sici?&sici=1079-8986\(200112\)7:4<441:TRTMLI>2.0.CO;2-O](http://links.jstor.org/sici?&sici=1079-8986(200112)7:4<441:TRTMLI>2.0.CO;2-O))
- Gamut, L. T. F. (1991); *Logic, Language, and Meaning, Volume 2: Intensional Logic and Logical Grammar*, Chicago, IL: University of Chicago Press, ISBN 0-226-28088-8
- Hilbert, David; and Ackermann, Wilhelm (1950); *Principles of Mathematical Logic*, Chelsea (English translation of *Grundzüge der theoretischen Logik*, 1928 German first edition)
- Hodges, Wilfrid (2001); "Classical Logic I: First Order Logic", in Goble, Lou (ed.); *The Blackwell Guide to Philosophical Logic*, Blackwell
- Ebbinghaus, Heinz-Dieter; Flum, Jörg; and Thomas, Wolfgang (1994); *Mathematical Logic*, Undergraduate Texts in Mathematics, Berlin, DE/New York, NY: Springer-Verlag, Second Edition, ISBN 978-0-387-94258-2
- Rautenberg, Wolfgang (2010), *A Concise Introduction to Mathematical Logic* (<http://www.springerlink.com/content/978-1-4419-1220-6/>) (3rd ed.), New York, NY: Springer Science+Business Media, doi: 10.1007/978-1-4419-1221-3 (<http://dx.doi.org/10.1007/978-1-4419-1221-3>), ISBN 978-1-4419-1220-6
- Tarski, Alfred and Givant, Steven (1987); *A Formalization of Set Theory without Variables*. Providence RI: American Mathematical Society.

External links

- 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
- Stanford Encyclopedia of Philosophy: Shapiro, Stewart; " Classical Logic (<http://plato.stanford.edu/entries/logic-classical/>)". Covers syntax, model theory, and metatheory for first-order logic in the natural deduction style.
- Magnus, P. D.; *forall x: an introduction to formal logic* (<http://www.fecundity.com/logic/>). Covers formal semantics and proof theory for first-order logic.
- Metamath (<http://us.metamath.org/index.html>): an ongoing online project to reconstruct mathematics as a huge first-order theory, using first-order logic and the axiomatic set theory ZFC. *Principia Mathematica* modernized.
- Podnieks, Karl; *Introduction to mathematical logic* (<http://www.ltn.lv/~podnieks/>)
- *Cambridge Mathematics Tripos Notes* (<http://john.fremlin.de/schoolwork/logic/index.html>) (typeset by John Fremlin). These notes cover part of a past Cambridge Mathematics Tripos course taught to undergraduates students (usually) within their third year. The course is entitled "Logic, Computation and Set Theory" and covers Ordinals and cardinals, Posets and Zorn's Lemma, Propositional logic, Predicate logic, Set theory and Consistency issues related to ZFC and other set theories.

Free Boolean algebra

In abstract algebra, a branch of mathematics, a **free Boolean algebra** is a Boolean algebra $\langle B, F \rangle$, such that the set B (called the *carrier*) has a subset whose elements are called generators. The generators satisfy the following properties:

- Each element of B that is not a generator can be expressed as a finite combination of generators, using the elements of F , which are operations;
- The generators are as "independent" as possible, in that any equation holding for finite terms formed from the generators using the operations in F , also holds for all elements of all possible Boolean algebras.

A simple example

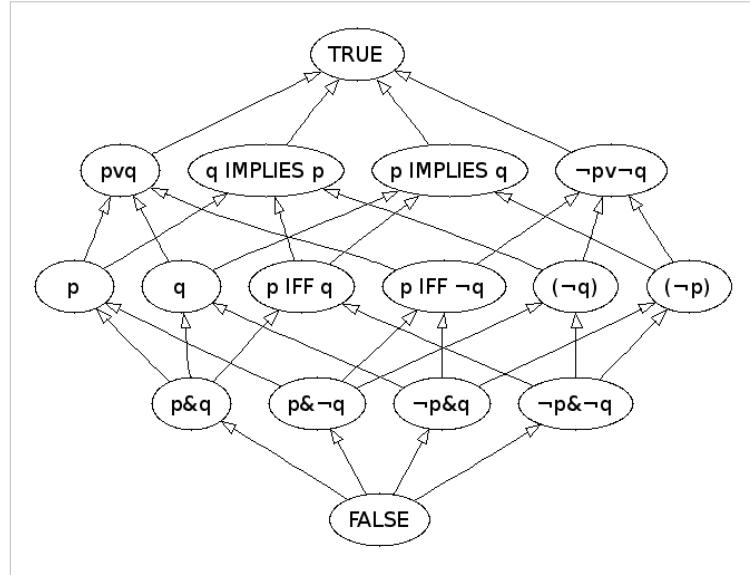
The generators of a free Boolean algebra can represent independent propositions. Consider, for example, the propositions "John is tall" and "Mary is rich". These generate a Boolean algebra with four atoms, namely:

- John is tall, and Mary is rich;
- John is tall, and Mary is not rich;
- John is not tall, and Mary is rich;
- John is not tall, and Mary is not rich.

Other elements of the Boolean algebra are then logical disjunctions of the atoms, such as "John is tall and Mary is not rich, or John is not tall and Mary is rich". In addition there is one more element, FALSE, which can be thought of as the empty disjunction; that is, the disjunction of no atoms.

This example yields a Boolean algebra with 16 elements; in general, for finite n , the free Boolean algebra with n generators has 2^n atoms, and therefore 2^{2^n} elements.

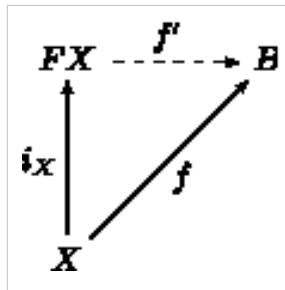
If there are infinitely many generators, a similar situation prevails except that now there are no atoms. Each element of the Boolean algebra is a combination of finitely many of the generating propositions, with two such elements deemed identical if they are logically equivalent.



Category-theoretic definition

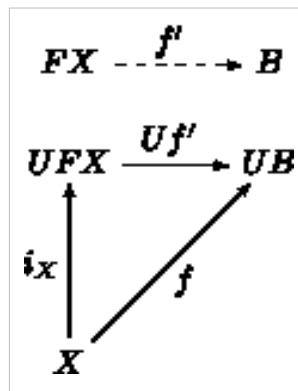
In the language of category theory, free Boolean algebras can be defined simply in terms of an adjunction between the category of sets and functions, **Set**, and the category of Boolean algebras and Boolean algebra homomorphisms, **BA**. In fact, this approach generalizes to any algebraic structure definable in the framework of universal algebra.

Above, we said that a free Boolean algebra is a Boolean algebra with a set of generators that behave a certain way; alternatively, one might start with a set and ask which algebra it generates. Every set X generates a free Boolean algebra FX defined as the algebra such that for every algebra B and function $f : X \rightarrow B$, there is a unique Boolean algebra homomorphism $f' : FX \rightarrow B$ that extends f . Diagrammatically,



where i_X is the inclusion, and the dashed arrow denotes uniqueness. The idea is that once one chooses where to send the elements of X , the laws for Boolean algebra homomorphisms determine where to send everything else in the free algebra FX . If FX contained elements inexpressible as combinations of elements of X , then f' wouldn't be unique, and if the elements of X weren't sufficiently independent, then f' wouldn't be well defined! It's easily shown that FX is unique (up to isomorphism), so this definition makes sense. It's also easily shown that a free Boolean algebra with generating set X , as defined originally, is isomorphic to FX , so the two definitions agree.

One shortcoming of the above definition is that the diagram doesn't capture that f' is a homomorphism; since it's a diagram in **Set** each arrow denotes a mere function. We can fix this by separating it into two diagrams, one in **BA** and one in **Set**. To relate the two, we introduce a functor $U : \mathbf{BA} \rightarrow \mathbf{Set}$ that "forgets" the algebraic structure, mapping algebras and homomorphisms to their underlying sets and functions.



If we interpret the top arrow as a diagram in **BA** and the bottom triangle as a diagram in **Set**, then this diagram properly expresses that every function $f : X \rightarrow B$ extends to a unique Boolean algebra homomorphism $f' : FX \rightarrow B$. The functor U can be thought of as a device to pull the homomorphism f' back into **Set** so it can be related to f .

The remarkable aspect of this is that the latter diagram is one of the various (equivalent) definitions of when two functors are adjoint. Our F easily extends to a functor **Set** \rightarrow **BA**, and our definition of X generating a free Boolean algebra FX is precisely that U has a left adjoint F .

Topological realization

The free Boolean algebra with κ generators, where κ is a finite or infinite cardinal number, may be realized as the collection of all clopen subsets of $\{0,1\}^\kappa$, given the product topology assuming that $\{0,1\}$ has the discrete topology. For each $\alpha < \kappa$, the α th generator is the set of all elements of $\{0,1\}^\kappa$ whose α th coordinate is 1. In particular, the free Boolean algebra with \aleph_0 generators is the collection of all clopen subsets of a Cantor space. Surprisingly, this collection is countable. In fact, while the free Boolean algebra with n generators, n finite, has cardinality 2^{2^n} , the free Boolean algebra with \aleph_0 generators has cardinality \aleph_0 .

For more on this topological approach to free Boolean algebra, see Stone's representation theorem for Boolean algebras.

References

- Steve Awodey (2006) *Category Theory* (Oxford Logic Guides 49). Oxford University Press.
- Paul Halmos and Steven Givant (1998) *Logic as Algebra*. Mathematical Association of America.
- Saunders Mac Lane (1998) *Categories for the Working Mathematician*. 2nd ed. (Graduate Texts in Mathematics 5). Springer-Verlag.
- Saunders Mac Lane (1999) *Algebra*, 3d. ed. American Mathematical Society. ISBN 0-8218-1646-2.
- Robert R. Stoll, 1963. *Set Theory and Logic*, chpt. 6.7. Dover reprint 1979.

Heyting algebra

In mathematics, a **Heyting algebra**, named after Arend Heyting, is a bounded lattice (with join and meet operations written \vee and \wedge and with least element 0 and greatest element 1) equipped with a binary operation $a \rightarrow b$ of *implication* such that $(a \rightarrow b) \wedge a \leq b$, and moreover $a \rightarrow b$ is the greatest such in the sense that if $c \wedge a \leq b$ then $c \leq a \rightarrow b$. From a logical standpoint, $A \rightarrow B$ is by this definition the weakest proposition for which modus ponens, the inference rule $A \rightarrow B, A \vdash B$, is sound. Equivalently a Heyting algebra is a residuated lattice whose monoid operation $a \bullet b$ is $a \wedge b$; yet another definition is as a posetal cartesian closed category with all finite sums. Like Boolean algebras, Heyting algebras form a variety axiomatizable with finitely many equations.

As lattices, Heyting algebras can be shown to be distributive. Every Boolean algebra is a Heyting algebra when $a \rightarrow b$ is defined as usual as $\neg a \vee b$, as is every complete distributive latticeWikipedia:Please clarify when $a \rightarrow b$ is taken to be the supremum of the set of all c for which $a \wedge c \leq b$. The open sets of a topological space form a complete distributive lattice and hence a Heyting algebra. In the finite case every nonempty distributive lattice, in particular every nonempty finite chain, is automatically bounded and complete and hence a Heyting algebra.

It follows from the definition that $1 \leq 0 \rightarrow a$, corresponding to the intuition that any proposition a is implied by a contradiction 0. Although the negation operation $\neg a$ is not part of the definition, it is definable as $a \rightarrow 0$. The definition implies that $a \wedge \neg a = 0$, making the intuitive content of $\neg a$ the proposition that to assume a would lead to a contradiction, from which any other proposition would then follow. It can further be shown that $a \leq \neg \neg a$, although the converse, $\neg \neg a \leq a$, is not true in general, that is, double negation does not hold in general in a Heyting algebra.

Heyting algebras generalize Boolean algebras in the sense that a Heyting algebra satisfying $a \vee \neg a = 1$ (excluded middle), equivalently $\neg \neg a = a$ (double negation), is a Boolean algebra. Those elements of a Heyting algebra of the form $\neg a$ comprise a Boolean lattice, but in general this is not a subalgebra of H (see below).

Heyting algebras serve as the algebraic models of propositional intuitionistic logic in the same way Boolean algebras model propositional classical logic. Complete Heyting algebras are a central object of study in pointless topology. The internal logic of an elementary topos is based on the Heyting algebra of subobjects of the terminal object 1 ordered by inclusion, equivalently the morphisms from 1 to the subobject classifier Ω .

Every Heyting algebra with exactly one coatom is subdirectly irreducible, whence every Heyting algebra can be made an SI by adjoining a new top. It follows that even among the finite Heyting algebras there exist infinitely many that are subdirectly irreducible, no two of which have the same equational theory. Hence no finite set of finite Heyting algebras can supply all the counterexamples to non-laws of Heyting algebra. This is in sharp contrast to Boolean algebras, whose only SI is the two-element one, which on its own therefore suffices for all counterexamples to non-laws of Boolean algebra, the basis for the simple truth table decision method. Nevertheless it is decidable whether an equation holds of all Heyting algebras.^[1]

Heyting algebras are less often called **pseudo-Boolean algebras**, or even **Brouwer lattices**, although the latter term may denote the dual definition, or have a slightly more general meaning.

Formal definition

A Heyting algebra H is a bounded lattice such that for all a and b in H there is a greatest element x of H such that

$$a \wedge x \leq b.$$

This element is the **relative pseudo-complement** of a with respect to b , and is denoted $a \rightarrow b$. We write 1 and 0 for the largest and the smallest element of H , respectively.

In any Heyting algebra, one defines the **pseudo-complement** $\neg a$ of any element a by setting $\neg a = (a \rightarrow 0)$. By definition, $a \wedge \neg a = 0$, and $\neg a$ is the largest element having this property. However, it is not in general true that $a \vee \neg a = 1$, thus \neg is only a pseudo-complement, not a true complement, as would be the case in a Boolean algebra.

A **complete Heyting algebra** is a Heyting algebra that is a complete lattice.

A **subalgebra** of a Heyting algebra H is a subset H_1 of H containing 0 and 1 and closed under the operations \wedge , \vee and \rightarrow . It follows that it is also closed under \neg . A subalgebra is made into a Heyting algebra by the induced operations.

Alternative definitions

Lattice-theoretic definitions

An equivalent definition of Heyting algebras can be given by considering the mappings:

$$\begin{cases} f_a: H \rightarrow H \\ f_a(x) = a \wedge x \end{cases}$$

for some fixed a in H . A bounded lattice H is a Heyting algebra if and only if every mapping f_a is the lower adjoint of a monotone Galois connection. In this case the respective upper adjoint g_a is given by $g_a(x) = a \rightarrow x$, where \rightarrow is defined as above.

Yet another definition is as a residuated lattice whose monoid operation is \wedge . The monoid unit must then be the top element 1. Commutativity of this monoid implies that the two residuals coincide as $a \rightarrow b$.

Bounded lattice with an implication operation

Given a bounded lattice A with largest and smallest elements 1 and 0, and a binary operation \rightarrow , these together form a Heyting algebra if and only if the following hold:

1. $a \rightarrow a = 1$
2. $a \wedge (a \rightarrow b) = a \wedge b$
3. $b \wedge (a \rightarrow b) = b$
4. $a \rightarrow (b \wedge c) = (a \rightarrow b) \wedge (a \rightarrow c)$

where 4 is the distributive law for \rightarrow .

Characterization using the axioms of intuitionistic logic

This characterization of Heyting algebras makes the proof of the basic facts concerning the relationship between intuitionist propositional calculus and Heyting algebras immediate. (For these facts, see the sections "Provable identities" and "Universal constructions".) One should think of the element 1 as meaning, intuitively, "provably true." Compare with the axioms at Intuitionistic logic#Axiomatization.

Given a set A with three binary operations \rightarrow , \wedge and \vee , and two distinguished elements 0 and 1, then A is a Heyting algebra for these operations (and the relation \leq defined by the condition that $a \leq b$ when $a \rightarrow b = 1$) if and only if the following conditions hold for any elements x , y and z of A :

1. If $x \rightarrow y = 1$ and $y \rightarrow x = 1$ then $x = y$,
2. If $1 \rightarrow y = 1$, then $y = 1$,
3. $x \rightarrow (y \rightarrow x) = 1$,
4. $(x \rightarrow (y \rightarrow z)) \rightarrow ((x \rightarrow y) \rightarrow (x \rightarrow z)) = 1$,
5. $x \wedge y \rightarrow x = 1$,
6. $x \wedge y \rightarrow y = 1$,
7. $x \rightarrow (y \rightarrow (x \wedge y)) = 1$,
8. $x \rightarrow x \vee y = 1$,
9. $y \rightarrow x \vee y = 1$,
10. $(x \rightarrow z) \rightarrow ((y \rightarrow z) \rightarrow (x \vee y \rightarrow z)) = 1$,
11. $0 \rightarrow x = 1$.

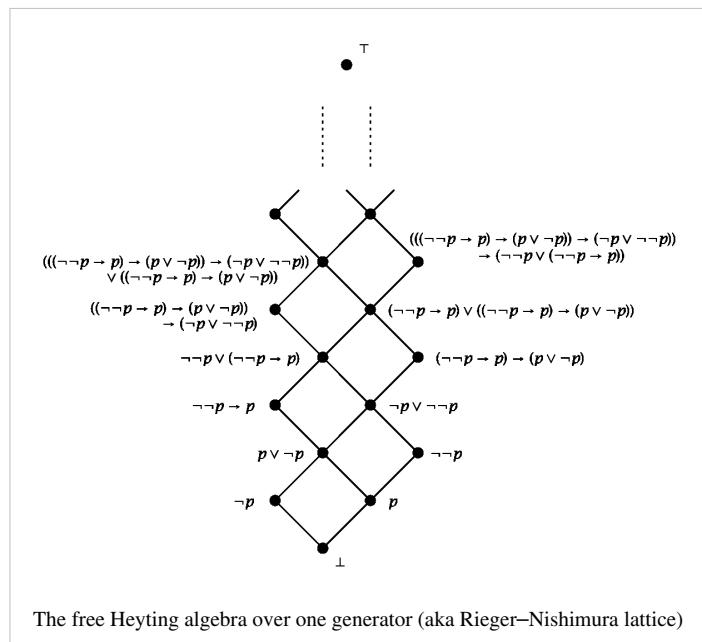
Finally, we define $\neg x$ to be $x \rightarrow 0$.

Condition 1 says that equivalent formulas should be identified. Condition 2 says that provably true formulas are closed under modus ponens. Conditions 3 and 4 are *then* conditions. Conditions 5, 6 and 7 are *and* conditions. Conditions 8, 9 and 10 are *or* conditions. Condition 11 is a *false* condition.

Of course, if a different set of axioms were chosen for logic, we could modify ours accordingly.

Examples

- Every Boolean algebra is a Heyting algebra, with $p \rightarrow q$ given by $\neg p \vee q$.
- Every totally ordered set that is a bounded lattice is also a Heyting algebra, where $p \rightarrow q$ is equal to q when $p > q$, and 1 otherwise.
- The simplest Heyting algebra that is not already a Boolean algebra is the totally ordered set $\{0, \frac{1}{2}, 1\}$ with \rightarrow defined as above, yielding the operations:



The free Heyting algebra over one generator (aka Rieger–Nishimura lattice)

$a \wedge b$				$a \vee b$				$a \rightarrow b$				$a \quad \neg a$	
b	0	$\frac{1}{2}$	1	b	0	$\frac{1}{2}$	1	b	0	$\frac{1}{2}$	1	a	$\neg a$
b	0	$\frac{1}{2}$	1	a				a				0	1
0	0	0	0	0	0	$\frac{1}{2}$	1	0	1	1	1	$\frac{1}{2}$	0
$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	1	$\frac{1}{2}$	0	1	1	1	0
1	0	$\frac{1}{2}$	1	1	1	1	1	1	0	$\frac{1}{2}$	1		

Notice that $\frac{1}{2} \vee \neg \frac{1}{2} = \frac{1}{2} \vee (\frac{1}{2} \rightarrow 0) = \frac{1}{2} \vee 0 = \frac{1}{2}$ falsifies the law of excluded middle.

- Every topology provides a complete Heyting algebra in the form of its open set lattice. In this case, the element $A \rightarrow B$ is the interior of the union of A^c and B , where A^c denotes the complement of the open set A . Not all complete Heyting algebras are of this form. These issues are studied in pointless topology, where complete Heyting algebras are also called **frames** or **locales**.
- Every interior algebra provides a Heyting algebra in the form of its lattice of open elements. Every Heyting algebra is of this form as a Heyting algebra can be completed to a Boolean algebra by taking its free Boolean extension as a bounded distributive lattice and then treating it as a generalized topology in this Boolean algebra.
- The Lindenbaum algebra of propositional intuitionistic logic is a Heyting algebra.
- The global elements of the subobject classifier Ω of an elementary topos form a Heyting algebra; it is the Heyting algebra of truth values of the intuitionistic higher-order logic induced by the topos.

Properties

General properties

The ordering \leq on a Heyting algebra H can be recovered from the operation \rightarrow as follows: for any elements a, b of H , $a \leq b$ if and only if $a \rightarrow b = 1$.

In contrast to some many-valued logics, Heyting algebras share the following property with Boolean algebras: if negation has a fixed point (i.e. $\neg a = a$ for some a), then the Heyting algebra is the trivial one-element Heyting algebra.

Provable identities

Given a formula $F(A_1, A_2, \dots, A_n)$ of propositional calculus (using, in addition to the variables, the connectives $\wedge, \vee, \neg, \rightarrow$, and the constants 0 and 1), it is a fact, proved early on in any study of Heyting algebras, that the following two conditions are equivalent:

1. The formula F is provably true in intuitionist propositional calculus.
2. The identity $F(a_1, a_2, \dots, a_n) = 1$ is true for any Heyting algebra H and any elements $a_1, a_2, \dots, a_n \in H$.

The metain implication $1 \Rightarrow 2$ is extremely useful and is the principal practical method for proving identities in Heyting algebras. In practice, one frequently uses the deduction theorem in such proofs.

Since for any a and b in a Heyting algebra H we have $a \leq b$ if and only if $a \rightarrow b = 1$, it follows from $1 \Rightarrow 2$ that whenever a formula $F \rightarrow G$ is provably true, we have $F(a_1, a_2, \dots, a_n) \leq G(a_1, a_2, \dots, a_n)$ for any Heyting algebra H , and any elements $a_1, a_2, \dots, a_n \in H$. (It follows from the deduction theorem that $F \rightarrow G$ is provable [from nothing] if and only if G is provable from F , that is, if G is a provable consequence of F .) In particular, if F and G are provably equivalent, then $F(a_1, a_2, \dots, a_n) = G(a_1, a_2, \dots, a_n)$, since \leq is an order relation.

$1 \Rightarrow 2$ can be proved by examining the logical axioms of the system of proof and verifying that their value is 1 in any Heyting algebra, and then verifying that the application of the rules of inference to expressions with value 1 in a Heyting algebra results in expressions with value 1. For example, let us choose the system of proof having modus ponens as its sole rule of inference, and whose axioms are the Hilbert-style ones given at Intuitionistic logic#Axiomatization. Then the facts to be verified follow immediately from the axiom-like definition of Heyting algebras given above.

$1 \Rightarrow 2$ also provides a method for proving that certain propositional formulas, though tautologies in classical logic, *cannot* be proved in intuitionist propositional logic. In order to prove that some formula $F(A_1, A_2, \dots, A_n)$ is not provable, it is enough to exhibit a Heyting algebra H and elements $a_1, a_2, \dots, a_n \in H$ such that $F(a_1, a_2, \dots, a_n) \neq 1$.

If one wishes to avoid mention of logic, then in practice it becomes necessary to prove as a lemma a version of the deduction theorem valid for Heyting algebras: for any elements a, b and c of a Heyting algebra H , we have $(a \wedge b) \rightarrow c = a \rightarrow (b \rightarrow c)$.

For more on the metain implication $2 \Rightarrow 1$, see the section "Universal constructions" below.

Distributivity

Heyting algebras are always distributive. Specifically, we always have the identities

1. $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
2. $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

The distributive law is sometimes stated as an axiom, but in fact it follows from the existence of relative pseudo-complements. The reason is that, being the lower adjoint of a Galois connection, \wedge preserves all existing suprema. Distributivity in turn is just the preservation of binary suprema by \wedge .

By a similar argument, the following infinite distributive law holds in any complete Heyting algebra:

$$x \wedge \bigvee Y = \bigvee \{x \wedge y \mid y \in Y\}$$

for any element x in H and any subset Y of H . Conversely, any complete lattice satisfying the above infinite distributive law is a complete Heyting algebra, with

$$a \rightarrow b = \bigvee \{c \mid a \wedge c \leq b\}$$

being its relative pseudo-complement operation.

Regular and complemented elements

An element x of a Heyting algebra H is called **regular** if either of the following equivalent conditions hold:

1. $x = \neg\neg x$.
2. $x = \neg y$ for some y in H .

The equivalence of these conditions can be restated simply as the identity $\neg\neg\neg x = \neg x$, valid for all x in H .

Elements x and y of a Heyting algebra H are called **complements** to each other if $x \wedge y = 0$ and $x \vee y = 1$. If it exists, any such y is unique and must in fact be equal to $\neg x$. We call an element x **complemented** if it admits a complement. It is true that if x is complemented, then so is $\neg x$, and then x and $\neg x$ are complements to each other. However, confusingly, even if x is not complemented, $\neg x$ may nonetheless have a complement (not equal to x). In any Heyting algebra, the elements 0 and 1 are complements to each other. For instance, it is possible that $\neg x$ is 0 for every x different from 0, and 1 if $x = 0$, in which case 0 and 1 are the only regular elements.

Any complemented element of a Heyting algebra is regular, though the converse is not true in general. In particular, 0 and 1 are always regular.

For any Heyting algebra H , the following conditions are equivalent:

1. H is a Boolean algebra;
2. every x in H is regular;^[2]
3. every x in H is complemented.^[3]

In this case, the element $a \rightarrow b$ is equal to $\neg a \vee b$.

The regular (resp. complemented) elements of any Heyting algebra H constitute a Boolean algebra H_{reg} (resp. H_{comp}), in which the operations \wedge , \neg and \rightarrow , as well as the constants 0 and 1, coincide with those of H . In the case of H_{comp} , the operation \vee is also the same, hence H_{comp} is a subalgebra of H . In general however, H_{reg} will not be a subalgebra of H , because its join operation \vee_{reg} may be differ from \vee . For $x, y \in H_{\text{reg}}$, we have $x \vee_{\text{reg}} y = \neg(\neg x \wedge \neg y)$. See below for necessary and sufficient conditions in order for \vee_{reg} to coincide with \vee .

The De Morgan laws in a Heyting algebra

One of the two De Morgan laws is satisfied in every Heyting algebra, namely

$$\forall x, y \in H : \quad \neg(x \vee y) = \neg x \wedge \neg y.$$

However, the other De Morgan law does not always hold. We have instead a weak de Morgan law:

$$\forall x, y \in H : \quad \neg(x \wedge y) = \neg\neg(\neg x \vee \neg y).$$

The following statements are equivalent for all Heyting algebras H :

1. H satisfies both De Morgan laws,
2. $\neg(x \wedge y) = \neg x \vee \neg y$ for all $x, y \in H$,
3. $\neg(x \wedge y) = \neg x \vee \neg y$ for all regular $x, y \in H$,
4. $\neg\neg(x \vee y) = \neg\neg x \vee \neg\neg y$ for all $x, y \in H$,
5. $\neg\neg(x \vee y) = x \vee y$ for all regular $x, y \in H$,
6. $\neg(\neg x \wedge \neg y) = x \vee y$ for all regular $x, y \in H$,
7. $\neg x \vee \neg\neg x = 1$ for all $x \in H$.

Condition 2 is the other De Morgan law. Condition 6 says that the join operation \vee_{reg} on the Boolean algebra H_{reg} of regular elements of H coincides with the operation \vee of H . Condition 7 states that every regular element is complemented, i.e., $H_{\text{reg}} = H_{\text{comp}}$.

We prove the equivalence. Clearly the metain implications $1 \Rightarrow 2$, $2 \Rightarrow 3$ and $4 \Rightarrow 5$ are trivial. Furthermore, $3 \Leftrightarrow 4$ and $5 \Leftrightarrow 6$ result simply from the first De Morgan law and the definition of regular elements. We show that $6 \Rightarrow 7$ by taking $\neg x$ and $\neg\neg x$ in place of x and y in 6 and using the identity $a \wedge \neg a = 0$. Notice that $2 \Rightarrow 1$ follows from the first De Morgan law, and $7 \Rightarrow 6$ results from the fact that the join operation \vee on the subalgebra H_{comp} is just the restriction of \vee to H_{comp} , taking into account the characterizations we have given of conditions 6 and 7. The metain implication $5 \Rightarrow 2$ is a trivial consequence of the weak De Morgan law, taking $\neg x$ and $\neg y$ in place of x and y in 5.

Heyting algebras satisfying the above properties are related to De Morgan logic in the same way Heyting algebras in general are related to intuitionist logic.

Heyting algebra morphisms

Definition

Given two Heyting algebras H_1 and H_2 and a mapping $f : H_1 \rightarrow H_2$, we say that f is a **morphism** of Heyting algebras if, for any elements x and y in H_1 , we have:

1. $f(0) = 0$,
2. $f(1) = 1$,
3. $f(x \wedge y) = f(x) \wedge f(y)$,
4. $f(x \vee y) = f(x) \vee f(y)$,
5. $f(x \rightarrow y) = f(x) \rightarrow f(y)$,
6. $f(\neg x) = \neg f(x)$.

We put condition 6 in brackets because it follows from the others, as $\neg x$ is just $x \rightarrow 0$, and one may or may not wish to consider \neg to be a basic operation.

It follows from conditions 3 and 5 (or 1 alone, or 2 alone) that f is an increasing function, that is, that $f(x) \leq f(y)$ whenever $x \leq y$.

Assume H_1 and H_2 are structures with operations $\rightarrow, \wedge, \vee$ (and possibly \neg) and constants 0 and 1, and f is a surjective mapping from H_1 to H_2 with properties 1 through 5 (or 1 through 6) above. Then if H_1 is a Heyting algebra, so too is H_2 . This follows from the characterization of Heyting algebras as bounded lattices (thought of as algebraic structures rather than partially ordered sets) with an operation \rightarrow satisfying certain identities.

Properties

The identity map $f(x) = x$ from any Heyting algebra to itself is a morphism, and the composite $g \circ f$ of any two morphisms f and g is a morphism. Hence Heyting algebras form a category.

Examples

Given a Heyting algebra H and any subalgebra H_1 , the inclusion mapping $i : H_1 \rightarrow H$ is a morphism.

For any Heyting algebra H , the map $x \mapsto \neg\neg x$ defines a morphism from H onto the Boolean algebra of its regular elements H_{reg} . This is *not* in general a morphism from H to itself, since the join operation of H_{reg} may be different from that of H .

Quotients

Let H be a Heyting algebra, and let $F \subseteq H$. We call F a **filter** on H if it satisfies the following properties:

1. $1 \in F$,
2. If $x, y \in F$ then $x \wedge y \in F$,
3. If $x \in F$, $y \in H$, and $x \leq y$ then $y \in F$.

The intersection of any set of filters on H is again a filter. Therefore, given any subset S of H there is a smallest filter containing S . We call it the filter **generated** by S . If S is empty, $F = \{1\}$. Otherwise, F is equal to the set of x in H such that there exist $y_1, y_2, \dots, y_n \in S$ with $y_1 \wedge y_2 \wedge \dots \wedge y_n \leq x$.

If H is a Heyting algebra and F is a filter on H , we define a relation \sim on H as follows: we write $x \sim y$ whenever $x \rightarrow y$ and $y \rightarrow x$ both belong to F . Then \sim is an equivalence relation; we write H/F for the quotient set. There is a unique Heyting algebra structure on H/F such that the canonical surjection $p_F : H \rightarrow H/F$ becomes a Heyting algebra morphism. We call the Heyting algebra H/F the **quotient** of H by F .

Let S be a subset of a Heyting algebra H and let F be the filter generated by S . Then H/F satisfies the following universal property:

- Given any morphism of Heyting algebras $f: H \rightarrow H'$ satisfying $f(y) = 1$ for every $y \in S$, f factors uniquely through the canonical surjection $p_F: H \rightarrow H/F$. That is, there is a unique morphism $f': H/F \rightarrow H'$ satisfying $f'p_F = f$. The morphism f' is said to be *induced* by f .

Let $f: H_1 \rightarrow H_2$ be a morphism of Heyting algebras. The **kernel** of f , written $\ker f$, is the set $f^{-1}[\{1\}]$. It is a filter on H_1 . (Care should be taken because this definition, if applied to a morphism of Boolean algebras, is dual to what would be called the kernel of the morphism viewed as a morphism of rings.) By the foregoing, f induces a morphism $f': H_1/(\ker f) \rightarrow H_2$. It is an isomorphism of $H_1/(\ker f)$ onto the subalgebra $f[H_1]$ of H_2 .

Universal constructions

Heyting algebra of propositional formulas in n variables up to intuitionist equivalence

The metain implication $2 \Rightarrow 1$ in the section "Provable identities" is proved by showing that the result of the following construction is itself a Heyting algebra:

- Consider the set L of propositional formulas in the variables A_1, A_2, \dots, A_n .
- Endow L with a preorder \leq by defining $F \leq G$ if G is an (intuitionist) logical consequence of F , that is, if G is provable from F . It is immediate that \leq is a preorder.
- Consider the equivalence relation $F \sim G$ induced by the preorder $F \leq G$. (It is defined by $F \sim G$ if and only if $F \leq G$ and $G \leq F$. In fact, \sim is the relation of (intuitionist) logical equivalence.)
- Let H_0 be the quotient set L/\sim . This will be the desired Heyting algebra.
- We write $[F]$ for the equivalence class of a formula F . Operations $\rightarrow, \wedge, \vee$ and \neg are defined in an obvious way on L . Verify that given formulas F and G , the equivalence classes $[F \rightarrow G]$, $[F \wedge G]$, $[F \vee G]$ and $[\neg F]$ depend only on $[F]$ and $[G]$. This defines operations $\rightarrow, \wedge, \vee$ and \neg on the quotient set $H_0 = L/\sim$. Further define 1 to be the class of provably true statements, and set $0 = [\perp]$.
- Verify that H_0 , together with these operations, is a Heyting algebra. We do this using the axiom-like definition of Heyting algebras. H_0 satisfies conditions THEN-1 through FALSE because all formulas of the given forms are axioms of intuitionist logic. MODUS-PONENS follows from the fact that if a formula $T \rightarrow F$ is provably true, where T is provably true, then F is provably true (by application of the rule of inference modus ponens). Finally, EQUIV results from the fact that if $F \rightarrow G$ and $G \rightarrow F$ are both provably true, then F and G are provable from each other (by application of the rule of inference modus ponens), hence $[F] = [G]$.

As always under the axiom-like definition of Heyting algebras, we define \leq on H_0 by the condition that $x \leq y$ if and only if $x \rightarrow y = 1$. Since, by the deduction theorem, a formula $F \rightarrow G$ is provably true if and only if G is provable from F , it follows that $[F] \leq [G]$ if and only if $F \leq G$. In other words, \leq is the order relation on L/\sim induced by the preorder \leq on L .

Free Heyting algebra on an arbitrary set of generators

In fact, the preceding construction can be carried out for any set of variables $\{A_i : i \in I\}$ (possibly infinite). One obtains in this way the *free* Heyting algebra on the variables $\{A_i\}$, which we will again denote by H_0 . It is free in the sense that given any Heyting algebra H given together with a family of its elements $\langle a_i : i \in I \rangle$, there is a unique morphism $f: H_0 \rightarrow H$ satisfying $f[A_i] = a_i$. The uniqueness of f is not difficult to see, and its existence results essentially from the metain implication $1 \Rightarrow 2$ of the section "Provable identities" above, in the form of its corollary that whenever F and G are provably equivalent formulas, $F(\langle a_i \rangle) = G(\langle a_i \rangle)$ for any family of elements $\langle a_i \rangle$ in H .

Heyting algebra of formulas equivalent with respect to a theory T

Given a set of formulas T in the variables $\{A_i\}$, viewed as axioms, the same construction could have been carried out with respect to a relation $F \leq G$ defined on L to mean that G is a provable consequence of F and the set of axioms T . Let us denote by H_T the Heyting algebra so obtained. Then H_T satisfies the same universal property as H_0 above, but with respect to Heyting algebras H and families of elements $\langle a_i \rangle$ satisfying the property that $J(\langle a_i \rangle) = 1$ for any axiom $J(\langle A_i \rangle)$ in T . (Let us note that H_T taken with the family of its elements $\langle [A_i] \rangle$, itself satisfies this property.) The existence and uniqueness of the morphism is proved the same way as for H_0 , except that one must modify the metain implication $1 \Rightarrow 2$ in "Provable identities" so that 1 reads "provably true from T ," and 2 reads "any elements a_1, a_2, \dots, a_n in H satisfying the formulas of T ."

The Heyting algebra H_T that we have just defined can be viewed as a quotient of the free Heyting algebra H_0 on the same set of variables, by applying the universal property of H_0 with respect to H_T , and the family of its elements $\langle [A_i] \rangle$.

Every Heyting algebra is isomorphic to one of the form H_T . To see this, let H be any Heyting algebra, and let $\langle a_i : i \in I \rangle$ be a family of elements generating H (for example, any surjective family). Now consider the set T of formulas $J(\langle A_i \rangle)$ in the variables $\langle A_i : i \in I \rangle$ such that $J(\langle a_i \rangle) = 1$. Then we obtain a morphism $f: H_T \rightarrow H$ by the universal property of H_T , which is clearly surjective. It is not difficult to show that f is injective.

Comparison to Lindenbaum algebras

The constructions we have just given play an entirely analogous role with respect to Heyting algebras to that of Lindenbaum algebras with respect to Boolean algebras. In fact, The Lindenbaum algebra B_T in the variables $\{A_i\}$ with respect to the axioms T is just our $H_{T \cup T_1}$, where T_1 is the set of all formulas of the form $\neg\neg F \rightarrow F$, since the additional axioms of T_1 are the only ones that need to be added in order to make all classical tautologies provable.

Heyting algebras as applied to intuitionistic logic

If one interprets the axioms of the intuitionistic propositional logic as terms of a Heyting algebra, then they will evaluate to the largest element, 1, in *any* Heyting algebra under any assignment of values to the formula's variables. For instance, $(P \wedge Q) \rightarrow P$ is, by definition of the pseudo-complement, the largest element x such that $P \wedge Q \wedge x \leq P$. This inequation is satisfied for any x , so the largest such x is 1.

Furthermore the rule of modus ponens allows us to derive the formula Q from the formulas P and $P \rightarrow Q$. But in any Heyting algebra, if P has the value 1, and $P \rightarrow Q$ has the value 1, then it means that $P \wedge 1 \leq Q$, and so $1 \wedge 1 \leq Q$; it can only be that Q has the value 1.

This means that if a formula is deducible from the laws of intuitionistic logic, being derived from its axioms by way of the rule of modus ponens, then it will always have the value 1 in all Heyting algebras under any assignment of values to the formula's variables. However one can construct a Heyting algebra in which the value of Peirce's law is not always 1. Consider the 3-element algebra $\{0, \frac{1}{2}, 1\}$ as given above. If we assign $\frac{1}{2}$ to P and 0 to Q , then the value of Peirce's law $((P \rightarrow Q) \rightarrow P) \rightarrow P$ is $\frac{1}{2}$. It follows that Peirce's law cannot be intuitionistically derived. See Curry–Howard isomorphism for the general context of what this implies in type theory.

The converse can be proven as well: if a formula always has the value 1, then it is deducible from the laws of intuitionistic logic, so the *intuitionistically valid* formulas are exactly those that always have a value of 1. This is similar to the notion that *classically valid* formulas are those formulas that have a value of 1 in the two-element Boolean algebra under any possible assignment of true and false to the formula's variables — that is, they are formulas which are tautologies in the usual truth-table sense. A Heyting algebra, from the logical standpoint, is then a generalization of the usual system of truth values, and its largest element 1 is analogous to 'true'. The usual two-valued logic system is a special case of a Heyting algebra, and the smallest non-trivial one, in which the only elements of the algebra are 1 (true) and 0 (false).

Decision problems

The problem of whether a given equation holds in every Heyting algebra was shown to be decidable by S. Kripke in 1965. The precise computational complexity of the problem was established by R. Statman in 1979, who showed it was PSPACE-complete^[4] and hence at least as hard as deciding equations of Boolean algebra (shown NP-complete in 1971 by S. Cook) and conjectured to be considerably harder. The elementary or first-order theory of Heyting algebras is undecidable.^[5] It remains open whether the universal Horn theory of Heyting algebras, or word problem, is decidable.^[6] Apropos of the word problem it is known that Heyting algebras are not locally finite (no Heyting algebra generated by a finite nonempty set is finite), in contrast to Boolean algebras which are locally finite and whose word problem is decidable. It is unknown whether free complete Heyting algebras exist except in the case of a single generator where the free Heyting algebra on one generator is trivially completable by adjoining a new top.

Notes

- [1] Kripke, S. A.: 1965, 'Semantical analysis of intuitionistic logic I'. In: J. N. Crossley and M. A. E. Dummett (eds.): *Formal Systems and Recursive Functions*. Amsterdam: North-Holland, pp. 92–130.
- [2] Rutherford (1965), Th.26.2 p.78.
- [3] Rutherford (1965), Th.26.1 p.78.
- [4] R. Statman. Intuitionistic propositional logic is polynomial-space complete. *Theoretical Comput. Sci.*, 9:67{72, 1979.
- [5] Andrzej Grzegorczyk (1951) "Undecidability of some topological theories," *Fundamenta Mathematicae* 38: 137-52.
- [6] Peter T. Johnstone, *Stone Spaces*, (1982) Cambridge University Press, Cambridge, ISBN 0-521-23893-5. (See paragraph 4.11)

References

- Rutherford, Daniel Edwin (1965). *Introduction to Lattice Theory*. Oliver and Boyd.
- F. Borceux, *Handbook of Categorical Algebra 3*, In *Encyclopedia of Mathematics and its Applications*, Vol. 53, Cambridge University Press, 1994.
- G. Gierz, K.H. Hoffmann, K. Keimel, J. D. Lawson, M. Mislove and D. S. Scott, *Continuous Lattices and Domains*, In *Encyclopedia of Mathematics and its Applications*, Vol. 93, Cambridge University Press, 2003.
- S. Ghilardi. *Free Heyting algebras as bi-Heyting algebras*, *Math. Rep. Acad. Sci. Canada XVI*, 6:240–244, 1992.

External links

- Heyting algebra (GFDLed)

Monadic Boolean algebra

In abstract algebra, a **monadic Boolean algebra** is an algebraic structure with signature

$$\langle A, \cdot, +, ', 0, 1, \exists \rangle$$

where $\langle A, \cdot, +, ', 0, 1 \rangle$ is a Boolean algebra.

The monadic/unary operator \exists denotes the existential quantifier, which satisfies the identities (using the received prefix notation for \exists):

- $\exists 0 = 0$
- $\exists x \geq x$
- $\exists(x + y) = \exists x + \exists y$
- $\exists x \exists y = \exists(x \exists y)$.

$\exists x$ is the *existential closure* of x . Dual to \exists is the unary operator \forall , the universal quantifier, defined as $\forall x := (\exists x')'$.

A monadic Boolean algebra has a dual formulation that takes \forall as primitive and \exists as defined, so that $\exists x := (\forall x')'$. Hence the dual algebra has signature $\langle A, \cdot, +, ', 0, 1, \forall \rangle$, with $\langle A, \cdot, +, ', 0, 1 \rangle$ a Boolean algebra, as before. Moreover, \forall satisfies the following dualized version of the above identities:

1. $\forall 1 = 1$
2. $\forall x \leq x$
3. $\forall(xy) = \forall x \forall y$
4. $\forall x + \forall y = \forall(x + \forall y)$.

$\forall x$ is the *universal closure* of x .

Discussion

Monadic Boolean algebras have an important connection to topology. If \forall is interpreted as the interior operator of topology, (1)-(3) above plus the axiom $\forall(\forall x) = \forall x$ make up the axioms for an interior algebra. But $\forall(\forall x) = \forall x$ can be proved from (1)-(4). Moreover, an alternative axiomatization of monadic Boolean algebras consists of the (reinterpreted) axioms for an interior algebra, plus $\forall(\forall x)' = (\forall x)'$ (Halmos 1962: 22). Hence monadic Boolean algebras are the semisimple interior/closure algebras such that:

- The universal (dually, existential) quantifier interprets the interior (closure) operator;
- All open (or closed) elements are also clopen.

A more concise axiomatization of monadic Boolean algebra is (1) and (2) above, plus $\forall(x \vee \forall y) = \forall x \vee \forall y$ (Halmos 1962: 21). This axiomatization obscures the connection to topology.

Monadic Boolean algebras form a variety. They are to monadic predicate logic what Boolean algebras are to propositional logic, and what polyadic algebras are to first-order logic. Paul Halmos discovered monadic Boolean algebras while working on polyadic algebras; Halmos (1962) reprints the relevant papers. Halmos and Givant (1998) includes an undergraduate treatment of monadic Boolean algebra.

Monadic Boolean algebras also have an important connection to modal logic. The modal logic S5, viewed as a theory in S4, is a model of monadic Boolean algebras in the same way that S4 is a model of interior algebra. Likewise, monadic Boolean algebras supply the algebraic semantics for S5. Hence **S5-algebra** is a synonym for monadic Boolean algebra.

References

- Paul Halmos, 1962. *Algebraic Logic*. New York: Chelsea.
- ----- and Steven Givant, 1998. *Logic as Algebra*. Mathematical Association of America.

Skew lattice

In abstract algebra, a **skew lattice** is an algebraic structure that is a non-commutative generalization of a lattice. While the term *skew lattice* can be used to refer to any non-commutative generalization of a lattice, over the past twenty years it has been used primarily as follows.

Definition

A **skew lattice** is a set S equipped with two associative, idempotent binary operations \wedge and \vee , called *meet* and *join*, that satisfy the following dual pair of absorption laws

$$x \wedge (x \vee y) = x = (y \vee x) \wedge x \text{ and } x \vee (x \wedge y) = x = (y \wedge x) \vee x.$$

Given that \vee and \wedge are associative and idempotent, these identities are equivalent to the dualities:

$$x \vee y = x \text{ iff } x \wedge y = y \text{ and } x \wedge y = x \text{ iff } x \vee y = y.^{[1]}$$

Historical background

For over 60 years, noncommutative variations of lattices have been studied with differing motivations. For some the motivation has been an interest in the conceptual boundaries of lattice theory; for others it was a search for noncommutative forms of logic and Boolean algebra; and for others it has been the behavior of idempotents in rings. A *noncommutative lattice*, generally speaking, is an algebra $(S; \wedge, \vee)$ where \wedge and \vee are associative, idempotent binary operations connected by absorption identities guaranteeing that \wedge in some way dualizes \vee . The precise identities chosen depends upon the underlying motivation, with differing choices producing distinct varieties of algebras. *Pascual Jordan*, motivated by questions in quantum logic, initiated a study of *noncommutative lattices* in his 1949 paper, *Über Nichtkommutative Verbande*,^[2] choosing the absorption identities

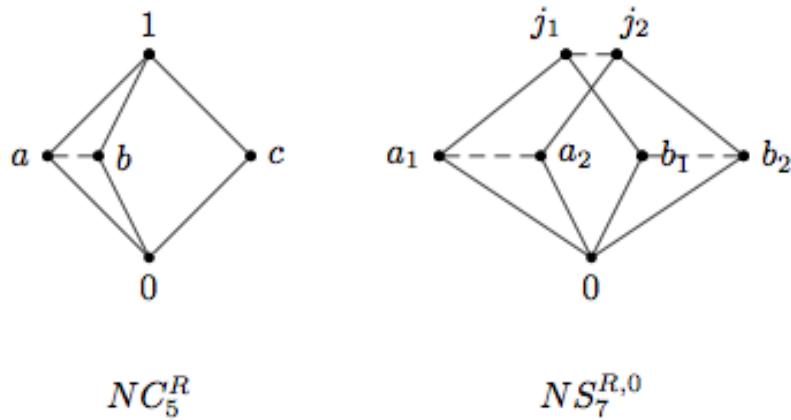
$$x \wedge (y \vee x) = x = (x \wedge y) \vee x.$$

He referred to those algebras satisfying them as *Schrägverbände*. By varying or augmenting these identities, Jordan and others obtained a number of varieties of noncommutative lattices. Beginning with Jonathan Leech's 1989 paper, *Skew lattices in rings*,^[3] skew lattices as defined above have been the primary objects of study. This was aided by previous results about bands. This was especially the case for many of the basic properties.

Basic properties

Natural partial order and natural quasiorder

In a skew lattice S , the natural partial order is defined by $y \leq x$ if $x \wedge y = y = y \wedge x$, or dually, $x \vee y = x = y \vee x$. The natural preorder on S is given by $y \preceq x$ if $y \wedge x \wedge y = y$ or dually $x \vee y \vee x = x$. While \leq and \preceq agree on lattices, \leq properly refines \preceq in the noncommutative case. The induced natural equivalence D is defined by $x D y$ if $x \preceq y \preceq x$, that is, $x \wedge y \wedge x = x$ and $y \wedge x \wedge y = y$ or dually, $x \vee y \vee x = x$ and $y \vee x \vee y = y$. The blocks of the partition S/D are lattice ordered by $A > B$ iff $a \in A$ and $b \in B$ exist such that $a > b$. This permits us to write Hasse diagrams of skew lattices such as the following pair:



E.g., in the diagram on the left above, that a and b are D related is expressed by the dashed segment. The slanted lines reveal the natural partial order between elements of the distinct D -classes. The elements $1, c$ and 0 form the singleton D -classes.

Rectangular Skew Lattices

Skew lattices consisting of a single D -class are called **rectangular**. They are characterized by the equivalent identities: $x \wedge y \wedge x = x$, $y \vee x \vee y = y$ and $x \vee y = y \wedge x$. Rectangular skew lattices are isomorphic to skew lattices having the following construction (and conversely): given nonempty sets L and R , on $L \times R$ define $(x, y) \vee (z, w) = (z, y)$ and $(x, y) \wedge (z, w) = (x, w)$. The D -class partition of a skew lattice S , as indicated in the above diagrams, is the unique partition of S into its maximal rectangular subalgebras. Moreover, D is a congruence with the induced quotient algebra S/D being the maximal lattice image of S , thus making every skew lattice S a lattice of rectangular subalgebras. This is the Clifford-McLean Theorem for skew lattices, first given for bands separately by Clifford and McLean. It is also known as *the First Decomposition Theorem for skew lattices*.

Right (left) handed skew lattices and the Kimura factorization

A skew lattice is right-handed if it satisfies the identity $x \wedge y \wedge x = y \wedge x$ or dually, $x \vee y \vee x = x \vee y$. These identities essentially assert that $x \wedge y = y$ and $x \vee y = x$ in each D -class. Every skew lattice S has a unique maximal right-handed image S/L where the congruence L is defined by xLy if both $x \wedge y = x$ and $y \wedge x = y$ (or dually, $x \vee y = y$ and $y \vee x = x$). Likewise a skew lattice is left-handed if $x \wedge y = x$ and $x \vee y = y$ in each D -class. Again the maximal left-handed image of a skew lattice S is the image S/R where the congruence R is defined in dual fashion to L . Many examples of skew lattices are either right or left-handed. In the lattice of congruences, $R \vee L = D$ and $R \cap L$ is the identity congruence Δ . The induced epimorphism $S \rightarrow S/D$ factors through both induced epimorphisms $S \rightarrow S/L$ and $S \rightarrow S/R$. Setting $T = S/D$, the homomorphism $k : S \rightarrow S/L \times S/R$ defined by $k(x) = (L_x, R_x)$, induces an isomorphism $k* : S \sim S/L \times_T S/R$. This is the Kimura factorization of S into a fibred product of its maximal right and left-handed images.

$$\begin{array}{ccc} S & \longrightarrow & S/R \\ \downarrow & & \downarrow \\ S/L & \longrightarrow & S/D \end{array}$$

Like the Clifford-McLean Theorem, Kimura factorization (or the *Second Decomposition Theorem for skew lattices*) was first given for regular bands (that satisfy the middle absorption identity, $xyxzx = xyzx$). Indeed both \wedge

and \vee are regular band operations. The above symbols D , R and L come, of course, from basic semigroup theory. For more details on the basic properties of a skew lattice please read [4][5][6][7][8][9] and [10].

Subvarieties of skew lattices

Skew lattices form a variety. Rectangular skew lattices, left-handed and right-handed skew lattices all form subvarieties that are central to the basic structure theory of skew lattices. Here are several more.

Symmetric Skew Lattices

A skew lattice S is symmetric if for any $x, y \in S$, $x \wedge y = y \wedge x$ iff $x \vee y = y \vee x$. Occurrences of commutation are thus unambiguous for such skew lattices, with subsets of pairwise commuting elements generating commutative subalgebras, i.e. sublattices. (This is not true for skew lattices in general.) Equational bases for this subvariety, first given by Spinks [11] are: $x \vee y \vee (x \wedge y) = (y \wedge x) \vee y \vee x$ and $x \wedge y \wedge (x \vee y) = (y \vee x) \wedge y \wedge x$. A **lattice section** of a skew lattice S is a sublattice T of S meeting each D -class of S at a single element. T is thus an internal copy of the lattice S/D with the composition $T \subset S \rightarrow S/D$ being an isomorphism. All symmetric skew lattices for which $|S/D| \leq \aleph_0$, admit a lattice section. Symmetric or not, having a lattice section T guarantees that S also has internal copies of S/L and S/R given respectively by $T[R] = \bigcup_{t \in T} R_t$ and $T[L] = \bigcup_{t \in T} L_t$, where R_t and L_t are the R and L congruence classes of t in T . Thus $T[R] \subset S \rightarrow S/L$ and $T[L] \subset S \rightarrow S/R$ are isomorphisms (See). This leads to a commuting diagram of embedding dualiz

$$\begin{array}{ccc} T[\mathcal{L}] & \xleftarrow{\quad} & T \\ \downarrow & & \downarrow \\ S & \xleftarrow{\quad} & T/\mathcal{R} \end{array}$$

Cancellative Skew Lattices

A skew lattice is cancellative if $x \vee y = x \vee z$ and $x \wedge y = x \wedge z$ implies $y = z$ and likewise $x \vee z = y \vee z$ and $x \wedge z = y \wedge z$ implies $x = y$. Cancellative skew lattices are symmetric and can be shown to form a variety. Unlike lattices, they need not be distributive, and conversely.

Distributive Skew Lattices

Distributive skew lattices are determined by the identities: $x \wedge (y \vee z) \wedge x = (x \wedge y \wedge x) \vee (x \wedge z \wedge x)$ (D1) $x \vee (y \wedge z) \vee x = (x \vee y \vee x) \wedge (x \vee z \vee x)$. (D'1) Unlike lattices, (D1) and (D'1) are not equivalent in general for skew lattices, but they are for symmetric skew lattices. (See,, [12]) The condition (D1) can be strengthened to $x \wedge (y \vee z) \wedge w = (x \wedge y \wedge w) \vee (x \wedge z \wedge w)$ (D2) in which case (D'1) is a consequence. A skew lattice S satisfies both (D2) and its dual, $x \vee (y \wedge z) \vee w = (x \vee y \vee w) \wedge (x \vee z \vee w)$, if and only if it factors as the product of a distributive lattice and a rectangular skew lattice. In this latter case (D2) can be strengthened to $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ and $(y \vee z) \wedge w = (y \wedge w) \vee (z \wedge w)$. (D3) On its own, (D3) is equivalent to (D2) when symmetry is added. (See.) We thus have six subvarieties of skew lattices determined respectively by (D1), (D2), (D3) and their duals.

Normal Skew Lattices

As seen above, \wedge and \vee satisfy the identity $xyxzx = xyzx$. Bands satisfying the stronger identity, $xyzx = xzyx$, are called normal. A skew lattice is normal skew if it satisfies

$$x \wedge y \wedge z \wedge x = x \wedge z \wedge y \wedge x. (N)$$

For each element a in a normal skew lattice S , the set $a \wedge S \wedge a$ defined by $\{a \wedge x \wedge a | x \in S\}$ or equivalently $\{x \in S | x \leq a\}$ is a sublattice of S , and conversely. (Thus normal skew lattices have also been called local lattices.) When both \wedge and \vee are normal, S splits isomorphically into a product $T \times D$ of a lattice T and a rectangular skew lattice D , and conversely. Thus both normal skew lattices and split skew lattices form varieties. Returning to distribution, $(D2) = (D1) + (N)$ so that $(D2)$ characterizes the variety of distributive, normal skew lattices, and $(D3)$ characterizes the variety of symmetric, distributive, normal skew lattices.

Categorical Skew Lattices

A skew lattice is categorical if nonempty composites of coset bijections are coset bijections. Categorical skew lattices form a variety. Skew lattices in rings and normal skew lattices are examples of algebras on this variety.^[4] Let $a > b > c$ with $a \in A, b \in B$ and $c \in C$, φ be the coset bijection from A to B taking a to b , ψ be the coset bijection from B to C taking b to c and finally χ be the coset bijection from A to C taking a to c . A skew lattice S is categorical if one always has the equality $\psi \circ \varphi = \chi$, ie., if the composite partial bijection $\psi \circ \varphi$ if nonempty is a coset bijection from a C -coset of A to an A -coset of C . That is $(A \wedge b \wedge A) \cap (C \vee b \vee C) = (C \vee a \vee C) \wedge b \wedge (C \vee a \vee C) = (A \wedge c \wedge A) \vee b \vee (A \wedge c \wedge A)$. All distributive skew lattices are categorical. Though symmetric skew lattices might not be. In a sense they reveal the independence between the properties of symmetry and distributivity.

For more details on these and other subvarieties of skew lattices please read [1][13] and.

Skew Boolean algebras

A zero element in a skew lattice S is an element 0 of S such that for all $x \in S$, $0 \wedge x = 0 = x \wedge 0$ or, dually, $0 \vee x = x = x \vee 0$. (0)

A Boolean skew lattice is a symmetric, distributive normal skew lattice with 0 , $(S; \vee, \wedge, 0)$, such that $a \wedge S \wedge a$ is a Boolean lattice for each $a \in S$. Given such skew lattice S , a difference operator \setminus is defined on by $x \setminus y = x - x \wedge y \wedge x$ where the latter is evaluated in the Boolean lattice $x \wedge S \wedge x$. In the presence of $(D3)$ and (0) , \setminus is characterized by the identities: $y \wedge x / y = 0 = x / y \wedge y$ and $(x \wedge y \wedge x) \vee x / y = x = x / y \vee (x \wedge y \wedge x)$. (SB) One thus has a variety of skew Boolean algebras $(S; \vee, \wedge, /, 0)$ characterized by identities $(D3)$, (0) and (SB). A primitive skew Boolean algebra consists of 0 and a single non-0 D-class. Thus it is the result of adjoining a 0 to a rectangular skew lattice D via (0) with $x / y = x$, if $y = 0$ and 0 otherwise. Every skew Boolean algebra is a subdirect product of primitive algebras. Skew Boolean algebras play an important role in the study of discriminator varieties and other generalizations in universal algebra of Boolean behavior. For more details on skew Boolean algebras see [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24].

Skew lattices in rings

Let A be a Ring and let $E(A)$ denote the set of all Idempotents in A . For all $x, y \in A$ set $x \wedge y = xy$ and $x \vee y = x + y - xy$.

Clearly \wedge but also \vee is associative. If a subset $S \subseteq E(A)$ is closed under \wedge and \vee , then (S, \wedge, \vee) is a distributive, cancellative skew lattice. To find such skew lattices in $E(A)$ one looks at bands in $E(A)$, especially the ones that are maximal with respect to some constraint. In fact, every multiplicative band in (A) that is maximal with respect to being right regular ($=$) is also closed under \vee and so forms a right-handed skew lattice. In general, every right regular band in $E(A)$ generates a right-handed skew lattice in $E(A)$. Dual remarks also hold for left regular bands (bands satisfying the identity $xyx = xy$) in $E(A)$. Maximal regular bands need not to be closed under \vee as defined; counterexamples are easily found using multiplicative rectangular bands. These cases are closed, however, under the cubic variant of \vee defined by $x \nabla y = x + y + yx - xyx - yxy$ since in these cases $x \nabla y$ reduces to yx to give the dual rectangular band. By replacing the condition of regularity by normality

$(xyzw = xzyw)$, every maximal normal multiplicative band S in $E(A)$ is also closed under ∇ with $(S; \wedge, \vee, /, 0)$, where S is closed under multiplication. When $E(A)$ itself is closed under multiplication, then it is a normal band and thus forms a Boolean skew lattice. In fact, any skew Boolean algebra can be embedded into such an algebra. (See.^[25]) When A has a multiplicative identity 1 , the condition that $E(A)$ is multiplicatively closed is well-known to imply that $E(A)$ forms a Boolean algebra. Skew lattices in rings continue to be a good source of examples and motivation. For more details read.^{[22][26][27][28][29]}

Primitive skew lattices

Skew lattices consisting of exactly two D-classes are called primitive skew lattices. Given such a skew lattice S with D -classes $A > B$ in S/D , then for any $a \in A$ and $b \in B$, the subsets

$$A \wedge b \wedge A = \{ u \wedge b \wedge u : u \in A \} \subseteq B \text{ and } B \vee a \vee B = \{ v \vee a \vee v : v \in B \} \subseteq A$$

are called, respectively, *cosets of A in B* and *cosets of B in A*. These cosets partition B and A with $b \in A \wedge b \wedge A$ and $a \in B \vee a \vee B$. Cosets are always rectangular subalgebras in their D -classes. What is more, the partial order \geq induces a coset bijection $\varphi : B \vee a \vee B \rightarrow A \wedge b \wedge A$ defined by:

$$\phi(x) = y \text{ iff } x > y, \text{ for } x \in B \vee a \vee B \text{ and } y \in A \wedge b \wedge A.$$

Collectively, coset bijections describe \geq between the subsets A and B . They also determine \vee and \wedge for pairs of elements from distinct D -classes. Indeed, given $a \in A$ and $b \in B$, let φ be the coset bijection between the cosets $B \vee a \vee B$ in A and $A \wedge b \wedge A$ in B . Then:

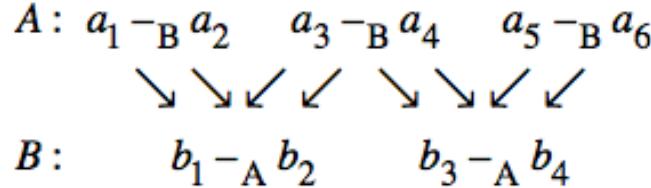
$$a \vee b = a \vee \varphi^{-1}(b), b \vee a = \varphi^{-1}(b) \vee a \text{ and } a \wedge b = \varphi(a) \wedge b, b \wedge a = b \wedge \varphi(a).$$

In general, given $a, c \in A$ and $b, d \in B$ with $a > b$ and $c > d$, then a, c belong to a common B -coset in A and b, d belong to a common A -coset in B if and only if $a > b // c > d$. Thus each coset bijection is, in some sense, a maximal collection of mutually parallel pairs $a > b$.

Every primitive skew lattice S factors as the fibred product of its maximal left and right-handed primitive images $S/R \times_2 S/L$. Right-handed primitive skew lattices are constructed as follows. Let $A = \cup_i A_i$ and $B = \cup_j B_j$ be partitions of disjoint nonempty sets A and B , where all A_i and B_j share a common size. For each pair i, j pick a fixed bijection $\varphi_{i,j}$ from A_i onto B_j . On A and B separately set $x \wedge y = y$ and $x \vee y = x$; but given $a \in A$ and $b \in B$, set

$$a \vee b = a, b \vee a = a', a \wedge b = b \text{ and } b \wedge a = b'$$

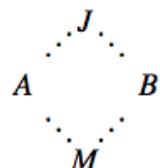
where $\varphi_{i,j}(a') = b$ and $\varphi_{i,j}(a) = b'$ with a' belonging to the cell A_i of a and b' belonging to the cell B_j of b . The various $\varphi_{i,j}$ are the coset bijections. This is illustrated in the following partial Hasse diagram where $|A_i| = |B_j| = 2$ and the arrows indicate the $\varphi_{i,j}$ -outputs and \geq from A and B .



One constructs left-handed primitive skew lattices in dual fashion. All right [left] handed primitive skew lattices can be constructed in this fashion. (See Section 1.)

The coset structure of skew lattices

A nonrectangular skew lattice S is covered by its maximal primitive skew lattices: given comparable D -classes $A > B$ in S/D , $A \cup B$ forms a maximal primitive subalgebra of S and every D -class in S lies in such a subalgebra. The coset structures on these primitive subalgebras combine to determine the outcomes $x \vee y$ and $x \wedge y$ at least when x and y are comparable under \preceq . It turns out that $x \vee y$ and $x \wedge y$ are determined in general by cosets and their bijections, although in a slightly less direct manner than the \preceq -comparable case. In particular, given two incomparable D-classes A and B with join D-class J and meet D-class M in S/D , interesting connections arise between the two coset decompositions of J (or M) with respect to A and B . (See Section 3.)



Thus a skew lattice may be viewed as a coset atlas of rectangular skew lattices placed on the vertices of a lattice and coset bijections between them, the latter seen as partial isomorphisms between the rectangular algebras with each coset bijection determining a corresponding pair of cosets. This perspective gives, in essence, the Hasse diagram of the skew lattice, which is easily drawn in cases of relatively small order. (See the diagrams in Section 3 above.) Given a chain of D-classes $A > B > C$ in S/D , one has three sets of coset bijections: from A to B , from B to C and from A to C . In general, given coset bijections $\varphi : A \rightarrow B$ and $\psi : B \rightarrow C$, the composition of partial bijections $\psi\varphi$ could be empty. If it is not, then a unique coset bijection $\chi : A \rightarrow C$ exists such that $\psi\varphi \subseteq \chi$. (Again, χ is a bijection between a pair of cosets in A and C .) This inclusion can be strict. It is always an equality (given $\psi\varphi \neq \emptyset$) on a given skew lattice S precisely when S is categorical. In this case, by including the identity maps on each rectangular D-class and adjoining empty bijections between properly comparable D-classes, one has a category of rectangular algebras and coset bijections between them. The simple examples in Section 3 are categorical.

References

- [1] Leech, J, Skew lattices in rings, *Algebra Universalis*, 26(1989), 48-72.
- [2] Jordan, P. Über Nichtkommutative Verbände, *Arch. Math.* 2 (1949), 56–59.
- [3] Leech, J. Skew lattices in rings, *Algebra Universalis*, 26(1989), 48-72
- [4] Leech, J. Recent developments in the theory of skew lattices, *Semigroup Forum*, 52(1996), 7-24.
- [5] Leech, J, Magic squares, finite planes and simple quasilattices, *Ars Combinatoria* 77(2005), 75-96.
- [6] Leech, J. The geometry of skew lattices, *Semigroup Forum*, 52(1993), 7-24.
- [7] Leech, J. Normal skew lattices, *Semigroup Forum*, 44(1992), 1-8.
- [8] Cvetko-Vah, K, Internal decompositions of skew lattices, *Communications in Algebra*, 35 (2007), 243-247
- [9] Cvetko-Vah, K, A new proof of Spinks' Theorem, *Semigroup Forum* 73 (2006), 267-272.
- [10] Laslo, G and Leech, J, Green's relations on noncommutative lattices, *Acta Sci. Math. (Szeged)*, 68 (2002), 501-533.
- [11] Spinks, M, Automated deduction in non-commutative lattice theory, *Tech. Report 3/98*, Monash U, GSCIT, 1998
- [12] Spinks, M, Automated deduction in non-commutative lattice theory, *Tech. Report 3/98*, Monash University, Gippsland School of Computing and Information Technology, June 1998
- [13] Cvetko-Vah, Karin ; Kinyon, M. ; Leech, J. ; Spinks, M. Skew Lattices with cancellation. Pre-Print. *Journal of Algebra and Its Applications*, 2008
- [14] Bignall, R. J., Quasiprimal Varieties and Components of Universal Algebras, Dissertation, The Flinders University of South Australia, 1976.
- [15] Bignall, R J, A non-commutative multiple-valued logic, Proc. 21st International Symposium on Multiple-valued Logic, 1991, IEEE Computer Soc. Press, 49-54.
- [16] Bignall, R J and J Leech, Skew Boolean algebras and discriminator varieties, *Algebra Universalis*, 33(1995), 387-398.
- [17] Bignall, R J and M Spinks, Propositional skew Boolean logic, Proc. 26th International Symposium on Multiple-valued Logic, 1996, IEEE Computer Soc. Press, 43-48.

- [18] Bignall, R J and M Spinks, Implicative BCS-algebra subreducts of skew Boolean algebras, *Scientiae Mathematicae Japonicae*, 58 (2003), 629-638.
- [19] Bignall, R J and M Spinks, On binary discriminator varieties (I): Implicative BCS-algebras, *International Journal of Algebra and Computation*, to appear.
- [20] Cornish, W H, Boolean skew algebras, *Acta Math. Acad. Sci. Hung.*, 36 (1980), 281-291.
- [21] Leech, J, Skew Boolean algebras, *Algebra Universalis*, 27(1990), 497-506.
- [22] Leech and Spinks, Skew Boolean algebras generated from generalized Boolean algebras, *Algebra Universalis* 58 (2008), 287-302.
- [23] Spinks, M, Contributions to the Theory of Pre-BCK Algebras, Monash University Dissertation, 2002.
- [24] Spinks, M and R Veroff, Axiomatizing the skew Boolean propositional calculus, *J.Automated Reasoning*, 37 (2006), 3-20.
- [25] Cvetko-Vah, K, Skew lattices in matrix rings, *Algebra Universalis* 53 (2005), 471-479.
- [26] Cvetko-Vah, K, Pure skew lattices in rings, *Semigroup Forum* 68 (2004), 268-279.
- [27] Cvetko-Vah, K, Pure ∇ -bands, *Semigroup Forum* 71 (2005), 93-101.
- [28] Cvetko-Vah, K, Skew lattices in rings, Dissertation, University of Ljubljana, 2005.
- [29] Cvetko-Vah, K and J Leech, Associativity of the ∇ -operation on bands in rings, *Semigroup Forum* 76 (2008), 32-50

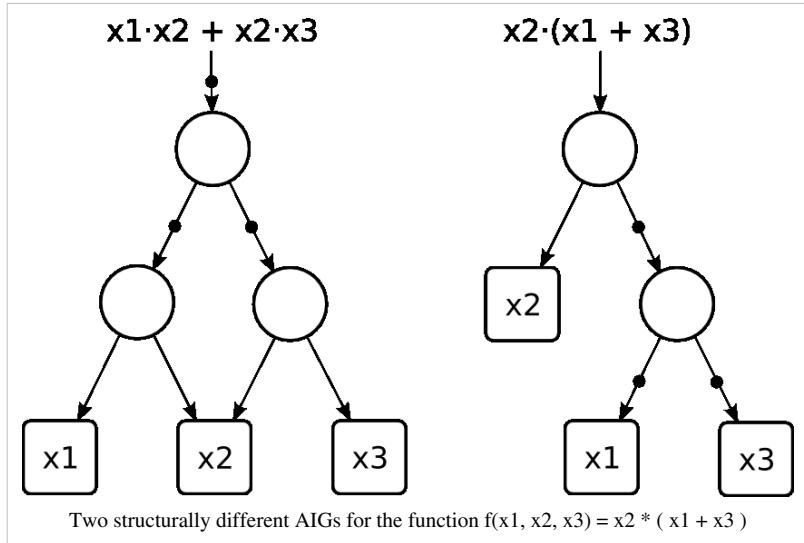
Technical applications

And-inverter graph

An **and-inverter graph (AIG)** is a directed, acyclic graph that represents a structural implementation of the logical functionality of a circuit or network. An AIG consists of two-input nodes representing logical conjunction, terminal nodes labeled with variable names, and edges optionally containing markers indicating logical negation. This representation of a logic function is rarely structurally efficient for large circuits, but is an efficient representation for manipulation of boolean functions. Typically, the abstract graph is represented as a data structure in software.

Conversion from the network of logic gates to AIGs is fast and scalable. It only requires that every gate be expressed in terms of AND gates and inverters. This conversion does not lead to unpredictable increase in memory use and runtime. This makes the AIG an efficient representation in comparison with either the binary decision diagram (BDD) or the "sum-of-products" ($\Sigma\pi$) form, that is, the canonical form in Boolean algebra known as the disjunctive normal form (DNF). The BDD and DNF may also be viewed as circuits, but they involve

formal constraints that deprive them of scalability. For example, $\Sigma\pi$ s are circuits with at most two levels while BDDs are canonical, that is, they require that input variables be evaluated in the same order on all paths.



Circuits composed of simple gates, including AIGs, are an "ancient" research topic. The interest in AIGs started in the late 1950s and continued in the 1970s when various local transformations have been developed. These transformations were implemented in several logic synthesis and verification systems, such as Darringer et al. and Smith et al., which reduce circuits to improve area and delay during synthesis, or to speed up formal equivalence checking. Several important techniques were discovered early at IBM, such as combining and reusing multi-input logic expressions and subexpressions, now known as structural hashing.

Recently there has been a renewed interest in AIGs as a functional representation for a variety of tasks in synthesis and verification. That is because representations popular in the 1990s (such as BDDs) have reached their limits of scalability in many of their applications. Another important development was the recent emergence of much more efficient boolean satisfiability (SAT) solvers. When coupled with AIGs as the circuit representation, they lead to remarkable speedups in solving a wide variety of boolean problems.

AIGs found successful use in diverse EDA applications. A well-tuned combination of AIGs and boolean satisfiability made an impact on formal verification, including both model checking and equivalence checking. Another recent work shows that efficient circuit compression techniques can be developed using AIGs. There is a growing understanding that logic and physical synthesis problems can be solved using AIGs simulation and boolean satisfiability compute functional properties (such as symmetries) and node flexibilities (such as don't-cares, resubstitutions, and SPFDS). This work shows that AIGs are a promising *unifying* representation, which can bridge

logic synthesis, technology mapping, physical synthesis, and formal verification. This is, to a large extent, due to the simple and uniform structure of AIGs, which allow rewriting, simulation, mapping, placement, and verification to share the same data structure.

In addition to combinational logic, AIGs have also been applied to sequential logic and sequential transformations. Specifically, the method of structural hashing was extended to work for AIGs with memory elements (such as D-type flip-flops with an initial state, which, in general, can be unknown) resulting in a data structure that is specifically tailored for applications related to retiming.

Ongoing research includes implementing a modern logic synthesis system completely based on AIGs. The prototype called ABC^[1] features an AIG package, several AIG-based synthesis and equivalence-checking techniques, as well as an experimental implementation of sequential synthesis. One such technique combines technology mapping and retiming in a single optimization step. These optimizations can be implemented using networks composed of arbitrary gates, but the use of AIGs makes them more scalable and easier to implement.

Implementations

- Logic Synthesis and Verification System ABC^[1]
- A set of utilities for AIGs AIGER^[2]
- OpenAccess Gear^[3]

References

- [1] <http://www.eecs.berkeley.edu/~alanmi/abc/>
- [2] <http://fmv.jku.at/aiger/index.html>
- [3] http://www.si2.org/openeda.si2.org/help/group_ld.php?group=73

Boolean analysis

Boolean analysis was introduced by Flament (1976). The goal of a Boolean analysis is to detect deterministic dependencies between the items of a questionnaire or similar data-structures in observed response patterns. These deterministic dependencies have the form of logical formulas connecting the items. Assume, for example, that a questionnaire contains items i , j , and k . Examples of such deterministic dependencies are then $i \rightarrow j$, $i \wedge j \rightarrow k$, and $i \vee j \rightarrow k$.

Since the basic work of Flament (1976) a number of different methods for Boolean analysis have been developed. See, for example, Buggenhaut and Degreef (1987), Duquenne (1987), item tree analysis Leeuwe (1974), Schrepp (1999), or Theuns (1998). These methods share the goal to derive deterministic dependencies between the items of a questionnaire from data, but differ in the algorithms to reach this goal.

Boolean analysis is an explorative method to detect deterministic dependencies between items. The detected dependencies must be confirmed in subsequent research. Methods of Boolean analysis do not assume that the detected dependencies describe the data completely. There may be other probabilistic dependencies as well. Thus, a Boolean analysis tries to detect interesting deterministic structures in the data, but has not the goal to uncover all structural aspects in the data set. Therefore, it makes sense to use other methods, like for example latent class analysis, together with a Boolean analysis.

Application areas

The investigation of deterministic dependencies has some tradition in educational psychology. The items represent in this area usually skills or cognitive abilities of subjects. Bart and Airasian (1974) use Boolean analysis to establish logical implications on a set of Piagetian tasks. Other examples in this tradition are the learning hierarchies of Gagné (1968) or the theory of structural learning of Scandura (1971).

There are several attempts to use boolean analysis, especially item tree analysis to construct knowledge spaces from data. Examples can be found in Held and Korossy (1998), or Schrepp (2002).

Methods of Boolean analysis are used in a number of social science studies to get insight into the structure of dichotomous data. Bart and Krus (1973) use, for example, Boolean analysis to establish a hierarchical order on items that describe socially unaccepted behavior. Janssens (1999) used a method of Boolean analysis to investigate the integration process of minorities into the value system of the dominant culture. Romme (1995a) introduced Boolean comparative analysis to the management sciences, and applied it in a study of self-organizing processes in management teams (Romme 1995b).

Relations to other areas

Boolean analysis has some relations to other research areas. There is a close connection between Boolean analysis and knowledge spaces. The theory of knowledge spaces provides a theoretical framework for the formal description of human knowledge. A knowledge domain is in this approach represented by a set Q of problems. The knowledge of a subject in the domain is then described by the subset of problems from Q he or she is able to solve. This set is called the *knowledge state* of the subject. Because of dependencies between the items (for example, if solving item j implies solving item i) not all elements of the power set of Q will, in general, be possible knowledge states. The set of all possible knowledge states is called the *knowledge structure*. Methods of Boolean analysis can be used to construct a knowledge structure from data (for example, Theuns, 1998 or Schrepp, 1999). The main difference between both research areas is that Boolean analysis concentrates on the extraction of structures from data while knowledge space theory focus on the structural properties of the relation between a knowledge structure and the logical formulas which describe it.

Closely related to knowledge space theory is formal concept analysis (Ganter and Wille, 1996). Similar to knowledge space theory this approach concentrates on the formal description and visualization of existing dependencies. In contrast Boolean analysis offers a way to construct such dependencies from data.

Another related field is data mining. Data mining deals with the extraction of knowledge from large databases. Several data mining algorithms extract dependencies of the form $j \rightarrow i$ (called association rules) from the database.

The main difference between Boolean analysis and the extraction of association rules in data mining is the interpretation of the extracted implications. The goal of a Boolean analysis is to extract implications from the data which are (with the exception of random errors in the response behavior) true for all rows in the data set. For data mining applications it is sufficient to detect implications which fulfill a predefined level of accuracy.

It is, for example in a marketing scenario, of interest to find implications which are true for more than x% of the rows in the data set. An online bookshop may be interested, for example, to search for implications of the form *If a customer orders book A he also orders book B* if they are fulfilled by more than 10% of the available customer data.

References

- Flament, C. (1976). L'analyse booleenne de questionnaire. Paris: Mouton.
- Buggenhaut, J., & Degreef, E. (1987). On dichotomization methods in Boolean analysis of questionnaires. In E. E. Roskam & R. Suck (Eds.), Mathematical psychology in progress (pp. 447–453). Amsterdam, NY: North Holland.
- Duquenne, V. (1987). Conceptual implications between attributes and some representation properties for finite lattices. In B. Ganter, R. Wille & K. E. Wolff (Eds.), Beiträge zur Begriffsanalyse: Vorträge der Arbeitstagung Begriffsanalyse, Darmstadt 1986 (pp. 213–239). Mannheim: BI Wissenschafts-Verlag.
- Leeuwe, J. F. J. van (1974). Item tree analysis. Nederlands Tijdschrift voor de Psychologie, 29, 475–484.
- Schrepp, M. (1999). On the empirical construction of implications on bi-valued test items. Journal of Mathematical Social Sciences, 38(3), 361–375.
- Theuns, P (1998). Building a knowledge space via Boolean analysis of co-occurrence data. In C. E. Dowling, F. S. Roberts, and P. Theuns (Eds.), Recent Progress in Mathematical Psychology (pp. 173–194). Hillsdale, NJ: Erlbaum.
- Bart, W. A., & Airasian P. W. (1974). Determination of the ordering among seven Piagetian tasks by an ordering-theoretic method. Journal of Educational Psychology, 66(2), 277–284.
- Gagné, R. M. (1968). Learning hierarchies. Educational Psychology, 6, 1–9.
- Scandura J. M. (1971). Deterministic theorizing in structural learning: Three levels of empiricism. Journal of Structural Learning, 3, 21–53.
- Bart, W. M., & Krus, D. J. (1973). An ordering-theoretic method to determine hierarchies among items. Educational and psychological measurement, 33, 291–300.
- Janssens, R. (1999). A Boolean approach to the measurement of group processes and attitudes. The concept of integration as an example. Mathematical Social Sciences, 38, 275–293.
- Held, T., & Korossy, K. (1998). Data-analysis as heuristic for establishing theoretically founded item structures. Zeitschrift für Psychologie, 206, 169–188.
- Ganter, B., & Wille, R. (1996). Formale Begriffsanalyse: Mathematische Grundlagen. Berlin: Springer.
- Romme, A.G.L. (1995). Boolean comparative analysis of qualitative data. Quality and Quantity, 29, 317-329.
- Romme, A.G.L. (1995). A Self-organizing processes in top management teams: a Boolean comparative approach. Journal of Business Research, 34, 11-34.
- Schrepp, M. (2003). A method for the analysis of hierarchical dependencies between items of a questionnaire. Methods of Psychological Research — Online, 19 ,43–79.

Boolean operations in computer-aided design

In computer-aided design (CAD), **Boolean operations** include the operations of subtraction, intersection, and union. The word "Boolean" refers to its similarity to the logical NOT, AND and OR operations, but it is somewhat a misnomer. Boolean operations exist in a wide variety of different applications; often used in coding and string generation with flowchart-like programming methods. More closely these operations are related to operations with sets. For example result of the "Subtract" operation represents set of points of the first object that do not belong to the second. The "Intersection" operation represents those that belong to both, and the "Union" operation represents those that belong to either.

Boolean operations are considered central functionality in most CAD systems. They are quite complex to implement. Their availability and robustness are tightly related to maturity of the software.

Both objects in an operation are expected to have same dimension (solids in three dimensional systems and entities with non-zero area in two dimensions).

There are divisions in object types due to different modelling technologies to represent its geometry. This often creates problem of executing a Boolean operation on different object types. For example subtraction of a meshed object imported from a STL file from another object imported from STEP often means that the operation either is not available or produces a meshed (non-parametric) object.

Non-manifold Boolean operations

In application to the common B-rep modelling techniques, clipping of one surface with another surface can be considered as a generalised Boolean operation. This means that Boolean operations can be generalized to non-solid BRep surfaces when its intersection is a well-defined (set of) curve on both surfaces.

References

- Mortenson, Michael, *Geometric Modeling*, N.Y.: Wiley Computer Publishing (1997) ISBN 0-47-12957-7.
- De Berg, M.; Cheong, O.; Van Kreveld, M.; Overmars, M. (2008), *Computational Geometry: Algorithms and Applications* (3), pp. 39–40, doi: 10.1007/978-3-540-77974-2 (<http://dx.doi.org/10.1007/978-3-540-77974-2>), ISBN 978-3-540-77973-5.

Circuit minimization

In Boolean algebra, **circuit minimization** is the problem of obtaining the smallest logic circuit (Boolean formula) that represents a given Boolean function or truth table. The general circuit minimization problem is believed to be intractable, but there are effective heuristics such as Karnaugh maps and the Quine–McCluskey algorithm that facilitate the process.

Purpose

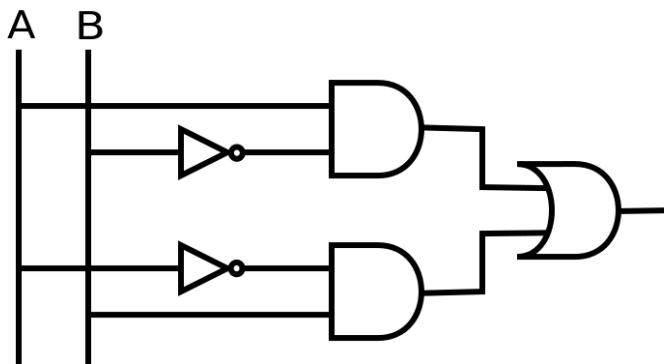
The problem with having a complicated circuit (i.e. one with many elements, such as logical gates) is that each element takes up physical space in its implementation and costs time and money to produce in itself. Circuit minimization may be one form of logic optimization used to reduce the area of complex logic in integrated circuits.

Example

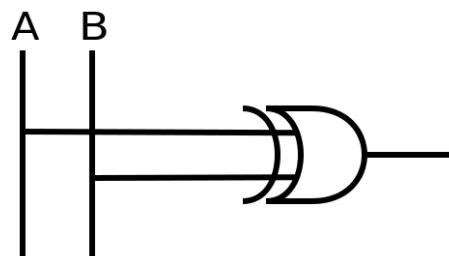
While there are many ways to minimize a circuit, this is an example that minimizes (or simplifies) a boolean function. Note that the boolean function carried out by the circuit is directly related to the algebraic expression from which the function is implemented.^[1] Consider the circuit used to represent $(A \wedge \bar{B}) \vee (\bar{A} \wedge B)$. It is evident that two negations, two conjunctions, and a disjunction are used in this statement. This means that to build the circuit one would need two inverters, two AND gates, and an OR gate.

We can simplify (minimize) the circuit by applying logical identities or using intuition. Since the example states that A is true when B is false or the other way around, we can conclude that this simply means $A \neq B$. In terms of logical gates, inequality simply means an XOR gate (exclusive or). Therefore, $(A \wedge \bar{B}) \vee (\bar{A} \wedge B) \iff A \neq B$. Then the two circuits shown below are equivalent:

Original Circuit



Simplified (Minimized) Circuit



You can additionally check the correctness of the result using a truth table.

References

[1] M. Mano, C. Kime. "Logic and Computer Design Fundamentals" (Fourth Edition). Pg 54

Further reading

- De Micheli, Giovanni. *Synthesis and Optimization of Digital Circuits*. McGraw-Hill, 1994, Part III, Logic-Level Syntesis and Optimization
- Zvi Kohavi, Niraj K. Jha. *Switching and Finite Automata Theory*. 3rd ed. Cambridge University Press. 2009. ISBN 978-0-521-85748-2, chapters 4–6
- Knuth, Donald E. (2010). "chapter 7.1.2: Boolean Evaluation". *The Art of Computer Programming 4A*. Addison-Wesley. pp. 96–133. ISBN 0-201-03804-8.

Espresso heuristic logic minimizer

The **Espresso logic minimizer** is a computer program using heuristic and specific algorithms for efficiently reducing the complexity of digital electronic gate circuits. Espresso was developed at IBM by Robert Brayton. Rudell later published the variant Espresso-MV in 1986 under the title "Multiple-Valued Logic Minimization for PLA Synthesis". Espresso has inspired many derivatives.

Introduction

Electronic devices are composed of numerous blocks of digital circuits, the combination of which performs the required task. The efficient implementation of logic functions in the form of logic gate circuits (such that no more logic gates are used than are necessary) is necessary to minimize production costs, and/or maximize a device's performance.

Designing digital logic circuits

All digital systems are composed of two elementary functions: memory elements for storing information, and combinational circuits that transform that information. State machines, like counters, are a combination of memory elements and combinational logic circuits. Since memory elements are standard logic circuits they are selected out of a limited set of alternative circuits; so designing digital functions comes down to designing the combinational gate circuits and interconnecting them.

In general the instantiation of logic circuits from high-level abstraction is referred to as Logic Synthesis, which can be carried out by hand, but usually some formal method by computer is applied. In this article the design methods for combinational logic circuits are briefly summarized.

The starting point for the design of a digital logic circuit is its desired functionality, having derived from the analysis of the system as a whole, the logic circuit is to make part of. The description can be stated in some algorithmic form or by logic equations, but may be summarized in the form of a table as well. The below example shows a part of such a table for a 7-segment driver that translates the binary code for the values of a decimal digit into the signals that cause the respective segments of the display to light up.

Digit	Code	Segments						
		A	B	C	D	E	F	G
0	0000	1	1	1	1	1	0	-A-
1	0001	0	1	1	0	0	0	0
2	0010	1	1	0	1	1	0	1
3	0011	1	1	1	1	0	0	1

4	0100	0 1 1 0 0 1 1	-G-
5	0101	1 0 1 1 0 1 1	
6	0110	1 0 1 1 1 1 1	E C
7	0111	1 1 1 0 0 0 0	
8	1000	1 1 1 1 1 1 1	-D-
9	1001	1 1 1 1 0 1 1	

The implementation process starts with a **logic minimization** phase, to be described below, in order to simplify the function table by combining the separate terms into larger ones containing fewer variables.

Next, the minimized result may be split up in smaller parts by a factorization procedure and is eventually mapped onto the available basic logic cells of the target technology. This operation is commonly referred to as Logic Optimization.

Classical minimization methods

Minimizing Boolean functions by hand using the classical Karnaugh maps is a laborious, tedious and error prone process. It isn't suited for more than 6 input variables and practical only for up to 4 variables, while product term sharing for multiple output functions is even harder to carry out. Moreover, this method doesn't lend itself to be automated in the form of a computer program. However, since modern logic functions are generally not constrained to such a small number of variables, while the cost as well as the risk of making errors is prohibitive for manual implementation of logic functions, the use of computers became indispensable.

The first alternative method to become popular was the tabular method developed by Quine and McCluskey. Starting with the truth table for a set of logic functions, by combining the minterms for which the functions are active—the ON-cover—or for which the function value is irrelevant—the Don't-Care-cover or DC-cover—a set of prime implicants is composed. Finally a systematic procedure is followed to find the smallest set of prime implicants the output functions can be realised with.

Although this Quine–McCluskey algorithm is very well suited to be implemented in a computer program, the result is still far from efficient in terms of processing time and memory usage. Adding a variable to the function will roughly double both of them, because the truth table length increases exponentially with the number of variables. A similar problem occurs when increasing the number of output functions of a combinational function block. As a result the Quine–McCluskey method is practical only for functions with a limited number of input variables and output functions.

Espresso algorithm

A radically different approach to this issue is followed in the ESPRESSO algorithm, developed by Brayton e.a. at the University of California, Berkeley. Rather than expanding a logic function into minterms, the program manipulates "cubes", representing the product terms in the ON-, DC- and OFF-covers iteratively. Although the minimization result is not guaranteed to be the global minimum, in practice this is very closely approximated, while the solution is always free from redundancy. Compared to the other methods, this one is essentially more efficient, reducing memory usage and computation time by several orders of magnitude. Its name reflects the way of instantly making a cup of fresh coffee. There is hardly any restriction to the number of variables, output functions and product terms of a combinational function block. In general, e.g. tens of variables with tens of output functions are readily dealt with.

The input for ESPRESSO is a function table of the desired functionality; the result is a minimized table, describing either the ON-cover or the OFF-cover of the function, depending on the selected options. By default the product terms will be shared as much as possible by the several output functions, but the program can be instructed to handle each of the output functions separately. This allows for efficient implementation in two-level logic arrays such as a PLA (Programmable Logic Array) or a PAL (Programmable Array Logic).

The ESPRESSO algorithm proved so successful that it has been incorporated as a standard logic function minimization step into virtually any contemporary logic synthesis tool. For implementing a function in multi-level logic, the minimization result is optimized by factorization and mapped onto the available basic logic cells in the target technology, whether this concerns an FPGA (Field Programmable Gate Array) or an ASIC (Application Specific Integrated Circuit).

Software

Minilog

Minilog is a logic minimization program exploiting this ESPRESSO algorithm. It is able to generate a two-level gate implementation for a combinational function block with up to 40 inputs and outputs or a synchronous state machine with up to 256 states. It is part of the **Publicad** educational design package, that can be downloaded from the website Publicad^[1] - free Publicad toolkit including Minilog logic minimization program.

Logic Friday

Logic Friday is a free Windows program that provides a graphical interface to ESPRESSO, as well as to misII, another module in the Berkeley Octtools package. With Logic Friday users can enter a logic function as a truth table, equation, or gate diagram, minimize the function, and then view the results in both of the other two representations. Logic Friday is available at <http://www.sontrak.com>.

Espresso sources

The source of the original Espresso program is available from the website of the University of California, Berkeley, at Pubs/Downloads/Espresso^[2]. A version of Espresso that has been updated for modern POSIX systems is available at [3]

References

- [1] <http://cid-b7034765ca89b294.skydrive.live.com/browse.aspx/Quick-Install>
 - [2] <http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm>
 - [3] <ftp://ftp.cs.man.ac.uk/pub/amulet/balsa/other-software/espresso-ab-1.0.tar.gz>
-

Logic gate

A **logic gate** is an idealized or physical device implementing a Boolean function, that is, it performs a logical operation on one or more logical inputs, and produces a single logical output. Depending on the context, the term may refer to an **ideal logic gate**, one that has for instance zero rise time and unlimited fan-out, or it may refer to a non-ideal physical device^[1] (see Ideal and real op-amps for comparison).

Logic gates are primarily implemented using diodes or transistors acting as electronic switches, but can also be constructed using electromagnetic relays (relay logic), fluidic logic, pneumatic logic, optics, molecules, or even mechanical elements. With amplification, logic gates can be cascaded in the same way that Boolean functions can be composed, allowing the construction of a physical model of all of Boolean logic, and therefore, all of the algorithms and mathematics that can be described with Boolean logic.

Logic circuits include such devices as multiplexers, registers, arithmetic logic units (ALUs), and computer memory, all the way up through complete microprocessors, which may contain more than 100 million gates. In practice, the gates are made from field-effect transistors (FETs), particularly MOSFETs (metal–oxide–semiconductor field-effect transistors).

Compound logic gates AND-OR-Invert (AOI) and OR-AND-Invert (OAI) are often employed in circuit design because their construction using MOSFETs is simpler and more efficient than the sum of the individual gates.

In reversible logic, Toffoli gates are used.

Electronic gates

To build a functionally complete logic system, relays, valves (vacuum tubes), or transistors can be used. The simplest family of logic gates using bipolar transistors is called resistor-transistor logic (RTL). Unlike diode logic gates, RTL gates can be cascaded indefinitely to produce more complex logic functions. These gates were used in early integrated circuits. For higher speed, the resistors used in RTL were replaced by diodes, leading to diode-transistor logic (DTL). Transistor-transistor logic (TTL) then supplanted DTL with the observation that one transistor could do the job of two diodes even more quickly, using only half the space. In virtually every type of contemporary chip implementation of digital systems, the bipolar transistors have been replaced by complementary field-effect transistors (MOSFETs) to reduce size and power consumption still further, thereby resulting in complementary metal–oxide–semiconductor (CMOS) logic.

For small-scale logic, designers now use prefabricated logic gates from families of devices such as the TTL 7400 series by Texas Instruments and the CMOS 4000 series by RCA, and their more recent descendants. Increasingly, these fixed-function logic gates are being replaced by programmable logic devices, which allow designers to pack a large number of mixed logic gates into a single integrated circuit. The field-programmable nature of programmable logic devices such as FPGAs has removed the 'hard' property of hardware; it is now possible to change the logic design of a hardware system by reprogramming some of its components, thus allowing the features or function of a hardware implementation of a logic system to be changed.

Electronic logic gates differ significantly from their relay-and-switch equivalents. They are much faster, consume much less power, and are much smaller (all by a factor of a million or more in most cases). Also, there is a fundamental structural difference. The switch circuit creates a continuous metallic path for current to flow (in either direction) between its input and its output. The semiconductor logic gate, on the other hand, acts as a high-gain voltage amplifier, which sinks a tiny current at its input and produces a low-impedance voltage at its output. It is not possible for current to flow between the output and the input of a semiconductor logic gate.

Another important advantage of standardized integrated circuit logic families, such as the 7400 and 4000 families, is that they can be cascaded. This means that the output of one gate can be wired to the inputs of one or several other gates, and so on. Systems with varying degrees of complexity can be built without great concern of the designer for

the internal workings of the gates, provided the limitations of each integrated circuit are considered.

The output of one gate can only drive a finite number of inputs to other gates, a number called the 'fanout limit'. Also, there is always a delay, called the 'propagation delay', from a change in input of a gate to the corresponding change in its output. When gates are cascaded, the total propagation delay is approximately the sum of the individual delays, an effect which can become a problem in high-speed circuits. Additional delay can be caused when a large number of inputs are connected to an output, due to the distributed capacitance of all the inputs and wiring and the finite amount of current that each output can provide.

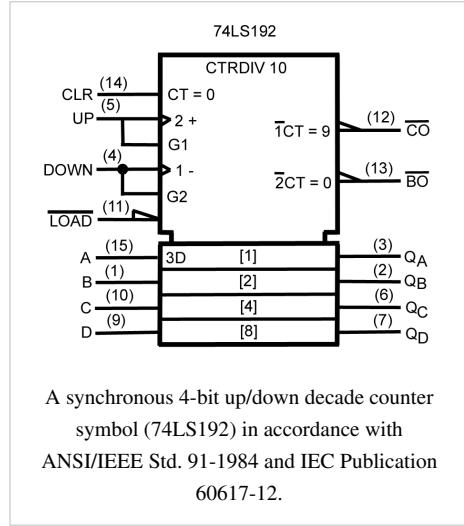
Symbols

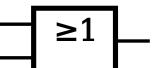
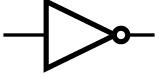
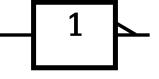
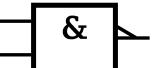
There are two sets of symbols for elementary logic gates in common use, both defined in ANSI/IEEE Std 91-1984 and its supplement ANSI/IEEE Std 91a-1991. The "distinctive shape" set, based on traditional schematics, is used for simple drawings, and derives from MIL-STD-806 of the 1950s and 1960s. It is sometimes unofficially described as "military", reflecting its origin. The "rectangular shape" set, based on IEC 60617-12 and other early industry standards, has rectangular outlines for all types of gate and allows representation of a much wider range of devices than is possible with the traditional symbols.^[2] The IEC's system has been adopted by other standards, such as EN 60617-12:1999 in Europe and BS EN 60617-12:1999 in the United Kingdom.

The goal of IEEE Std 91-1984 was to provide a uniform method of describing the complex logic functions of digital circuits with schematic symbols. These functions were more complex than simple AND and OR gates. They could be medium scale circuits such as a 4-bit counter to a large scale circuit such as a microprocessor. IEC 617-12 and its successor IEC 60617-12 do not explicitly show the "distinctive shape" symbols, but do not prohibit them. These are, however, shown in ANSI/IEEE 91 (and 91a) with this note: "The distinctive-shape symbol is, according to IEC Publication 617, Part 12, not preferred, but is not considered to be in contradiction to that standard." This compromise was reached between the respective IEEE and IEC working groups to permit the IEEE and IEC standards to be in mutual compliance with one another.

A third style of symbols was in use in Europe and is still preferred by some, see the column "DIN 40700" in the table in the German Wikipedia.

In the 1980s, schematics were the predominant method to design both circuit boards and custom ICs known as gate arrays. Today custom ICs and the field-programmable gate array are typically designed with Hardware Description Languages (HDL) such as Verilog or VHDL.



Type	Distinctive shape	Rectangular shape	Boolean algebra between A & B	Truth table																		
AND			$A \cdot B$	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>A AND B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	A AND B	0	0	0	0	1	0	1	0	0	1	1	1
INPUT		OUTPUT																				
A	B	A AND B																				
0	0	0																				
0	1	0																				
1	0	0																				
1	1	1																				
OR			$A + B$	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>A OR B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	A OR B	0	0	0	0	1	1	1	0	1	1	1	1
INPUT		OUTPUT																				
A	B	A OR B																				
0	0	0																				
0	1	1																				
1	0	1																				
1	1	1																				
NOT			\bar{A}	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th></th> <th>NOT A</th> </tr> </thead> <tbody> <tr> <td>0</td> <td></td> <td>1</td> </tr> <tr> <td>1</td> <td></td> <td>0</td> </tr> </tbody> </table>	INPUT		OUTPUT	A		NOT A	0		1	1		0						
INPUT		OUTPUT																				
A		NOT A																				
0		1																				
1		0																				
<p>In electronics a NOT gate is more commonly called an inverter. The circle on the symbol is called a <i>bubble</i>, and is used in logic diagrams to indicate a logic negation between the external logic state and the internal logic state (1 to 0 or vice versa). On a circuit diagram it must be accompanied by a statement asserting that the <i>positive logic convention</i> or <i>negative logic convention</i> is being used (high voltage level = 1 or high voltage level = 0, respectively). The <i>wedge</i> is used in circuit diagrams to directly indicate an active-low (high voltage level = 0) input or output without requiring a uniform convention throughout the circuit diagram. This is called <i>Direct Polarity Indication</i>. See IEEE Std 91/91A and IEC 60617-12. Both the <i>bubble</i> and the <i>wedge</i> can be used on distinctive-shape and rectangular-shape symbols on circuit diagrams, depending on the logic convention used. On pure logic diagrams, only the <i>bubble</i> is meaningful.</p>																						
NAND			$A \cdot \bar{B}$	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>A NAND B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	A NAND B	0	0	1	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																				
A	B	A NAND B																				
0	0	1																				
0	1	1																				
1	0	1																				
1	1	0																				

NOR			$A + \bar{B}$	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>A NOR B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	A NOR B	0	0	1	0	1	0	1	0	0	1	1	0
INPUT		OUTPUT																				
A	B	A NOR B																				
0	0	1																				
0	1	0																				
1	0	0																				
1	1	0																				
XOR			$A \oplus B$	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>A XOR B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	A XOR B	0	0	0	0	1	1	1	0	1	1	1	0
INPUT		OUTPUT																				
A	B	A XOR B																				
0	0	0																				
0	1	1																				
1	0	1																				
1	1	0																				
XNOR			$A \oplus \bar{B}$ or $A \odot B$	<table border="1"> <thead> <tr> <th colspan="2">INPUT</th> <th>OUTPUT</th> </tr> <tr> <th>A</th> <th>B</th> <th>A XNOR B</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	INPUT		OUTPUT	A	B	A XNOR B	0	0	1	0	1	0	1	0	0	1	1	1
INPUT		OUTPUT																				
A	B	A XNOR B																				
0	0	1																				
0	1	0																				
1	0	0																				
1	1	1																				

Two more gates are the exclusive-OR or XOR function and its inverse, exclusive-NOR or XNOR. The two input Exclusive-OR is true only when the two input values are *different*, false if they are equal, regardless of the value. If there are more than two inputs, the gate generates a true at its output if the number of trues at its input is *odd* ([3]). In practice, these gates are built from combinations of simpler logic gates.

Universal logic gates

Charles Sanders Peirce (winter of 1880–81) showed that NOR gates alone (or alternatively NAND gates alone) can be used to reproduce the functions of all the other logic gates, but his work on it was unpublished until 1933.^[4] The first published proof was by Henry M. Sheffer in 1913, so the NAND logical operation is sometimes called Sheffer stroke; the logical NOR is sometimes called *Peirce's arrow*. Consequently, these gates are sometimes called *universal logic gates*.

De Morgan equivalent symbols

By use of De Morgan's theorem, an *AND* function is identical to an *OR* function with negated inputs and outputs. Likewise, an *OR* function is identical to an *AND* function with negated inputs and outputs. A *NAND* gate is equivalent to an *OR* gate with negated inputs, and a *NOR* gate is equivalent to an *AND* gate with negated inputs.

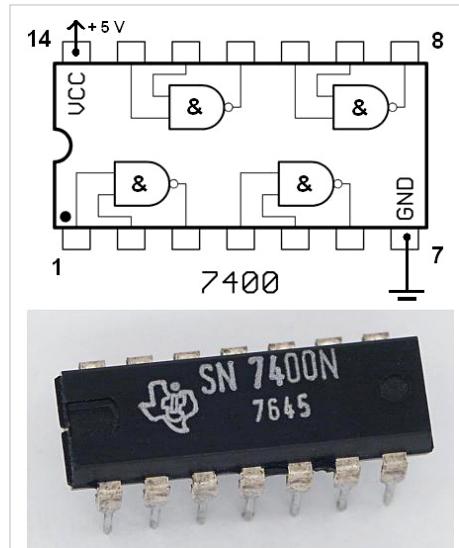
This leads to an alternative set of symbols for basic gates that use the opposite core symbol (*AND* or *OR*) but with the inputs and outputs negated. Use of these alternative symbols can make logic circuit diagrams much clearer and help to show accidental connection of an active high output to an active low input or vice-versa. Any connection that has logic negations at both ends can be replaced by a negationless connection and a suitable change of gate or vice-versa. Any connection that has a negation at one end and no negation at the other can be made easier to interpret by instead using the De Morgan equivalent symbol at either of the two ends. When negation or polarity indicators on both ends of a connection match, there is no logic negation in that path (effectively, bubbles "cancel"), making it easier to follow logic states from one symbol to the next. This is commonly seen in real logic diagrams - thus the reader must not get into the habit of associating the shapes exclusively as *OR* or *AND* shapes, but also take into account the bubbles at both inputs and outputs in order to determine the "true" logic function indicated.

De Morgan's theorem is most commonly used to implement logic gates as combinations of only *NAND* gates, or as combinations of only *NOR* gates, for economic reasons.

Data storage

Logic gates can also be used to store data. A storage element can be constructed by connecting several gates in a "latch" circuit. More complicated designs that use clock signals and that change only on a rising or falling edge of the clock are called edge-triggered "flip-flops". The combination of multiple flip-flops in parallel, to store a multiple-bit value, is known as a register. When using any of these gate setups the overall system has memory; it is then called a sequential logic system since its output can be influenced by its previous state(s).

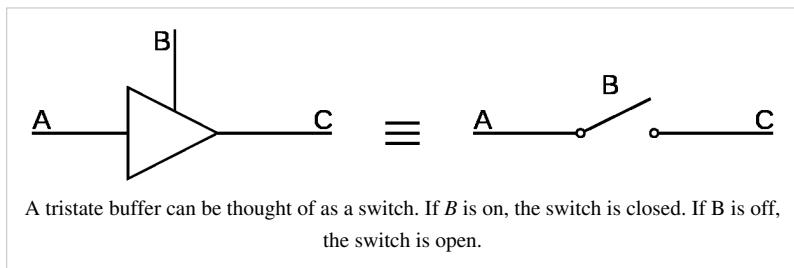
These logic circuits are known as computer memory. They vary in performance, based on factors of speed, complexity, and reliability of storage, and many different types of designs are used based on the application.



The 7400 chip, containing four NANDs. The two additional pins supply power (+5 V) and connect the ground.

Three-state logic gates

A three-state logic gate is a type of logic gate that can have three different outputs: high (H), low (L) and high-impedance (Z). The high-impedance state plays no role in the logic, which is strictly binary. These devices are used on buses of the CPU to allow multiple chips to send data. A group of three-states driving a line with a suitable control circuit is basically equivalent to a multiplexer, which may be physically distributed over separate devices or plug-in cards.



A tristate buffer can be thought of as a switch. If B is on, the switch is closed. If B is off, the switch is open.

In electronics, a high output would mean the output is sourcing current from the positive power terminal (positive voltage). A low output would mean the output is sinking current to the negative power terminal (zero voltage). High impedance would mean that the output is effectively disconnected from the circuit.

History and development

The binary number system was refined by Gottfried Wilhelm Leibniz (published in 1705) and he also established that by using the binary system, the principles of arithmetic and logic could be combined. In a 1886 letter, Charles Sanders Peirce described how logical operations could be carried out by electrical switching circuits.^[5] Eventually, vacuum tubes replaced relays for logic operations. Lee De Forest's modification, in 1907, of the Fleming valve can be used as AND logic gate. Ludwig Wittgenstein introduced a version of the 16-row truth table as proposition 5.101 of *Tractatus Logico-Philosophicus* (1921). Walther Bothe, inventor of the coincidence circuit, got part of the 1954 Nobel Prize in physics, for the first modern electronic AND gate in 1924. Konrad Zuse designed and built electromechanical logic gates for his computer Z1 (from 1935-38). Claude E. Shannon introduced the use of Boolean algebra in the analysis and design of switching circuits in 1937. Active research is taking place in molecular logic gates.

Implementations

Since the 1990s, most logic gates are made in CMOS technology (i.e. NMOS and PMOS transistors are used). Often millions of logic gates are packaged in a single integrated circuit.

There are several logic families with different characteristics (power consumption, speed, cost, size) such as: RDL (resistor-diode logic), RTL (resistor-transistor logic), DTL (diode-transistor logic), TTL (transistor-transistor logic) and CMOS (complementary metal oxide semiconductor). There are also sub-variants, e.g. standard CMOS logic vs. advanced types using still CMOS technology, but with some optimizations for avoiding loss of speed due to slower PMOS transistors.

Non-electronic implementations are varied, though few of them are used in practical applications. Many early electromechanical digital computers, such as the Harvard Mark I, were built from relay logic gates, using electro-mechanical relays. Logic gates can be made using pneumatic devices, such as the Sorteberg relay or mechanical logic gates, including on a molecular scale.^[6] Logic gates have been made out of DNA (see DNA nanotechnology)^[7] and used to create a computer called MAYA (see MAYA II). Logic gates can be made from quantum mechanical effects (though quantum computing usually diverges from boolean design). Photonic logic gates use non-linear optical effects.

References

- [1] Jaeger, Microelectronic Circuit Design, McGraw-Hill 1997, ISBN 0-07-032482-4, pp. 226-233
- [2] Overview of IEEE Standard 91-1984 Explanation of Logic Symbols (<http://www.ti.com/lit/ml/sdyz001a/sdyz001a.pdf>), Doc. No. SDYZ001A, Texas Instruments Semiconductor Group, 1996
- [3] <http://www-inst.eecs.berkeley.edu/~cs61c/resources/dg-BOOL-handout.pdf>
- [4] Peirce, C. S. (manuscript winter of 1880–81), "A Boolean Algebra with One Constant", published 1933 in *Collected Papers* v. 4, paragraphs 12–20. Reprinted 1989 in *Writings of Charles S. Peirce* v. 4, pp. 218-21, Google Preview (<http://books.google.com/books?id=E7ZUnx3FqrcC&q=378+Winter>). See Roberts, Don D. (2009), *The Existential Graphs of Charles S. Peirce*, p. 131.
- [5] Peirce, C. S., "Letter, Peirce to A. Marquand", dated 1886, *Writings of Charles S. Peirce*, v. 5, 1993, pp. 541–3. Google Preview (http://books.google.com/books?id=DnvLHp919_wC&q=Marquand). See Burks, Arthur W., "Review: Charles S. Peirce, *The new elements of mathematics*", *Bulletin of the American Mathematical Society* v. 84, n. 5 (1978), pp. 913–18, see 917. PDF Eprint (http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body&id=pdf_1&handle=euclid.bams/1183541145).
- [6] Mechanical Logic gates (focused on molecular scale) (<http://www.zyvex.com/nanotech/mechano.html>)
- [7] DNA Logic gates (<https://digamma.cs.unm.edu/wiki/bin/view/McogPublicWeb/MolecularLogicGates>)

Further reading

- Awschalom, D.D.; Loss, D.; Samarth, N. (5 August 2002). *Semiconductor Spintronics and Quantum Computation* (http://books.google.com/books?id=tlDSx_8_5v4C). Berlin, Germany: Springer-Verlag. ISBN 978-3-540-42176-4. Retrieved 28 November 2012.
- Bostock, Geoff (1988). *Programmable logic devices: technology and applications* (<http://books.google.com/books?id=XEFTAAAAMAAJ>). New York: McGraw-Hill. ISBN 978-0-07-006611-3. Retrieved 28 November 2012.
- Brown, Stephen D.; Francis, Robert J.; Rose, Jonathan; Vranesic, Zvonko G. (1992). *Field Programmable Gate Arrays* (<http://books.google.com/books?id=8s4M-qYOWZIC>). Boston, MA: Kluwer Academic Publishers. ISBN 978-0-7923-9248-4. Retrieved 28 November 2012.

People

Augustus De Morgan

Augustus De Morgan	
	
	Augustus De Morgan (1806-1871)
Born	27 June 1806 Madurai, Madras Presidency, British Empire (present-day India)
Died	18 March 1871 (aged 64) London, England
Residence	India England
Nationality	British
Fields	Mathematician and logician
Institutions	University College London University College School
Alma mater	Trinity College University of Cambridge
Academic advisors	John Philips Higman George Peacock William Whewell
Notable students	Edward Routh James Joseph Sylvester Frederick Guthrie William Stanley Jevons Ada Lovelace Francis Guthrie Stephen Joseph Perry
Known for	De Morgan's laws De Morgan algebra Relation algebra Universal algebra
Influences	George Boole
Influenced	Thomas Corwin Mendenhall Isaac Todhunter

Notes

He was the father of William De Morgan.

Augustus De Morgan (27 June 1806 – 18 March 1871) was a British mathematician and logician. He formulated De Morgan's laws and introduced the term mathematical induction, making its idea rigorous.^[1]

Biography

Childhood

Augustus De Morgan was born in Madurai, Madras Presidency, India in 1806.^[2] His father was Colonel Augustus De Morgan, who held various appointments in the service of the East India Company. His mother descended from James Dodson, who computed a table of anti-logarithms, that is, the numbers corresponding to exact logarithms. Augustus De Morgan became blind in one eye a month or two after he was born. The family moved to England when Augustus was seven months old. As his father and grandfather had both been born in India, De Morgan used to say that he was neither English, nor Scottish, nor Irish, but a Briton "unattached", using the technical term applied to an undergraduate of Oxford or Cambridge who is not a member of any one of the Colleges.

When De Morgan was ten years old, his father died. Mrs. De Morgan resided at various places in the southwest of England, and her son received his elementary education at various schools of no great account. His mathematical talents went unnoticed until he was fourteen, when a family-friend discovered him making an elaborate drawing of a figure in Euclid with ruler and compasses. She explained the aim of Euclid to Augustus, and gave him an initiation into demonstration.

He received his secondary education from Mr. Parsons, a fellow of Oriel College, Oxford, who appreciated classics better than mathematics. His mother was an active and ardent member of the Church of England, and desired that her son should become a clergyman; but by this time De Morgan had begun to show his non-conforming disposition.

University education

In 1823, at the age of sixteen, he entered Trinity College, Cambridge, where he came under the influence of George Peacock and William Whewell, who became his lifelong friends; from the former he derived an interest in the renovation of algebra, and from the latter an interest in the renovation of logic—the two subjects of his future life work. His college tutor was John Philips Higman, FRS (1793–1855).

At college he played the flute for recreation and was prominent in the musical clubs. His love of knowledge for its own sake interfered with training for the great mathematical race; as a consequence he came out fourth wrangler. This entitled him to the degree of Bachelor of Arts; but to take the higher degree of Master of Arts and thereby become eligible for a fellowship it was then necessary to pass a theological test. To the signing of any such test De Morgan felt a strong objection, although he had been brought up in the Church of England. In about 1875 theological tests for academic degrees were abolished in the Universities of Oxford and Cambridge.

London University

As no career was open to him at his own university, he decided to go to the Bar, and took up residence in London; but he much preferred teaching mathematics to reading law. About this time the movement for founding London University (now University College London) took shape. The two ancient universities of Oxford and Cambridge were so guarded by theological tests that no Jew or Dissenter outside the Church of England could enter as a student, still less be appointed to any office. A body of liberal-minded men resolved to meet the difficulty by establishing in London a University on the principle of religious neutrality. De Morgan, then 22 years of age, was appointed professor of mathematics. His introductory lecture "On the study of mathematics" is a discourse upon mental education of permanent value, and has been recently reprinted in the United States.

The London University was a new institution, and the relations of the Council of management, the Senate of professors and the body of students were not well defined. A dispute arose between the professor of anatomy and his students, and in consequence of the action taken by the Council, several professors resigned, headed by De Morgan. Another professor of mathematics was appointed, who then drowned a few years later. De Morgan had shown himself a prince of teachers: he was invited to return to his chair, which thereafter became the continuous centre of his labours for thirty years.

The same body of reformers—headed by Lord Brougham, a Scotsman eminent both in science and politics who had instituted the London University—founded about the same time a Society for the Diffusion of Useful Knowledge. Its object was to spread scientific and other knowledge by means of cheap and clearly written treatises by the best writers of the time. One of its most voluminous and effective writers was De Morgan. He wrote a great work on *The Differential and Integral Calculus* which was published by the Society; and he wrote one-sixth of the articles in the *Penny Cyclopaedia*, published by the Society, and issued in penny numbers. When De Morgan came to reside in London he found a congenial friend in William Frend, notwithstanding his mathematical heresy about negative quantities. Both were arithmeticians and actuaries, and their religious views were somewhat similar. Frend lived in what was then a suburb of London, in a country-house formerly occupied by Daniel Defoe and Isaac Watts. De Morgan with his flute was a welcome visitor.

The London University of which De Morgan was a professor was a different institution from the University of London. The University of London was founded about ten years later by the Government for the purpose of granting degrees after examination, without any qualification as to residence. The London University was affiliated as a teaching college with the University of London, and its name was changed to University College. The University of London was not a success as an examining body; a teaching University was demanded. De Morgan was a highly successful teacher of mathematics. It was his plan to lecture for an hour, and at the close of each lecture to give out a number of problems and examples illustrative of the subject lectured on; his students were required to sit down to them and bring him the results, which he looked over and returned revised before the next lecture. In De Morgan's opinion, a thorough comprehension and mental assimilation of great principles far outweighed in importance any merely analytical dexterity in the application of half-understood principles to particular cases.

During this period, he also promoted the work of the self-taught Indian mathematician Ramchundra, who has been called De Morgan's Ramanujam. He supervised the publication in London of Ramchundra's book on "Maxima and Minima" in 1859. In the introduction to this book, he acknowledged being aware of the Indian tradition of logic, although it is not known whether this had any influence on his own work.

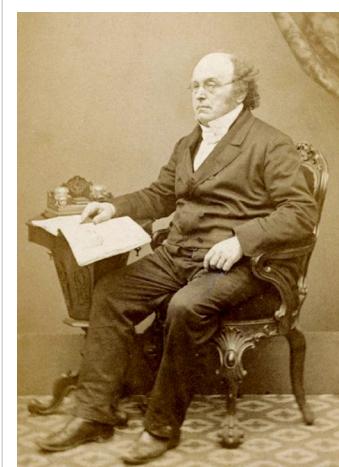
Family

In the autumn of 1837, he married Sophia Elizabeth, eldest daughter of William Frend and his wife, a granddaughter of Archdeacon Francis Blackburne.

De Morgan had three sons and four daughters, including fairytale author Mary de Morgan. His eldest son was the potter William De Morgan. His second son George acquired great distinction in mathematics at University College and the University of London. He and another like-minded alumnus conceived the idea of founding a Mathematical Society in London, where mathematical papers would be not only received (as by the Royal Society) but actually read and discussed. The first meeting was held in University College; De Morgan was the first president, his son the first secretary. It was the beginning of the London Mathematical Society.

Retirement and death

In 1866 the chair of mental philosophy in University College fell vacant. James Martineau, a Unitarian clergyman and professor of mental philosophy, was recommended formally by the Senate to the Council; but in the Council there were some who objected to a Unitarian clergyman, and others who objected to theistic philosophy. A layman of the school of Bain and Spencer was appointed. De Morgan considered that the old standard of religious neutrality had been hauled down, and forthwith resigned. He was now 60 years of age. His pupils secured him a pension of £500 p.a., but misfortunes followed. Two years later his son George — the "younger Bernoulli", as Augustus loved to hear him called, in allusion to the eminent father-and-son mathematicians of that name — died. This blow was followed by the death of a daughter. Five years after his resignation from University College De Morgan died of nervous prostration on 18 March 1871.



Augustus De Morgan.

Mathematical work

De Morgan was a brilliant and witty writer, whether as a controversialist or as a correspondent. In his time there flourished two Sir William Hamiltons who have often been conflated. One was Sir William Hamilton, 9th Baronet (that is, his title was inherited), a Scotsman, professor of logic and metaphysics at the University of Edinburgh; the other was a knight (that is, won the title), an Irishman, professor of astronomy in the University of Dublin. The baronet contributed to logic, especially the doctrine of the quantification of the predicate; the knight, whose full name was William Rowan Hamilton, contributed to mathematics, especially geometric algebra, and first described the Quaternions. De Morgan was interested in the work of both, and corresponded with both; but the correspondence with the Scotsman ended in a public controversy, whereas that with the Irishman was marked by friendship and terminated only by death. In one of his letters to Rowan, De Morgan says,

Be it known unto you that I have discovered that you and the other Sir W. H. are reciprocal polars with respect to me (intellectually and morally, for the Scottish baronet is a polar bear, and you, I was going to say, are a polar gentleman). When I send a bit of investigation to Edinburgh, the W. H. of that ilk says I took it from him. When I send you one, you take it from me, generalize it at a glance, bestow it thus generalized upon society at large, and make me the second discoverer of a known theorem.

The correspondence of De Morgan with Hamilton the mathematician extended over twenty-four years; it contains discussions not only of mathematical matters, but also of subjects of general interest. It is marked by geniality on the part of Hamilton and by wit on the part of De Morgan. The following is a specimen: Hamilton wrote,

My copy of Berkeley's work is not mine; like Berkeley, you know, I am an Irishman.

De Morgan replied,

Your phrase 'my copy is not mine' is not a bull. It is perfectly good English to use the same word in two different senses in one sentence, particularly when there is usage. Incongruity of language is no bull, for it expresses meaning. But incongruity of ideas (as in the case of the Irishman who was pulling up the rope, and finding it did not finish, cried out that somebody had cut off the other end of it) is the genuine bull.

De Morgan was full of personal peculiarities. On the occasion of the installation of his friend, Lord Brougham, as Rector of the University of Edinburgh, the Senate offered to confer on him the honorary degree of LL. D.; he declined the honour as a misnomer. He once printed his name: Augustus De Morgan, *H - O - M - O - P - A - U - C - A - R - U - M - L - I - T - E - R - A - R - U - M* (Latin for "man of few letters").^[citation needed]

He disliked the provinces outside London, and while his family enjoyed the seaside, and men of science were having a good time at a meeting of the British Association in the country he remained in the hot and dusty libraries of the metropolis. He said that he felt like Socrates, who declared that the farther he was from Athens the farther was he from happiness. He never sought to become a Fellow of the Royal Society, and he never attended a meeting of the Society; he said that he had no ideas or sympathies in common with the physical philosopher. His attitude was possibly due to his physical infirmity, which prevented him from being either an observer or an experimenter. He never voted at an election, and he never visited the House of Commons, or the Tower of London, or Westminster Abbey.

Were the writings of De Morgan published in the form of collected works, they would form a small library, for example his writings for the Useful Knowledge Society. Mainly through the efforts of Peacock and Whewell, a Philosophical Society had been inaugurated at Cambridge; and to its Transactions De Morgan contributed four memoirs on the foundations of algebra, and an equal number on formal logic. The best presentation of his view of algebra is found in a volume, entitled *Trigonometry and Double Algebra*, published in 1849; and his earlier view of formal logic is found in a volume published in 1847. His most distinctive work is styled a *Budget of Paradoxes*; it originally appeared as letters in the columns of the *Athenaeum* journal; it was revised and extended by De Morgan in the last years of his life, and was published posthumously by his widow.

George Peacock's theory of algebra was much improved by D. F. Gregory, a younger member of the Cambridge School, who laid stress not on the permanence of equivalent forms, but on the permanence of certain formal laws. This new theory of algebra as the science of symbols and of their laws of combination was carried to its logical issue by De Morgan; and his doctrine on the subject is still followed by English algebraists in general. Thus George Chrystal founds his *Textbook of Algebra* on De Morgan's theory; although an attentive reader may remark that he practically abandons it when he takes up the subject of infinite series. De Morgan's theory is stated in his volume on *Trigonometry and Double Algebra*. In the chapter (of the book) headed "On symbolic algebra" he writes:

In abandoning the meaning of symbols, we also abandon those of the words which describe them. Thus addition is to be, for the present, a sound void of sense. It is a mode of combination represented by $+$; when $+$ receives its meaning, so also will the word addition. It is most important that the student should bear in mind that, with one exception, no word nor sign of arithmetic or algebra has one atom of meaning throughout this chapter, the object of which is symbols, and their laws of combination, giving a symbolic algebra which may hereafter become the grammar of a hundred distinct significant algebras. If any one were to assert that $+$ and $-$ might mean reward and punishment, and A , B , C , etc., might stand for virtues and vices, the reader might believe him, or contradict him, as he pleases, but not out of this chapter. The one exception above noted, which has some share of meaning, is the sign $=$ placed between two symbols as in $A = B$. It indicates that the two symbols have the same resulting meaning, by whatever steps attained. That A and B , if quantities, are the same amount of quantity; that if operations, they are of the same effect, etc.

Here, it may be asked, why does the symbol $=$ prove refractory to the symbolic theory? De Morgan admits that there is one exception; but an exception proves the rule, not in the usual but illogical sense of establishing it, but in the old and logical sense of testing its validity. If an exception can be established, the rule must fall, or at least must be modified. Here I am talking not of grammatical rules, but of the rules of science or nature.

De Morgan proceeds to give an inventory of the fundamental symbols of algebra, and also an inventory of the laws of algebra. The symbols are 0 , 1 , $+$, $-$, \times , \div , $()^0$, and letters; these only, all others are derived. His inventory of the fundamental laws is expressed under fourteen heads, but some of them are merely definitions. The laws proper may be reduced to the following, which, as he admits, are not all independent of one another:

1. Law of signs. $++ = +$, $+ - = -$, $- + = -$, $- - = +$, $\times \times = \times$, $\times \div = \div$, $\div \times = \div$, $\div \div = \times$.
2. Commutative law. $a+b = b+a$, $ab=ba$.
3. Distributive law. $a(b+c) = ab+ac$.
4. Index laws. $a^b \times a^c = a^{b+c}$, $(a^b)^c = a^{bc}$, $(ab)^d = a^d \times b^d$.

5. $a-a=0$, $a\div a=1$.

The last two may be called the rules of reduction. De Morgan professes to give a complete inventory of the laws which the symbols of algebra must obey, for he says, "Any system of symbols which obeys these laws and no others, except they be formed by combination of these laws, and which uses the preceding symbols and no others, except they be new symbols invented in abbreviation of combinations of these symbols, is symbolic algebra." From his point of view, none of the above principles are rules; they are formal laws, that is, arbitrarily chosen relations to which the algebraic symbols must be subject. He does not mention the law, which had already been pointed out by Gregory, namely, $(a+b)+c = a+(b+c)$, $(ab)c = a(bc)$ and to which was afterwards given the name of the *law of association*. If the commutative law fails, the associative may hold good; but not *vice versa*. It is an unfortunate thing for the symbolist or formalist that in universal arithmetic m^n is not equal to n^m ; for then the commutative law would have full scope. Why does he not give it full scope? Because the foundations of algebra are, after all, real not formal, material not symbolic. To the formalists the index operations are exceedingly refractory, in consequence of which some take no account of them, but relegate them to applied mathematics. To give an inventory of the laws which the symbols of algebra must obey is an impossible task, and reminds one not a little of the task of those philosophers who attempt to give an inventory of the *a priori* knowledge of the mind.

Trigonometry and double algebra

De Morgan's work entitled *Trigonometry and Double Algebra* consists of two parts; the former of which is a treatise on trigonometry, and the latter a treatise on generalized algebra which he called "double algebra. The first stage in the development of algebra is *arithmetic*, where numbers only appear and symbols of operations such as $+$, \times , etc. The next stage is *universal arithmetic*, where letters appear instead of numbers, so as to denote numbers universally, and the processes are conducted without knowing the values of the symbols. Let a and b denote any numbers; then such an expression as $a - b$ may be impossible; so that in universal arithmetic there is always a proviso, *provided the operation is possible*. The third stage is *single algebra*, where the symbol may denote a quantity forwards or a quantity backwards, and is adequately represented by segments on a straight line passing through an origin. Negative quantities are then no longer impossible; they are represented by the backward segment. But an impossibility still remains in the latter part of such an expression as $a + b\sqrt{-1}$ which arises in the solution of the quadratic equation. The fourth stage is *double algebra*. The algebraic symbol denotes in general a segment of a line in a given plane. It is a double symbol because it involves two specifications, namely, length, and direction; and $\sqrt{-1}$ is interpreted as denoting a quadrant. The expression $a + b\sqrt{-1}$ then represents a line in the plane having an abscissa a and an ordinate b . Argand and Warren carried double algebra so far - but they were unable to interpret on this theory such an expression as $e^{a\sqrt{-1}}$. De Morgan attempted it by *reducing* such an expression to the form $b + q\sqrt{-1}$, and he considered that he had shown that it could be always so reduced. The remarkable fact is that this double algebra satisfies all the fundamental laws above enumerated, and as every apparently impossible combination of symbols has been interpreted it looks like the complete form of algebra. In chapter 6 he introduced hyperbolic functions and discussed the connection of common and hyperbolic trigonometry.
 If the above theory is true, the next stage of development ought to be *triple algebra* and if $a + b\sqrt{-1}$ truly represents a line in a given plane, it ought to be possible to find a third term which added to the above would represent a line in space. Argand and some others guessed that it was $a + b\sqrt{-1} + c\sqrt{-1}\sqrt{-1}$ although this contradicts the truth established by Euler that $\sqrt{-1}\sqrt{-1} = e^{-\frac{1}{2}\pi}$. De Morgan and many others worked hard at the problem, but nothing came of it until the problem was taken up by Hamilton. We now see the reason clearly: The symbol of double algebra denotes not a length and a direction; but a multiplier and *an angle*. In it the angles are confined to one plane. Hence the next stage will be a *quadruple algebra*, when the axis of the plane is made variable. And this gives the answer to the first question; double algebra is nothing but analytical plane trigonometry, and this is why it has been found to be the natural analysis for alternating currents. But De Morgan never got this far. He died with the belief "that double algebra must remain as the full development of the conceptions of arithmetic, so far as those symbols are concerned which arithmetic immediately suggests".

When the study of mathematics revived at the University of Cambridge, so did the study of logic. The moving spirit was Whewell, the Master of Trinity College, whose principal writings were a *History of the Inductive Sciences*, and *Philosophy of the Inductive Sciences*. Doubtless De Morgan was influenced in his logical investigations by Whewell; but other influential contemporaries were Sir William Rowan Hamilton of Edinburgh, and Professor Boole of Cork. De Morgan's work on *Formal Logic*, published in 1847, is principally remarkable for his development of the numerically definite syllogism. The followers of Aristotle say that from two particular propositions such as *Some M's are A's*, and *Some M's are B's* nothing follows of necessity about the relation of the A's and B's. But they go further and say in order that any relation about the A's and B's may follow of necessity, the middle term must be taken universally in one of the premises. De Morgan pointed out that from *Most M's are A's and Most M's are B's* it follows of necessity that *some A's are B's* and he formulated the numerically definite syllogism which puts this principle in exact quantitative form. Suppose that the number of the M's is m , of the M's that are A's is a , and of the M's that are B's is b ; then there are at least $(a + b - m)$ A's that are B's. Suppose that the number of souls on board a steamer was 1000, that 500 were in the saloon, and 700 were lost. It follows of necessity, that at least $700 + 500 - 1000$, that is, 200, saloon passengers were lost. This single principle suffices to prove the validity of all the Aristotelian moods. It is therefore a fundamental principle in necessary reasoning.

Here then De Morgan had made a great advance by introducing *quantification of the terms*. At that time Sir William Rowan Hamilton was teaching in Edinburgh a doctrine of the quantification of the predicate, and a correspondence sprang up. However, De Morgan soon perceived that Hamilton's quantification was of a different character; that it meant for example, substituting the two forms *The whole of A is the whole of B*, and *The whole of A is a part of B* for the Aristotelian form *All A's are B's*. Hamilton thought that he had placed the keystone in the Aristotelian arch, as he phrased it. Although it must have been a curious arch which could stand 2000 years without a keystone. As a consequence he had no room for De Morgan's innovations. He accused De Morgan of plagiarism, and the controversy raged for years in the columns of the *Athenaeum*, and in the publications of the two writers.

The memoirs on logic which De Morgan contributed to the Transactions of the Cambridge Philosophical Society subsequent to the publication of his book on *Formal Logic* are by far the most important contributions which he made to the science, especially his fourth memoir, in which he begins work in the broad field of the *logic of relatives*. This is the true field for the logician of the twentieth century, in which work of the greatest importance is to be done towards improving language and facilitating thinking processes which occur all the time in practical life. Identity and difference are the two relations which have been considered by the logician; but there are many others equally deserving of study, such as equality, equivalence, consanguinity, affinity, etc.

In the introduction to the *Budget of Paradoxes* De Morgan explains what he means by the word.

A great many individuals, ever since the rise of the mathematical method, have, each for himself, attacked its direct and indirect consequences. I shall call each of these persons a *paradoixer*, and his system a *paradox*. I use the word in the old sense: a paradox is something which is apart from general opinion, either in subject matter, method, or conclusion. Many of the things brought forward would now be called *crotchets*, which is the nearest word we have to old *paradox*. But there is this difference, that by calling a thing a crotchet we mean to speak lightly of it; which was not the necessary sense of paradox. Thus in the 16th century many spoke of the earth's motion as the *paradox of Copernicus* and held the ingenuity of that theory in very high esteem, and some I think who even inclined towards it. In the seventeenth century the deprivation of meaning took place, in England at least.

How can the sound paradoixer be distinguished from the false paradoixer? De Morgan supplies the following test:

The manner in which a paradoixer will show himself, as to sense or nonsense, will not depend upon what he maintains, but upon whether he has or has not made a sufficient knowledge of what has been done by others, especially as to the mode of doing it, a preliminary to inventing knowledge for himself... New knowledge, when to any purpose, must come by contemplation of old knowledge, in every matter which concerns thought; mechanical contrivance sometimes, not very often, escapes this rule. All the men who are now called

discoverers, in every matter ruled by thought, have been men versed in the minds of their predecessors and learned in what had been before them. There is not one exception.

I remember that just before the American Association met at Indianapolis in 1890, the local newspapers heralded a great discovery which was to be laid before the assembled savants — a young man living somewhere in the country had squared the circle. While the meeting was in progress I observed a young man going about with a roll of paper in his hand. He spoke to me, and complained that the paper containing his discovery had not been received. I asked him whether his object in presenting the paper was not to get it read, printed, and published so that everyone might inform himself of the result; to all of which he assented readily. But, said I, many men have worked at this question, and their results have been tested fully, and they are printed for the benefit of anyone who can read; have you informed yourself of their results? To this there was no assent, but the sickly smile of the false paradoxer.

The *Budget* consists of a review of a large collection of paradoxical books which De Morgan had accumulated in his own library, partly by purchase at bookstands, partly from books sent to him for review, partly from books sent to him by the authors. He gives the following classification: squarers of the circle, trisectors of the angle, duplicators of the cube, constructors of perpetual motion, subverters of gravitation, stagnators of the earth, builders of the universe. You will still find specimens of all these classes in the New World and in the new century. De Morgan gives his personal knowledge of paradoxers.

I suspect that I know more of the English class than any man in Britain. I never kept any reckoning: but I know that one year with another? — and less of late years than in earlier time? — I have talked to more than five in each year, giving more than a hundred and fifty specimens. Of this I am sure, that it is my own fault if they have not been a thousand. Nobody knows how they swarm, except those to whom they naturally resort. They are in all ranks and occupations, of all ages and characters. They are very earnest people, and their purpose is *bona fide*, the dissemination of their paradoxes. A great many — the mass, indeed — are illiterate, and a great many waste their means, and are in or approaching penury. These discoverers despise one another.

A paradoxer to whom De Morgan paid the compliment which Achilles paid Hector — to drag him round the walls again and again — was James Smith, a successful merchant of Liverpool. He found $\pi = 3\frac{1}{8}$. His mode of reasoning was a curious caricature of the *reductio ad absurdum* of Euclid. He said let $\pi = 3\frac{1}{8}$, and then showed that on that supposition, every other value of π must be absurd. Consequently $\pi = 3\frac{1}{8}$ is the true value. The following is a specimen of De Morgan's dragging round the walls of Troy:

Mr. Smith continues to write me long letters, to which he hints that I am to answer. In his last of 31 closely written sides of note paper, he informs me, with reference to my obstinate silence, that though I think myself and am thought by others to be a mathematical Goliath, I have resolved to play the mathematical snail, and keep within my shell. A mathematical *snail!* This cannot be the thing so called which regulates the striking of a clock; for it would mean that I am to make Mr. Smith sound the true time of day, which I would by no means undertake upon a clock that gains 19 seconds odd in every hour by false quadrative value of π . But he ventures to tell me that pebbles from the sling of simple truth and common sense will ultimately crack my shell, and put me *hors de combat*. The confusion of images is amusing: Goliath turning himself into a snail to avoid $\pi = 3\frac{1}{8}$ and James Smith, Esq., of the Mersey Dock Board: and put *hors de combat* by pebbles from a sling. If Goliath had crept into a snail shell, David would have cracked the Philistine with his foot. There is something like modesty in the implication that the crack-shell pebble has not yet taken effect; it might have been thought that the slinger would by this time have been singing — And thrice [and one-eighth] I routed all my foes, And thrice [and one-eighth] I slew the slain.

In the region of pure mathematics, De Morgan could detect easily the false from the true paradox; but he was not so proficient in the field of physics. His father-in-law was a paradoxer, and his wife a paradoxer; and in the opinion of the physical philosophers De Morgan himself scarcely escaped. His wife wrote a book describing the phenomena of

spiritualism, table-rapping, table-turning, etc.; and De Morgan wrote a preface in which he said that he knew some of the asserted facts, believed others on testimony, but did not pretend to know *whether* they were caused by spirits, or had some unknown and unimagined origin. From this alternative he left out ordinary material causes. Faraday delivered a lecture on *Spiritualism*, in which he laid it down that in the investigation we ought to set out with the idea of what is physically possible, or impossible; De Morgan did not believe this.

Relations

De Morgan discovered relation algebra in his *Syllabus of a Proposed System of Logic* (1966: 208-46), first published in 1860. This algebra was extended by Charles Sanders Peirce (who admired De Morgan and met him shortly before his death), and re-exposed and further extended in vol. 3 of Ernst Schröder's *Vorlesungen über die Algebra der Logik*. Relation algebra proved critical to the *Principia Mathematica* of Bertrand Russell and Alfred North Whitehead. In turn, this algebra became the subject of much further work, starting in 1940, by Alfred Tarski and his colleagues and students at the University of California.

Spiritualism

De Morgan later in his life became interested in the phenomena of Spiritualism. In 1849 he had investigated clairvoyance and was impressed by the subject. He later carried out paranormal investigations in his own home with the medium Maria Hayden. The result of these investigations was later published by his wife Sophia. De Morgan believed that his career as a scientist might have been affected if he had revealed his interest in the study of spiritualism so he helped to publish the book anonymously.^[3] The book was published in 1863 titled *From Matter to Spirit: The Result of Ten Years Experience in Spirit Manifestations*.

According to (Oppenheim, 1988) De Morgan's wife Sophia was a convinced spiritualist but De Morgan shared a third way position on spiritualist phenomena which Oppenheim defined as a "wait-and-see position", he was neither a believer or a skeptic, instead his viewpoint was that the methodology of the physical sciences does not automatically exclude psychic phenomena and that such phenomena may be explainable in time by the possible existence of natural forces which as yet physicists had not identified.^[4]

In the preface of *From Matter to Spirit* (1863) De Morgan stated:

Thinking it very likely that the universe may contain a few agencies—say half a million—about which no man knows anything, I can not but suspect that a small proportion of these agencies—say five thousand—may be severally competent to the production of all the [spiritualist] phenomena, or may be quite up to the task among them. The physical explanations which I have seen are easy, but miserably insufficient: the spiritualist hypothesis is sufficient, but ponderously difficult. Time and thought will decide, the second asking the first for more results of trial.

John Beloff in *Parapsychology: A Concise History* (1997) wrote that De Morgan was the first notable scientist in Britain to take an interest in the study of spiritualism and his studies had influenced the decision of William Crookes to also study spiritualism. De Morgan was also an atheist and because of this had debarred him from a position at Oxford and Cambridge.^[5]

Legacy

Beyond his great mathematical legacy, the headquarters of the London Mathematical Society is called *De Morgan House* and the student society of the Mathematics Department of University College London is called the August De Morgan Society.

The crater De Morgan on the Moon is named after him.

Selected writings

- 1836. *An Explanation of the Gnomonic Projection of the Sphere*^[6]. London: Baldwin.
- 1837. *Elements of Trigonometry, and Trigonometrical Analysis*^[7]. London: Taylor & Walton.
- 1837. *The Elements of Algebra*^[8]. London: Taylor & Walton.
- 1838. *An Essay on Probabilities*^[9]. London: Longman, Orme, Brown, Green & Longmans.
- 1840. *The Elements of Arithmetic*^[10]. London: Taylor & Walton.
- 1840. *First Notions of Logic, Preparatory to the Study of Geometry*.^[11] London: Taylor & Walton.
- 1842. *The Differential and Integral Calculus*^[12]. London: Baldwin.
- 1845. *The Globes, Celestial and Terrestrial*^[13]. London: Malby & Co.
- 1847. *Formal Logic or The Calculus of Inference*^[14]. London: Taylor & Walton.
- 1849. *Trigonometry and Double Algebra*^[15]. London: Taylor, Walton & Malbery.
- 1860. *Syllabus of a Proposed System of Logic*^[16]. London: Walton & Malbery.
- 1872. *A Budget of Paradoxes*^[17]. London: Longmans, Green.

References and notes

- [1] De Morgan, (1838) *Induction (mathematics), The Penny Cyclopaedia*.
- [2] The year of his birth may be found by solving a conundrum proposed by himself, "I was UNIQ-math-0-a68bc4e06c66b481-QINU years of age in the year UNIQ-math-1-a68bc4e06c66b481-QINU " (He was 43 in 1849). The problem is indeterminate, but it is made strictly determinate by the century of its utterance and the limit to a man's life. Those born in 1722 (1764-42), 1892 (1936-44) and 1980 (2025-45) are similarly privileged.
- [3] Geoffrey K. Nelson *Spiritualism and Society* 1969, p. 90
- [4] Janet Oppenheim *The Other World: Spiritualism and Psychical Research in England, 1850-1914* 1988 p. 335
- [5] John Beloff *Parapsychology: A Concise History* 1997, pp. 46-47
- [6] <http://books.google.com/books?id=GzMAAAAQAAJ>
- [7] <http://books.google.com/books?id=Kk8EAAAQAAJ>
- [8] <http://books.google.com/books?id=sRoPAAAAIAAJ>
- [9] <http://books.google.com/books?id=NtA3AAAAMAAJ>
- [10] <http://books.google.com/books?id=P0Vthoo21zgC>
- [11] <http://books.google.com/books?id=drcIAAAAQAAJ>
- [12] <http://books.google.com/books?id=mbw4AAAAMAAJ>
- [13] <http://books.google.com/books?id=zPYDAAAQAAJ>
- [14] <http://books.google.com/books?id=HscAAAAAMAAJ>
- [15] <http://books.google.com/books?id=7UwEAAAQAAJ>
- [16] <http://books.google.com/books?id=Od3jf5fZtgC>
- [17] <http://www.gutenberg.org/etext/23100>

Further reading

- De Morgan, A., 1966. *Logic: On the Syllogism and Other Logical Writings*. Heath, P., ed. Routledge. A useful collection of De Morgan's most important writings on logic.
- De Morgan, Sophia Elizabeth (1882). *Memoir of Augustus De Morgan* (<http://books.google.com/?id=YiGJG0tbMHcC&printsec=frontcover&dq=Augustus+De+Morgan>). London: Longmans, Green and Company.
- Grattan-Guinness, Ivor (2000). *The Search for Mathematical Roots 1870-1940*. Princeton University Press.
- Macfarlane, Alexander (1916). *Lectures on Ten British Mathematicians of the Nineteenth Century* (<http://www.gutenberg.net/etext06/tbmms10p.pdf>). New York: John Wiley and Sons.
- Merrill, Daniel D. (1990). *Augustus De Morgan and the Logic of Relations*. Dordrecht: Kluwer Academic Publishers.

External links

- O'Connor, John J.; Robertson, Edmund F., "Augustus De Morgan" (http://www-history.mcs.st-andrews.ac.uk/Biographies/De_Morgan.html), *MacTutor History of Mathematics archive*, University of St Andrews.
- Papers of Augustus De Morgan held by Senate House Library, University of London (<http://archives.ulrls.lon.ac.uk/dispatcher.aspx?action=search&database=ChoiceArchive&search=IN=MS913A>)
- Library of Augustus De Morgan (<http://www.shl.lon.ac.uk/specialcollections/demorgan.shtml>)
- Archival material relating to Augustus De Morgan (<http://www.nationalarchives.gov.uk/nra/searches/subjectView.asp?ID=P7831>) listed at the UK National Archives
- Portraits of Augustus De Morgan (<http://www.npg.org.uk/collections/search/person.php?LinkID=mp83094>) at the National Portrait Gallery, London
- Project Gutenberg eBook On the Study and Difficulties of Mathematics (<http://www.gutenberg.org/files/39088/39088-pdf.pdf>)

Charles Sanders Peirce

Charles Sanders Peirce



Charles Sanders Peirce

Born	September 10, 1839 in Cambridge, Massachusetts
Died	April 19, 1914 (aged 74) in Milford, Pennsylvania
Nationality	American
Fields	Logic, Mathematics, Statistics, ^[1] ^[2] Philosophy, Metrology, ^[3] Chemistry, ^[4] Experimental psychology, ^[5] Economics, ^[6] Linguistics, History of science
Religious stance	Episcopal but unconventional

Charles Sanders Peirce

General

- Charles Sanders Peirce
- Charles Sanders Peirce bibliography

Philosophical

- Categories (Peirce)
- Semiotic elements and classes of signs
- Pragmatic maxim
- Pragmaticism
- Synechism
- Tychism
- Classification of the sciences

Biographical

- Juliette Peirce
- Charles Santiago Sanders Peirce

Charles Sanders Peirce (/pɜːrs/,^[7] like "purse", September 10, 1839 – April 19, 1914) was an American philosopher, logician, mathematician, and scientist, sometimes known as "the father of pragmatism". He was educated as a chemist and employed as a scientist for 30 years. Today he is appreciated largely for his contributions to logic, mathematics, philosophy, scientific methodology, and semiotics, and for his founding of pragmatism.

In 1934, the philosopher Paul Weiss called Peirce "the most original and versatile of American philosophers and America's greatest logician".^[8] Webster's Biographical Dictionary said in 1943 that Peirce was "now regarded as the most original thinker and greatest logician of his time."^[9]

An innovator in mathematics, statistics, philosophy, research methodology, and various sciences, Peirce considered himself, first and foremost, a logician. He made major contributions to logic, but logic for him encompassed much of that which is now called epistemology and philosophy of science. He saw logic as the formal branch of semiotics, of which he is a founder. As early as 1886 he saw that logical operations could be carried out by electrical switching circuits; the same idea was used decades later to produce digital computers.^[10]

Life



Peirce's birthplace. Now Lesley University's Graduate School of Arts and Social Sciences

Peirce was born at 3 Phillips Place in Cambridge, Massachusetts. He was the son of Sarah Hunt Mills and Benjamin Peirce, himself a professor of astronomy and mathematics at Harvard University and perhaps the first serious research mathematician in America. At age 12, Charles read his older brother's copy of Richard Whately's *Elements of Logic*, then the leading English-language text on the subject. So began his lifelong fascination with logic and reasoning.^[11]

He went on to earn the B.A. and M.A. from Harvard; in 1863 the Lawrence Scientific School awarded him a B.Sc. that was Harvard's first *summa cum laude* chemistry degree;^[12] and otherwise his academic record was

undistinguished.^[13] At Harvard, he began lifelong friendships with Francis Ellingwood Abbot, Chauncey Wright, and William James.^[14] One of his Harvard instructors, Charles William Eliot, formed an unfavorable opinion of Peirce. This opinion proved fateful, because Eliot, while President of Harvard 1869–1909—a period encompassing nearly all of Peirce's working life—repeatedly vetoed Harvard's employing Peirce in any capacity.^[15]

Peirce suffered from his late teens onward from a nervous condition then known as "facial neuralgia", which would today be diagnosed as trigeminal neuralgia. Brent says that when in the throes of its pain "he was, at first, almost stupefied, and then aloof, cold, depressed, extremely suspicious, impatient of the slightest crossing, and subject to violent outbursts of temper".^[16] Its consequences may have led to the social isolation which made his life's later years so tragic.

Early employment

Between 1859 and 1891, Peirce was intermittently employed in various scientific capacities by the United States Coast Survey,^[17] where he enjoyed his highly influential father's protection^[18] until the latter's death in 1880. That employment exempted Peirce from having to take part in the Civil War; it would have been very awkward for him to do so, as the Boston Brahmin Peirces sympathized with the Confederacy.^[19] At the Survey, he worked mainly in geodesy and gravimetry, refining the use of pendulums to determine small local variations in the Earth's gravity. He

was elected a resident fellow of the American Academy of Arts and Sciences in January 1867.^[20] The Survey sent him to Europe five times,^[21] first in 1871 as part of a group sent to observe a solar eclipse; there, he sought out Augustus De Morgan, William Stanley Jevons, and William Kingdon Clifford,^[22] British mathematicians and logicians whose turn of mind resembled his own. From 1869 to 1872, he was employed as an Assistant in Harvard's astronomical observatory, doing important work on determining the brightness of stars and the shape of the Milky Way.^[23] On April 20, 1877 he was elected a member of the National Academy of Sciences.^[24] Also in 1877, he proposed measuring the meter as so many wavelengths of light of a certain frequency,^[25] the kind of definition employed from 1860 to 1883.

During the 1880s, Peirce's indifference to bureaucratic detail waxed while his Survey work's quality and timeliness waned. Peirce took years to write reports that he should have completed in months. Wikipedia:Avoid weasel words Meanwhile, he wrote entries, ultimately thousands during 1883–1909, on philosophy, logic, science, and other subjects for the encyclopedic *Century Dictionary*.^[26] In 1885, an investigation by the Allison Commission exonerated Peirce, but led to the dismissal of Superintendent Julius Hilgard and several other Coast Survey employees for misuse of public funds.^[27] In 1891, Peirce resigned from the Coast Survey at Superintendent Thomas Corwin Mendenhall's request.^[28] He never again held regular employment.

Johns Hopkins University

In 1879, Peirce was appointed Lecturer in logic at the new Johns Hopkins University, which was strong in a number of areas that interested him, such as philosophy (Royce and Dewey did their PhDs at Hopkins), psychology (taught by G. Stanley Hall and studied by Joseph Jastrow, who coauthored a landmark empirical study with Peirce), and mathematics (taught by J. J. Sylvester, who came to admire Peirce's work on mathematics and logic). 1883 saw publication of his *Studies in Logic by Members of the Johns Hopkins University* containing works by himself and Allan Marquand, Christine Ladd, Benjamin Ives Gilman, and Oscar Howard Mitchell. They were among his graduate students.^[29] This nontenured position proved to be the only academic appointment Peirce ever held.

Brent documents something Peirce never suspected, namely that his efforts to obtain academic employment, grants, and scientific respectability were repeatedly frustrated by the covert opposition of a major Canadian-American scientist of the day, Simon Newcomb.^[30] Peirce's efforts may also have been hampered by a difficult personality; Brent conjectures as to further psychological difficulty.^[31]

Peirce's personal life also handicapped him. His first wife, Harriet Melusina Fay ("Zina"), left him in 1875.^[32] He soon took up with a woman, Juliette, whose maiden name, given variously as Froissy and Pourtalai^[33] and nationality (she spoke French^[34]) remain uncertain,^[35] but his divorce from Zina became final only in 1883, whereupon he married Juliette.^[36] That year, Newcomb pointed out to a Johns Hopkins trustee that Peirce, while a Hopkins employee, had lived and traveled with a woman to whom he was not married; the ensuing scandal led to his dismissal in January 1884.^[37] Over the years Peirce sought academic employment at various universities without success.^[38] He had no children by either marriage.^[39]

Poverty

In 1887 Peirce spent part of his inheritance from his parents to buy 2,000 acres (8 km^2) of rural land near Milford, Pennsylvania, which never yielded an economic return.^[40] There he had an 1854 farmhouse remodeled to his design.^[41] The Peirces named the property "Arisbe". There they lived with few interruptions for the rest of their lives,^[42] Charles writing prolifically, much of it unpublished to this day (see Works). Living beyond their means soon led to grave financial and legal difficulties.^[43]

He spent much of his last two decades unable to afford heat in winter and subsisting on old bread donated by the local baker. Unable to afford new stationery, he wrote on the verso side of old manuscripts. An outstanding warrant for assault and unpaid debts led to his being a fugitive in New York City for a while.^[44] Several people, including his brother James Mills Peirce^[45] and his neighbors, relatives of Gifford Pinchot, settled his debts and paid his property taxes and mortgage.^[46]

Peirce did some scientific and engineering consulting and wrote much for meager pay, mainly encyclopedic dictionary entries, and reviews for *The Nation* (with whose editor, Wendell Phillips Garrison, he became friendly). He did translations for the Smithsonian Institution, at its director Samuel Langley's instigation. Peirce also did substantial mathematical calculations for Langley's research on powered flight. Hoping to make money, Peirce tried inventing.^[47] He began but did not complete a number of books.^[48] In 1888, President Grover Cleveland appointed him to the Assay Commission.^[49]



Cambridge, where Peirce was born and raised, New York City, where he often visited and sometimes lived, and Milford, where he spent the later years of his life with his second wife Juliette.



Juliette and Charles by a well at their home Arisbe in 1907



Arisbe in 2011

From 1890 on, he had a friend and admirer in Judge Francis C. Russell of Chicago,^[50] who introduced Peirce to editor Paul Carus and owner Edward C. Hegeler of the pioneering American philosophy journal *The Monist*, which eventually published at least 14 articles by Peirce.^[51] He wrote many texts in James Mark Baldwin's *Dictionary of Philosophy and Psychology* (1901–5); half of those credited to him appear to have been written actually by Christine Ladd-Franklin under his supervision.^[52] He applied in 1902 to

the newly formed Carnegie Institution for a grant to write a systematic book of his life's work. The application was doomed; his nemesis Newcomb served on the Institution's executive committee, and its President had been the President of Johns Hopkins at the time of Peirce's dismissal.^[53]

The one who did the most to help Peirce in these desperate times was his old friend William James, dedicating his *Will to Believe* (1897) to Peirce, and arranging for Peirce to be paid to give two series of lectures at or near Harvard (1898 and 1903).^[54] Most important, each year from 1907 until James's death in 1910, James wrote to his friends in the Boston intelligentsia to request financial aid for Peirce; the fund continued even after James died. Peirce reciprocated by designating James's eldest son as his heir should Juliette predecease him.^[55] It has been believed that this was also why Peirce used "Santiago" ("St. James" in English) as a middle name, but he appeared in print as early as 1890 as Charles Santiago Peirce. (See Charles Santiago Sanders Peirce for discussion and references).

Peirce died destitute in Milford, Pennsylvania, twenty years before his widow.

Reception

Bertrand Russell (1959) wrote,^[56] "Beyond doubt [...] he was one of the most original minds of the later nineteenth century, and certainly the greatest American thinker ever." (Russell and Whitehead's *Principia Mathematica*, published from 1910 to 1913, does not mention Peirce; Peirce's work was not widely known till later.)^[57] A. N. Whitehead, while reading some of Peirce's unpublished manuscripts soon after arriving at Harvard in 1924, was struck by how Peirce had anticipated his own "process" thinking. (On Peirce and process metaphysics, see Lowe 1964.) Karl Popper viewed Peirce as "one of the greatest philosophers of all times".^[58] Yet Peirce's achievements were not immediately recognized. His imposing contemporaries William James and Josiah Royce^[59] admired him, and Cassius Jackson Keyser at Columbia and C. K. Ogden wrote about Peirce with respect, but to no immediate effect.

The first scholar to give Peirce his considered professional attention was Royce's student Morris Raphael Cohen, the editor of an anthology of Peirce's writings titled *Chance, Love, and Logic* (1923) and the author of the first bibliography of Peirce's scattered writings.^[60] John Dewey studied under Peirce at Johns Hopkins and, from 1916 onwards, Dewey's writings repeatedly mention Peirce with deference. His 1938 *Logic: The Theory of Inquiry* is much influenced by Peirce.^[61] The publication of the first six volumes of the *Collected Papers* (1931–35), the most important event to date in Peirce studies and one that Cohen made possible by raising the needed funds,^[62] did not prompt an outpouring of secondary studies. The editors of those volumes, Charles Hartshorne and Paul Weiss, did not become Peirce specialists. Early landmarks of the secondary literature include the monographs by Buchler (1939), Feibleman (1946), and Goudge (1950), the 1941 Ph.D. thesis by Arthur W. Burks (who went on to edit volumes 7 and 8), and the studies edited by Wiener and Young (1952). The Charles S. Peirce Society was founded in 1946. Its *Transactions*, an academic quarterly specializing in Peirce, pragmatism, and American philosophy, has appeared since 1965.

In 1949, while doing unrelated archival work, the historian of mathematics Carolyn Eisele (1902–2000) chanced on an autograph letter by Peirce. So began her 40 years of research on Peirce the mathematician and scientist, culminating in Eisele (1976, 1979, 1985). Beginning around 1960, the philosopher and historian of ideas Max Fisch (1900–1995) emerged as an authority on Peirce; Fisch (1986)^[63] includes many of his relevant articles, including a wide-ranging survey (Fisch 1986: 422–48) of the impact of Peirce's thought through 1983.

Peirce has gained a significant international following, marked by university research centers devoted to Peirce studies and pragmatism in Brazil (CeneP/CIEP), Finland (HPRC, including Commens), Germany (Wirth's group, Hoffman's and Otte's group, and Deuser's and Härtle's group^[64]), France (L'I.R.S.C.E.), Spain (GEP), and Italy (CSP). His writings have been translated into several languages, including German, French, Finnish, Spanish, and Swedish. Since 1950, there have been French, Italian, Spanish, British, and Brazilian Peirceans of note. For many years, the North American philosophy department most devoted to Peirce was the University of Toronto's, thanks in good part to the leadership of Thomas Goudge and David Savan. In recent years, U.S. Peirce scholars have clustered

at Indiana University - Purdue University Indianapolis, home of the Peirce Edition Project (PEP), and the Pennsylvania State University.

Currently, considerable interest is being taken in Peirce's ideas by researchers wholly outside the arena of academic philosophy. The interest comes from industry, business, technology, intelligence organizations, and the military; and it has resulted in the existence of a substantial number of agencies, institutes, businesses, and laboratories in which ongoing research into and development of Peircean concepts are being vigorously undertaken.

—Robert Burch, 2001, updated 2010

Works

Peirce's reputation rests largely on a number of academic papers published in American scientific and scholarly journals such as *Proceedings of the American Academy of Arts and Sciences*, the *Journal of Speculative Philosophy*, *The Monist*, *Popular Science Monthly*, the *American Journal of Mathematics*, *Memoirs of the National Academy of Sciences*, *The Nation*, and others. See Articles by Peirce, published in his lifetime for an extensive list with links to them online. The only full-length book (neither extract nor pamphlet) that Peirce authored and saw published in his lifetime^[65] was *Photometric Researches* (1878), a 181-page monograph on the applications of spectrographic methods to astronomy. While at Johns Hopkins, he edited *Studies in Logic* (1883), containing chapters by himself and his graduate students. Besides lectures during his years (1879–1884) as Lecturer in Logic at Johns Hopkins, he gave at least nine series of lectures, many now published; see Lectures by Peirce.

Harvard University obtained from Peirce's widow soon after his death the papers found in his study, but did not microfilm them until 1964. Only after Richard Robin (1967)^[66] catalogued this *Nachlass* did it become clear that Peirce had left approximately 1650 unpublished manuscripts, totaling over 100,000 pages,^[67] mostly still unpublished except on microfilm. On the vicissitudes of Peirce's papers, see Houser (1989).^[68] Reportedly the papers remain in unsatisfactory condition.^[69]

The first published anthology of Peirce's articles was the one-volume *Chance, Love and Logic: Philosophical Essays*, edited by Morris Raphael Cohen, 1923, still in print. Other one-volume anthologies were published in 1940, 1957, 1958, 1972, 1994, and 2009, most still in print. The main posthumous editions^[70] of Peirce's works in their long trek to light, often multi-volume, and some still in print, have included:

1931–58: *Collected Papers of Charles Sanders Peirce* (CP), 8 volumes, includes many published works, along with a selection of previously unpublished work and a smattering of his correspondence. This long-time standard edition drawn from Peirce's work from the 1860s to 1913 remains the most comprehensive survey of his prolific output from 1893 to 1913. It is organized thematically, but texts (including lecture series) are often split up across volumes, while texts from various stages in Peirce's development are often combined, requiring frequent visits to editors' notes.^[71] Edited (1–6) by Charles Hartshorne and Paul Weiss and (7–8) by Arthur Burks, in print and online.

1975–87: *Charles Sanders Peirce: Contributions to The Nation*, 4 volumes, includes Peirce's more than 300 reviews and articles published 1869–1908 in *The Nation*. Edited by Kenneth Laine Ketner and James Edward Cook, online.

1976: *The New Elements of Mathematics* by Charles S. Peirce, 4 volumes in 5, included many previously unpublished Peirce manuscripts on mathematical subjects, along with Peirce's important published mathematical articles. Edited by Carolyn Eisele, out of print.

1977: *Semiotic and Significs: The Correspondence between C. S. Peirce and Victoria Lady Welby* (2nd edition 2001), included Peirce's entire correspondence (1903–1912) with Victoria, Lady Welby. Peirce's other published correspondence is largely limited to the 14 letters included in volume 8 of the *Collected Papers*, and the 20-odd pre-1890 items included so far in the *Writings*. Edited by Charles S. Hardwick with James Cook, out of print.

1982–now: *Writings of Charles S. Peirce, A Chronological Edition* (W), Volumes 1–6 & 8, of a projected 30. The limited coverage, and defective editing and organization, of the *Collected Papers* led Max Fisch and others in the

1970s to found the Peirce Edition Project (PEP), whose mission is to prepare a more complete critical chronological edition. Only seven volumes have appeared to date, but they cover the period from 1859–1892, when Peirce carried out much of his best-known work. W 8 was published in November 2010; and work continues on W 7, 9, and 11. In print and online.

1985: *Historical Perspectives on Peirce's Logic of Science: A History of Science*, 2 volumes. Auspitz has said,^[72] "The extent of Peirce's immersion in the science of his day is evident in his reviews in the *Nation* [...] and in his papers, grant applications, and publishers' prospectuses in the history and practice of science", referring latterly to *Historical Perspectives*. Edited by Carolyn Eisele, out of print.

1992: *Reasoning and the Logic of Things* collects in one place Peirce's 1898 series of lectures invited by William James. Edited by Kenneth Laine Ketner, with commentary by Hilary Putnam, in print.

1992–98: *The Essential Peirce* (EP), 2 volumes, is an important recent sampler of Peirce's philosophical writings. Edited (1) by Nathan Hauser and Christian Kloesel and (2) by PEP editors, in print.

1997: *Pragmatism as a Principle and Method of Right Thinking* collects Peirce's 1903 Harvard "Lectures on Pragmatism" in a study edition, including drafts, of Peirce's lecture manuscripts, which had been previously published in abridged form; the lectures now also appear in EP 2. Edited by Patricia Ann Turisi, in print.

2010: *Philosophy of Mathematics: Selected Writings* collects important writings by Peirce on the subject, many not previously in print. Edited by Matthew E. Moore, in print.

Mathematics

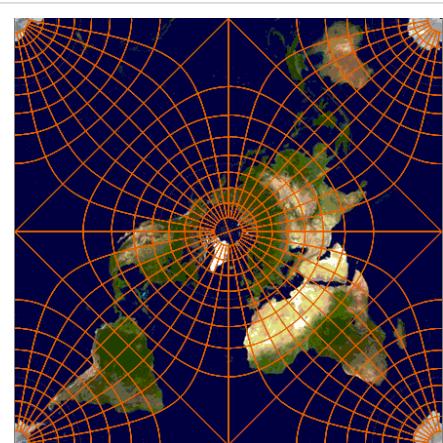
Peirce's most important work in pure mathematics was in logical and foundational areas. He also worked on linear algebra, matrices, various geometries, topology and Listing numbers, Bell numbers, graphs, the four-color problem, and the nature of continuity.

He worked on applied mathematics in economics, engineering, and map projections (such as the Peirce quincuncial projection), and was especially active in probability and statistics.^[73]

Discoveries

Peirce made a number of striking discoveries in formal logic and foundational mathematics, nearly all of which came to be appreciated only long after he died:

In 1860^[74] he suggested a cardinal arithmetic for infinite numbers, years before any work by Georg Cantor (who completed his dissertation in 1867) and without access to Bernard Bolzano's 1851 (posthumous) *Paradoxien des Unendlichen*.



The Peirce quincuncial projection of a sphere keeps angles true except at several isolated points and results in less distortion of area than in other projections.

↓
The Peirce arrow,
symbol for "(neither)...nor...", also called the Quine dagger.

In 1880–81^[75] he showed how Boolean algebra could be done via a repeated sufficient single binary operation (logical NOR), anticipating Henry M. Sheffer by 33 years. (See also De Morgan's Laws).

In 1881^[76] he set out the axiomatization of natural number arithmetic, a few years before Richard Dedekind and Giuseppe Peano. In the same paper Peirce gave, years before Dedekind, the first purely cardinal definition of a finite set in the sense now known as "Dedekind-finite", and implied by the same stroke an important formal definition of an infinite set (Dedekind-infinite), as a set that can be put into a one-to-one correspondence with one of its proper subsets.

In 1885^[77] he distinguished between first-order and second-order quantification.^[78] In the same paper he set out what can be read as the first (primitive) axiomatic set theory, anticipating Zermelo by about two decades (Brady 2000,^[79] pp. 132–3).

In 1886 he saw that Boolean calculations could be carried out via electrical switches, anticipating Claude Shannon by more than 50 years.

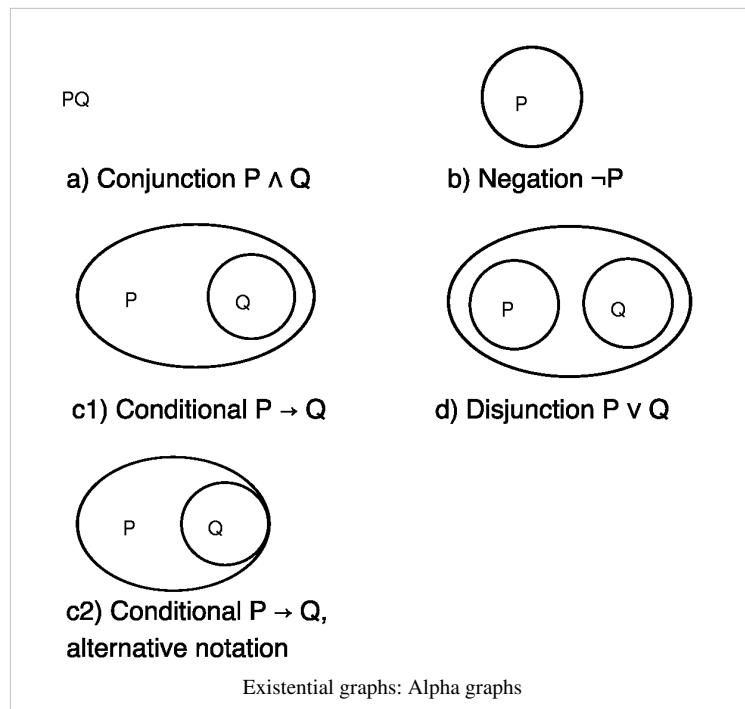
By the later 1890s^[80] he was devising existential graphs, a diagrammatic notation for the predicate calculus. Based on them are John F. Sowa's conceptual graphs and Sun-Joo Shin's diagrammatic reasoning.

The New Elements of Mathematics

Peirce wrote drafts for an introductory textbook, with the working title *The New Elements of Mathematics*, that presented mathematics from an original standpoint. Those drafts and many other of his previously unpublished mathematical manuscripts finally appeared in *The New Elements of Mathematics* by Charles S. Peirce (1976), edited by mathematician Carolyn Eisele.

Nature of mathematics

Peirce agreed with Auguste Comte in regarding mathematics as more basic than philosophy and the special sciences (of nature and mind). Peirce classified mathematics into three subareas: (1) mathematics of logic, (2) discrete series, and (3) pseudo-continua (as he called them, including the real numbers) and continua. Influenced by his father Benjamin, Peirce argued that mathematics studies purely hypothetical objects and is not just the science of quantity but is more broadly the science which draws necessary conclusions; that mathematics aids logic, not vice versa; and that logic itself is part of philosophy and is the science *about* drawing conclusions necessary and otherwise.^[81]



Mathematics of logic

Mathematical logic and foundations, some noted articles

- On an Improvement in Boole's Calculus of Logic (1867)
- Description of a Notation for the Logic of Relatives (1870)
- On the Algebra of Logic (1880)
- A Boolean Algebra with One Constant (1880 MS)
- On the Logic of Number (1881)
- Note B: The Logic of Relatives (1883)
- On the Algebra of Logic: A Contribution to the Philosophy of Notation (1884/1885)
- The Logic of Relatives (1897)
- The Simplest Mathematics (1902 MS)
- Prolegomena To an Apology For Pragmaticism (1906, on existential graphs)

Beginning with his first paper on the "Logic of Relatives" (1870), Peirce extended the theory of relations that Augustus De Morgan had just recently awakened from its Cinderella slumbers. Much of the mathematics of relations now taken for granted was "borrowed" from Peirce, not always with all due credit; on that and on how the young Bertrand Russell, especially his *Principles of Mathematics* and *Principia Mathematica*, did not do Peirce justice, see Anellis (1995). In 1918 the logician C. I. Lewis wrote, "The contributions of C.S. Peirce to symbolic logic are more numerous and varied than those of any other writer — at least in the nineteenth century."^[82] Beginning in 1940, Alfred Tarski and his students rediscovered aspects of Peirce's larger vision of relational logic, developing the perspective of relational algebra.

Relational logic gained applications. In mathematics, it influenced the abstract analysis of E. H. Moore and the lattice theory of Garrett Birkhoff. In computer science, the relational model for databases was developed with Peircean ideas in work of Edgar F. Codd, who was a doctoral student^[83] of Arthur W. Burks, a Peirce scholar. In economics, relational logic was used by Frank P. Ramsey, John von Neumann, and Paul Samuelson to study preferences and utility and by Kenneth J. Arrow in *Social Choice and Individual Values*, following Arrow's association with Tarski at City College of New York.

On Peirce and his contemporaries Ernst Schröder and Gottlob Frege, Hilary Putnam (1982)^[1] documented that Frege's work on the logic of quantifiers had little influence on his contemporaries, although it was published four years before the work of Peirce and his student Oscar Howard Mitchell. Putnam found that mathematicians and logicians learned about the logic of quantifiers through the independent work of Peirce and Mitchell, particularly through Peirce's "On the Algebra of Logic: A Contribution to the Philosophy of Notation" (1885), published in the premier American mathematical journal of the day, and cited by Peano and Schröder, among others, who ignored Frege. They also adopted and modified Peirce's notations, typographical variants of those now used. Peirce apparently was ignorant of Frege's work, despite their overlapping achievements in logic, philosophy of language, and the foundations of mathematics.

Peirce's work on formal logic had admirers besides Ernst Schröder:

- Philosophical algebraist William Kingdon Clifford^[84] and logician William Ernest Johnson, both British;
- The Polish school of logic and foundational mathematics, including Alfred Tarski;
- Arthur Prior, who praised and studied Peirce's logical work in a 1964 paper and in *Formal Logic* (saying on page 4 that Peirce "perhaps had a keener eye for essentials than any other logician before or since.").

A philosophy of logic, grounded in his categories and semiotic, can be extracted from Peirce's writings and, along with Peirce's logical work more generally, is exposed and defended in Hilary Putnam (1982); the Introduction in Nathan Houser *et al.* (1997);^[85] and Randall Dipert's chapter in Cheryl Misak (2004).^[86]

Continua

Continuity and synechism are central in Peirce's philosophy. He embraced infinitesimals and worked long on the mathematics of continua. He long held that the real numbers constitute a pseudo-continuum;^[87] that a true continuum is the real subject matter of *analysis situs* (topology); and that a true continuum of instants exceeds—and within any lapse of time has room for—any Aleph number (any infinite *multitude* as he called it) of instants.^[88]

In 1908 Peirce wrote that he found that a true continuum might have or lack such room. Jérôme Havenel (2008): "It is on May 26, 1908, that Peirce finally gave up his idea that in every continuum there is room for whatever collection of any multitude. From now on, there are different kinds of continua, which have different properties."^[89]

Probability and statistics

Peirce held that science achieves statistical probabilities, not certainties, and that spontaneity (absolute chance) is real (see Tychism on his view). Most of his statistical writings promote the frequency interpretation of probability (objective ratios of cases), and many of his writings express skepticism about (and criticize the use of) probability when such models are not based on objective randomization.^[90] Though Peirce was largely a frequentist, his possible world semantics introduced the "propensity" theory of probability before Karl Popper.^{[91][92]} Peirce (sometimes with Joseph Jastrow) investigated the probability judgments of experimental subjects, "perhaps the very first" elicitation and estimation of subjective probabilities in experimental psychology and (what came to be called) Bayesian statistics.

Peirce was one of the founders of statistics. He formulated modern statistics in "Illustrations of the Logic of Science" (1877–8) and "A Theory of Probable Inference" (1883). With a repeated measures design, he introduced blinded, controlled randomized experiments in 1884 (Hacking 1990:205) (before Ronald A. Fisher). He invented optimal design for experiments on gravity, in which he "corrected the means". He used correlation and smoothing. Peirce extended the work on outliers by Benjamin Peirce, his father. He introduced terms "confidence" and "likelihood" (before Jerzy Neyman and Fisher). (See Stephen Stigler's historical books and Ian Hacking 1990).

Philosophy

It is not sufficiently recognized that Peirce's career was that of a scientist, not a philosopher; and that during his lifetime he was known and valued chiefly as a scientist, only secondarily as a logician, and scarcely at all as a philosopher. Even his work in philosophy and logic will not be understood until this fact becomes a standing premise of Peircean studies.

—Max Fisch 1964, p. 486.

Peirce was a working scientist for 30 years, and arguably was a professional philosopher only during the five years he lectured at Johns Hopkins. He learned philosophy mainly by reading, each day, a few pages of Kant's *Critique of Pure Reason*, in the original German, while a Harvard undergraduate. His writings bear on a wide array of disciplines, including mathematics, logic, philosophy, statistics, astronomy, metrology, geodesy, experimental psychology, economics, linguistics, and the history and philosophy of science. This work has enjoyed renewed interest and approval, a revival inspired not only by his anticipations of recent scientific developments but also by his demonstration of how philosophy can be applied effectively to human problems.

Peirce's philosophy includes (see below in related sections) a pervasive three-category system, belief that truth is immutable and is both independent from actual opinion (fallibilism) and discoverable (no radical skepticism), logic as formal semiotic on signs, on arguments, and on inquiry's ways—including philosophical pragmatism (which he founded), critical common-sensism, and scientific method—and, in metaphysics: Scholastic realism, belief in God, freedom, and at least an attenuated immortality, objective idealism, and belief in the reality of continuity and of absolute chance, mechanical necessity, and creative love. In his work, fallibilism and pragmatism may seem to work somewhat like skepticism and positivism, respectively, in others' work. However, for Peirce, fallibilism is balanced by an anti-skepticism and is a basis for belief in the reality of absolute chance and of continuity,^[93] and pragmatism commits one to anti-nominalist belief in the reality of the general (CP 5.453–7).

For Peirce, First Philosophy, which he also called cenoscopy, is less basic than mathematics and more basic than the special sciences (of nature and mind). It studies positive phenomena in general, phenomena available to any person at any waking moment, and does not settle questions by resorting to special experiences.^[94] He divided such philosophy into (1) phenomenology (which he also called phaneroscopy or categorics), (2) normative sciences (esthetics, ethics, and logic), and (3) metaphysics; his views on them are discussed in order below.

Theory of categories

On May 14, 1867, the 27-year-old Peirce presented a paper entitled "On a New List of Categories" [95]^[95] to the American Academy of Arts and Sciences, which published it the following year. The paper outlined a theory of predication, involving three universal categories that Peirce developed in response to reading Aristotle, Kant, and Hegel, categories that Peirce applied throughout his work for the rest of his life. Peirce scholars generally regard the "New List" as foundational or breaking the ground for Peirce's "architectonic", his blueprint for a pragmatic philosophy. In the categories one will discern, concentrated, the pattern that one finds formed by the three grades of clearness in "How To Make Our Ideas Clear" (1878 paper foundational to pragmatism), and in numerous other trichotomies in his work.

"On a New List of Categories" is cast as a Kantian deduction; it is short but dense and difficult to summarize. The following table is compiled from that and later works.^[96] In 1893, Peirce restated most of it for a less advanced audience.^[97]

Peirce's Categories (technical name: the cenopythagorean categories)^[98]

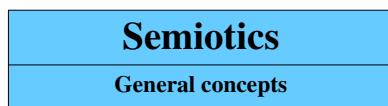
Name:	Typical characterizaton:	As universe of experience:	As quantity:	Technical definition:	Valence, "adicity":
Firstness. ^[99]	Quality of feeling.	Ideas, chance, possibility.	Vagueness, "some".	Reference to a ground (a ground is a pure abstraction of a quality). ^[100]	Essentially monadic (the quale, in the sense of the <i>such</i> , ^[101] which has the quality).
Secondness. ^[102]	Reaction, resistance, (dyadic) relation.	Brute facts, actuality.	Singularity, discreteness, "this".	Reference to a correlate (by its relate).	Essentially dyadic (the relate and the correlate).
Thirdness. ^[103]	Representation, mediation.	Habits, laws, necessity.	Generality, continuity, "all".	Reference to an interpretant*. ^[104]	Essentially triadic (sign, object, interpretant*).

*Note: An interpretant is an interpretation (human or otherwise) in the sense of the product of an interpretive process.

Esthetics and ethics

Peirce did not write extensively in esthetics and ethics,^[104] but came by 1902 to hold that esthetics, ethics, and logic, in that order, comprise the normative sciences.^[105] He characterized esthetics as the study of the good (grasped as the admirable), and thus of the ends governing all conduct and thought.^[106]

Philosophy: logic, or semiotic



Biosemiotics · Code
Cognitive semiotics
Computational semiotics
Confabulation
Connotation · Decode
Denotation · Encode · Lexical
Literary semiotics · Modality
Representation (arts) · Salience
Semiosis · Semiosphere
Semiotic elements & sign classes
Semiotics of culture
Sign · Sign relational complex
Sign relation · Umwelt · Value
Methods
Commutation test
Paradigmatic analysis
Syntagmatic analysis
Semioticians
Mikhail Bakhtin · Roland Barthes
Marcel Danesi · John Deely
Umberto Eco · Algirdas Julien Greimas
Félix Guattari · Louis Hjelmslev
Roman Jakobson · Roberta Kevelson
Kalevi Kull · Juri Lotman
Charles S. Peirce · Augusto Ponzio
Ferdinand de Saussure
Thomas Sebeok · Michael Silverstein
Eero Tarasti · Jakob von Uexküll
Vyacheslav Ivanov · Vladimir Toporov
Related topics
Structuralism
Post-structuralism
Tartu–Moscow Semiotic School
Aestheticization
Postmodernity

Logic as philosophical

Peirce regarded logic *per se* as a division of philosophy, as a normative science based on esthetics and ethics, as more basic than metaphysics,^[107] and as "the art of devising methods of research".^[108] More generally, as inference, "logic is rooted in the social principle", since inference depends on a standpoint that, in a sense, is unlimited.^[109] Peirce called (with no sense of depreciation) "mathematics of logic" much of the kind of thing which, in current research and applications, is called simply "logic". He was productive in both (philosophical) logic and logic's mathematics, which were connected deeply in his work and thought.

Peirce argued that logic is formal semiotic, the formal study of signs in the broadest sense, not only signs that are artificial, linguistic, or symbolic, but also signs that are semblances or are indexical such as reactions. Peirce held that "all this universe is perfused with signs, if it is not composed exclusively of signs",^[110] along with their representational and inferential relations. He argued that, since all thought takes time, all thought is in signs^[111] and sign processes ("semiosis") such as the inquiry process. He divided logic into: (1) speculative grammar, or stichiology, on how signs can be meaningful and, in relation to that, what kinds of signs there are, how they combine, and how some embody or incorporate others; (2) logical critic, or logic proper, on the modes of inference;

and (3) speculative or universal rhetoric, or methodeutic, the philosophical theory of inquiry, including pragmatism.

Presuppositions of logic

In his "F.R.L." [First Rule of Logic] (1899), Peirce states that the first, and "in one sense, the sole", rule of reason is that, *to learn, one needs to desire to learn* and desire it without resting satisfied with that which one is inclined to think. So, the first rule is, *to wonder*. Peirce proceeds to a critical theme in research practices and the shaping of theories:

...there follows one corollary which itself deserves to be inscribed upon every wall of the city of philosophy:

Do not block the way of inquiry.

Peirce adds, that method and economy are best in research but no outright sin inheres in trying any theory in the sense that the investigation via its trial adoption can proceed unimpeded and undiscouraged, and that "the one unpardonable offence" is a philosophical barricade against truth's advance, an offense to which "metaphysicians in all ages have shown themselves the most addicted". Peirce in many writings holds that logic precedes metaphysics (ontological, religious, and physical).

Peirce goes on to list four common barriers to inquiry: (1) Assertion of absolute certainty; (2) maintaining that something is absolutely unknowable; (3) maintaining that something is absolutely inexplicable because absolutely basic or ultimate; (4) holding that perfect exactitude is possible, especially such as to quite preclude unusual and anomalous phenomena. To refuse absolute theoretical certainty is the heart of *fallibilism*, which Peirce unfolds into refusals to set up any of the listed barriers. Peirce elsewhere argues (1897) that logic's presupposition of fallibilism leads at length to the view that chance and continuity are very real (tychism and synechism).

The First Rule of Logic pertains to the mind's presuppositions in undertaking reason and logic, presuppositions, for instance, that truth and the real do not depend on yours or my opinion of them but do depend on representational relation and consist in the destined end in investigation taken far enough (see below). He describes such ideas as, collectively, hopes which, in particular cases, one is unable seriously to doubt.^[112]

Four incapacities

The *Journal of Speculative Philosophy* series (1868–69), including

- Questions concerning certain Faculties claimed for Man (1868)
- Some Consequences of Four Incapacities (1868)
- Grounds of Validity of the Laws of Logic:

Further Consequences of Four Incapacities (1869)

In three articles in 1868–69,^{[113][114]} Peirce rejected mere verbal or hyperbolic doubt and first or ultimate principles, and argued that we have (as he numbered them):

1. No power of introspection. All knowledge of the internal world comes by hypothetical reasoning from known external facts.
2. No power of intuition (cognition without logical determination by previous cognitions). No cognitive stage is absolutely first in a process. All mental action has the form of inference.
3. No power of thinking without signs. A cognition must be interpreted in a subsequent cognition in order to be a cognition at all.
4. No conception of the absolutely incognizable.

(The above sense of the term "intuition" is almost Kant's, said Peirce. It differs from the current looser sense that encompasses instinctive or anyway half-conscious inference.)

Peirce argued that those incapacities imply the reality of the general and of the continuous, the validity of the modes of reasoning, and the falsity of philosophical Cartesianism (see below).

Peirce rejected the conception (usually ascribed to Kant) of the unknowable thing-in-itself and later said that to "dismiss make-believes" is a prerequisite for pragmatism.^[115]

Logic as formal semiotic

Peirce sought, through his wide-ranging studies through the decades, formal philosophical ways to articulate thought's processes, and also to explain the workings of science. These inextricably entangled questions of a dynamics of inquiry rooted in nature and nurture led him to develop his semiotic with very broadened conceptions of signs and inference, and, as its culmination, a theory of inquiry for the task of saying 'how science works' and devising research methods. This would be logic by the medieval definition taught for centuries: art of arts, science of sciences, having the way to the principles of all methods. Influences radiate from points on parallel lines of inquiry in Aristotle's work, in such *loci* as: the basic terminology of psychology in *On the Soul*; the founding description of sign relations in *On Interpretation*; and the differentiation of inference into three modes that are commonly translated into English as *abduction*, *deduction*, and *induction*, in the *Prior Analytics*, as well as inference by analogy (called *paradeigma* by Aristotle), which Peirce regarded as involving the other three modes.

Peirce began writing on semiotic in the 1860s, around the time when he devised his system of three categories. He called it both *semiotic* and *semeiotic*. Both are current in singular and plural. He based it on the conception of a triadic sign relation, and defined *semiosis* as "action, or influence, which is, or involves, a cooperation of *three* subjects, such as a sign, its object, and its interpretant, this tri-relative influence not being in any way resolvable into actions between pairs".^[116] As to signs in thought, Peirce emphasized the reverse:

To say, therefore, that thought cannot happen in an instant, but requires a time, is but another way of saying that every thought must be interpreted in another, or that all thought is in signs.

—Peirce 1868.

Peirce held that all thought is in signs, issuing in and from interpretation, where 'sign' is the word for the broadest variety of conceivable semblances, diagrams, metaphors, symptoms, signals, designations, symbols, texts, even mental concepts and ideas, all as determinations of a mind or *quasi-mind*, that which at least functions like a mind, as in the work of crystals or bees^[117] — the focus is on sign action in general rather than on psychology, linguistics, or social studies (fields which he also pursued).

Inquiry is a kind of inference process, a manner of thinking and semiosis. Global divisions of ways for phenomena to stand as signs, and the subsumption of inquiry and thinking within inference as a sign process, enable the study of inquiry on semiotics' three levels:

1. Conditions for meaningfulness. Study of significatory elements and combinations, their grammar.
2. Validity, conditions for true representation. Critique of arguments in their various separate modes.
3. Conditions for determining interpretations. Methodology of inquiry in its mutually interacting modes.

Peirce uses examples often from common experience, but defines and discusses such things as assertion and interpretation in terms of philosophical logic. In a formal vein, Peirce said:

On the Definition of Logic. Logic is *formal semiotic*. A sign is something, *A*, which brings something, *B*, its *interpretant* sign, determined or created by it, into the same sort of correspondence (or a lower implied sort) with something, *C*, its *object*, as that in which itself stands to *C*. This definition no more involves any reference to human thought than does the definition of a line as the place within which a particle lies during a lapse of time. It is from this definition that I deduce the principles of logic by mathematical reasoning, and by mathematical reasoning that, I aver, will support criticism of Weierstrassian severity, and that is perfectly evident. The word "formal" in the definition is also defined.

—Peirce, "Carnegie Application", *The New Elements of Mathematics* v. 4, p. 54.

Signs

A list of noted writings by Peirce on signs and sign relations is at Semiotic elements and classes of signs (Peirce)#References and further reading.

Sign relation

Anything is a sign — not absolutely as itself, but instead in some relation or other. The *sign relation* is the key. It defines three roles encompassing (1) the sign, (2) the sign's subject matter, called its *object*, and (3) the sign's meaning or ramification as formed into a kind of effect called its *interpretant* (a further sign, for example a translation). It is an irreducible *triadic relation*, according to Peirce. The roles are distinct even when the things that fill those roles are not. The roles are but three; a sign of an object leads to one or more interpretants, and, as signs, they lead to further interpretants.

Extension × intension = information. Two traditional approaches to sign relation, necessary though insufficient, are the way of *extension* (a sign's objects, also called breadth, denotation, or application) and the way of *intension* (the objects' characteristics, qualities, attributes referenced by the sign, also called depth, comprehension, significance, or connotation). Peirce adds a third, the way of *information*, including change of information, to integrate the other two approaches into a unified whole.^[118] For example, because of the equation above, if a term's total amount of information stays the same, then the more that the term 'intends' or signifies about objects, the fewer are the objects to which the term 'extends' or applies.

Determination. A sign depends on its object in such a way as to represent its object — the object enables and, in a sense, determines the sign. A physically causal sense of this stands out when a sign consists in an indicative reaction. The interpretant depends likewise on both the sign and the object — an object determines a sign to determine an interpretant. But this determination is not a succession of dyadic events, like a row of toppling dominoes; sign determination is triadic. For example, an interpretant does not merely represent something which represented an object; instead an interpretant represents something *as* a sign representing the object. The object (be it a quality or fact or law or even fictional) determines the sign to an interpretant through one's collateral experience^[119] with the object, in which the object is found or from which it is recalled, as when a sign consists in a chance semblance of an absent object. Peirce used the word "determine" not in a strictly deterministic sense, but in a sense of "specializes," *bestimmt*,^[120] involving variable amount, like an influence.^[121] Peirce came to define representation and interpretation in terms of (triadic) determination.^[122] The object determines the sign to determine another sign — the interpretant — to be related to the object *as the sign is related to the object*, hence the interpretant, fulfilling its function as sign of the object, determines a further interpretant sign. The process is logically structured to perpetuate itself, and is definitive of sign, object, and interpretant in general.

Semiotic elements

Peirce held there are exactly three basic elements in semiosis (sign action):

1. A *sign* (or *representamen*)^[123] represents, in the broadest possible sense of "represents". It is something interpretable as saying something about something. It is not necessarily symbolic, linguistic, or artificial—a cloud might be a sign of rain for instance, or ruins the sign of ancient civilization. As Peirce sometimes put it (he defined *sign* at least 76 times), the sign stands *for* the object *to* the interpretant. A sign represents its object in some respect, which respect is the sign's *ground*.
2. An *object* (or *semiotic object*) is a subject matter of a sign and an interpretant. It can be anything thinkable, a quality, an occurrence, a rule, etc., even fictional, such as Prince Hamlet.^[124] All of those are special or partial objects. The object most accurately is the universe of discourse to which the partial or special object belongs. For instance, a perturbation of Pluto's orbit is a sign about Pluto but ultimately not only about Pluto. An object either (i) is *immediate* to a sign and is the object as represented in the sign or (ii) is a *dynamic* object, the object as it really is, on which the immediate object is founded "as on bedrock".^[125]

3. An *interpretant* (or *interpretant sign*) is a sign's meaning or ramification as formed into a kind of idea or effect, an interpretation, human or otherwise. An interpretant is a sign (a) of the object and (b) of the interpretant's "predecessor" (the interpreted sign) as a sign of the same object. An interpretant either (i) is *immediate* to a sign and is a kind of quality or possibility such as a word's usual meaning, or (ii) is a *dynamic* interpretant, such as a state of agitation, or (iii) is a *final* or *normal* interpretant, a sum of the lessons which a sufficiently considered sign *would* have as effects on practice, and with which an actual interpretant may at most coincide.

Some of the understanding needed by the mind depends on familiarity with the object. To know what a given sign denotes, the mind needs some experience of that sign's object, experience outside of, and collateral to, that sign or sign system. In that context Peirce speaks of collateral experience, collateral observation, collateral acquaintance, all in much the same terms.

Classes of signs

Among Peirce's many sign typologies, three stand out, interlocked. The first typology depends on the sign itself, the second on how the sign stands for its denoted object, and the third on how the sign stands for its object to its interpretant. Also, each of the three typologies is a three-way division, a trichotomy, via Peirce's three phenomenological categories: (1) quality of feeling, (2) reaction, resistance, and (3) representation, mediation.

I. *Qualisign, sinsign, legisign* (also called *tone, token, type*, and also called *potisign, actisign, famisign*):^[126] This typology classifies every sign according to the sign's own phenomenological category—the qualisign is a quality, a possibility, a "First"; the sinsign is a reaction or resistance, a singular object, an actual event or fact, a "Second"; and the legisign is a habit, a rule, a representational relation, a "Third".

II. *Icon, index, symbol*: This typology, the best known one, classifies every sign according to the category of the sign's way of denoting its object—the icon (also called semblance or likeness) by a quality of its own, the index by factual connection to its object, and the symbol by a habit or rule for its interpretant.

III. *Rheme, dicisign, argument* (also called *sumisign, dicisign, suadisign*, also *seme, pheme, delome*, and regarded as very broadened versions of the traditional *term, proposition, argument*): This typology classifies every sign according to the category which the interpretant attributes to the sign's way of denoting its object—the rheme, for example a term, is a sign interpreted to represent its object in respect of quality; the dicisign, for example a proposition, is a sign interpreted to represent its object in respect of fact; and the argument is a sign interpreted to represent its object in respect of habit or law. This is the culminating typology of the three, where the sign is understood as a structural element of inference.

Lines of joint classification of signs.

Every sign is:^[1]

	1.		2.		3.
I.	Qualisign	or	Sinsign	or	Legisign
and			—	—	
II.	Icon	or	Index	or	Symbol
and			—	—	
III.	Rheme	or	Dicisign	or	Argument

Every sign belongs to one class or another within (I) and within (II) and within (III). Thus each of the three typologies is a three-valued parameter for every sign. The three parameters are not independent of each other; many co-classifications are absent, for reasons pertaining to the lack of either habit-taking or singular reaction in a quality, and the lack of habit-taking in a singular reaction. The result is not 27 but instead ten classes of signs fully specified at this level of analysis.

Modes of inference

Borrowing a brace of concepts from Aristotle, Peirce examined three basic modes of inference — *abduction*, *deduction*, and *induction* — in his "critique of arguments" or "logic proper". Peirce also called abduction "retroduction", "presumption", and, earliest of all, "hypothesis". He characterized it as guessing and as inference to an explanatory hypothesis. He sometimes expounded the modes of inference by transformations of the categorical syllogism Barbara (AAA), for example in "Deduction, Induction, and Hypothesis" (1878).^[127] He does this by rearranging the *rule* (Barbara's major premiss), the *case* (Barbara's minor premiss), and the *result* (Barbara's conclusion):

Deduction.	Induction.	Hypothesis (Abduction).
<p><i>Rule:</i> All the beans from this bag are white.</p> <p><i>Case:</i> These beans are from this bag.</p> <p>\therefore <i>Result:</i> These beans are white.</p>	<p><i>Case:</i> These beans are [randomly selected] from this bag.</p> <p><i>Result:</i> These beans are white.</p> <p>\therefore <i>Rule:</i> All the beans from this bag are white.</p>	<p><i>Rule:</i> All the beans from this bag are white.</p> <p><i>Result:</i> These beans [oddly] are white.</p> <p>\therefore <i>Case:</i> These beans are from this bag.</p>

Peirce 1883 in "A Theory of Probable Inference" (*Studies in Logic*) equated hypothetical inference with the induction of characters of objects (as he had done in effect before). Eventually dissatisfied, by 1900 he distinguished them once and for all and also wrote that he now took the syllogistic forms and the doctrine of logical extension and comprehension as being less basic than he had thought. In 1903 he presented the following logical form for abductive inference.^[128]

The surprising fact, C, is observed;

But if A were true, C would be a matter of course,

Hence, there is reason to suspect that A is true.

The logical form does not also cover induction, since induction neither depends on surprise nor proposes a new idea for its conclusion. Induction seeks facts to test a hypothesis; abduction seeks a hypothesis to account for facts. "Deduction proves that something *must* be; Induction shows that something *actually is* operative; Abduction merely suggests that something *may be*."^[129] Peirce did not remain quite convinced that one logical form covers all abduction.^[130] In his methodeutic or theory of inquiry (see below), he portrayed abduction as an economic initiative to further inference and study, and portrayed all three modes as clarified by their coordination in essential roles in inquiry: hypothetical explanation, deductive prediction, inductive testing.

Pragmatism

Some noted articles and lectures

- Illustrations of the Logic of Science (1877–78):
 - inquiry, pragmatism, statistics, inference
- 1. The Fixation of Belief (1877)
- 2. How to Make Our Ideas Clear (1878)
- 3. The Doctrine of Chances (1878)
- 4. The Probability of Induction (1878)
- 5. The Order of Nature (1878)
- 6. Deduction, Induction, and Hypothesis (1878)
- The Harvard lectures on pragmatism (1903)
- What Pragmatism Is (1905)
- Issues of Pragmaticism (1905)
- Pragmatism (1907 MS in EP 2)

Peirce's recipe for pragmatic thinking, which he called *pragmatism* and, later, *pragmaticism*, is recapitulated in several versions of the so-called *pragmatic maxim*. Here is one of his more emphatic reiterations of it:

Consider what effects that might *conceivably* have practical bearings you *conceive* the objects of your *conception* to have. Then, your *conception* of those effects is the whole of your *conception* of the object.

As a movement, pragmatism began in the early 1870s in discussions among Peirce, William James, and others in the Metaphysical Club. James among others regarded some articles by Peirce such as "The Fixation of Belief" (1877) and especially "How to Make Our Ideas Clear" (1878) as foundational to pragmatism.^[131] Peirce (CP 5.11–12), like James (*Pragmatism: A New Name for Some Old Ways of Thinking*, 1907), saw pragmatism as embodying familiar attitudes, in philosophy and elsewhere, elaborated into a new deliberate method for fruitful thinking about problems. Peirce differed from James and the early John Dewey, in some of their tangential enthusiasms, in being decidedly more rationalistic and realistic, in several senses of those terms, throughout the preponderance of his own philosophical moods. In 1905 Peirce coined the new name pragmaticism "for the precise purpose of expressing the original definition", saying that "all went happily" with James's and F.C.S. Schiller's variant uses of the old name "pragmatism" and that he coined the new name because of the old name's growing use in "literary journals, where it gets abused". Yet he cited as causes, in a 1906 manuscript, his differences with James and Schiller and, in a 1908 publication, his differences with James as well as literary author Giovanni Papini's declaration of pragmatism's indefinability. Peirce in any case regarded his views that truth is immutable and infinity is real, as being opposed by the other pragmatists, but he remained allied with them on other issues.^[132]

Pragmatism begins with the idea that belief is that on which one is prepared to act. Peirce's pragmatism is a method of clarification of conceptions of objects. It equates any conception of an object to a conception of that object's effects to a general extent of the effects' conceivable implications for informed practice. It is a method of sorting out conceptual confusions occasioned, for example, by distinctions that make (sometimes needed) formal yet not practical differences. He formulated both pragmatism and statistical principles as aspects of scientific logic, in his "Illustrations of the Logic of Science" series of articles. In the second one, "How to Make Our Ideas Clear", Peirce discussed three grades of clearness of conception:

1. Clearness of a conception familiar and readily used, even if unanalyzed and undeveloped.
2. Clearness of a conception in virtue of clearness of its parts, in virtue of which logicians called an idea "distinct", that is, clarified by analysis of just what makes it applicable. Elsewhere, echoing Kant, Peirce called a likewise distinct definition "nominal" (CP 5.553).
3. Clearness in virtue of clearness of conceivable practical implications of the object's conceived effects, such as fosters fruitful reasoning, especially on difficult problems. Here he introduced that which he later called the pragmatic maxim.

By way of example of how to clarify conceptions, he addressed conceptions about truth and the real as questions of the presuppositions of reasoning in general. In clearness's second grade (the "nominal" grade), he defined truth as a sign's correspondence to its object, and the real as the object of such correspondence, such that truth and the real are independent of that which you or I or any actual, definite community of inquirers think. After that needful but confined step, next in clearness's third grade (the pragmatic, practice-oriented grade) he defined truth as that opinion which *would* be reached, sooner or later but still inevitably, by research taken far enough, such that the real does depend on that ideal final opinion—a dependence to which he appeals in theoretical arguments elsewhere, for instance for the long-run validity of the rule of induction.^[133] Peirce argued that even to argue against the independence and discoverability of truth and the real is to presuppose that there is, about that very question under argument, a truth with just such independence and discoverability.

Peirce said that a conception's meaning consists in "all general modes of rational conduct" implied by "acceptance" of the conception—that is, if one were to accept, first of all, the conception as true, then what could one conceive to be consequent general modes of rational conduct by all who accept the conception as true?—the whole of such consequent general modes is the whole meaning. His pragmatism does not equate a conception's meaning, its

intellectual purport, with the conceived benefit or cost of the conception itself, like a meme (or, say, propaganda), outside the perspective of its being true, nor, since a conception is general, is its meaning equated with any definite set of actual consequences or upshots corroborating or undermining the conception or its worth. His pragmatism also bears no resemblance to "vulgar" pragmatism, which misleadingly connotes a ruthless and Machiavellian search for mercenary or political advantage. Instead the pragmatic maxim is the heart of his pragmatism as a method of experimental mental reflection^[134] arriving at conceptions in terms of conceivable confirmatory and disconfirmatory circumstances—a method hospitable to the formation of explanatory hypotheses, and conducive to the use and improvement of verification.^[135]

Peirce's pragmatism, as method and theory of definitions and conceptual clearness, is part of his theory of inquiry,^[136] which he variously called speculative, general, formal or universal rhetoric or simply methodeutic.^[137] He applied his pragmatism as a method throughout his work.

Theory of inquiry

Critical common-sensism

Critical common-sensism,^[137] treated by Peirce as a consequence of his pragmatism, is his combination of Thomas Reid's common-sense philosophy with a fallibilism that recognizes that propositions of our more or less vague common sense now indubitable may later come into actual question, for example because of science's transformation of our world. It includes efforts to work up genuine doubts in tests for a core group of common indubitables that varies slowly if at all.

Rival methods of inquiry

In *The Fixation of Belief* (1877), Peirce described inquiry in general not as the pursuit of truth *per se* but as the struggle to move from irritating, inhibitory doubt born of surprise, disagreement, and the like, and to reach a secure belief, belief being that on which one is prepared to act. That let Peirce frame scientific inquiry as part of a broader spectrum and as spurred, like inquiry generally, by actual doubt, not mere verbal, quarrelsome, or hyperbolic doubt, which he held to be fruitless. Peirce sketched **four methods** of settling opinion, ordered from least to most successful:

1. The method of **tenacity** (policy of sticking to initial belief) — which brings comforts and decisiveness but leads to trying to ignore contrary information and others' views as if truth were intrinsically private, not public. The method goes against the social impulse and easily falters since one may well notice when another's opinion seems as good as one's own initial opinion. Its successes can be brilliant but tend to be transitory.
2. The method of **authority** — which overcomes disagreements but sometimes brutally. Its successes can be majestic and long-lasting, but it cannot regulate people thoroughly enough to withstand doubts indefinitely, especially when people learn about other societies present and past.
3. The method of the **a priori** — which promotes conformity less brutally but fosters opinions as something like tastes, arising in conversation and comparisons of perspectives in terms of "what is agreeable to reason." Thereby it depends on fashion in paradigms and goes in circles over time. It is more intellectual and respectable but, like the first two methods, sustains accidental and capricious beliefs, destining some minds to doubt it.
4. The method of **science** — wherein inquiry supposes that the real is discoverable but independent of particular opinion, such that, unlike in the other methods, inquiry can, by its own account, go wrong (fallibilism), not only right, and thus purposely tests itself and criticizes, corrects, and improves itself.

Peirce held that, in practical affairs, slow and stumbling ratiocination is often dangerously inferior to instinct and traditional sentiment, and that the scientific method is best suited to theoretical research,^[138] which in turn should not be trammelled by the other methods and practical ends; reason's "first rule" is that, in order to learn, one must desire to learn and, as a corollary, must not block the way of inquiry. Scientific method excels the others finally by being deliberately designed to arrive — eventually — at the most secure beliefs, upon which the most successful

practices can be based. Starting from the idea that people seek not truth *per se* but instead to subdue irritating, inhibitory doubt, Peirce showed how, through the struggle, some can come to submit to truth for the sake of belief's integrity, seek as truth the guidance of potential conduct correctly to its given goal, and wed themselves to the scientific method.

Scientific method

Insofar as clarification by pragmatic reflection suits explanatory hypotheses and fosters predictions and testing, pragmatism points beyond the usual duo of foundational alternatives: deduction from self-evident truths, or *rationalism*; and induction from experiential phenomena, or *empiricism*.

Peirce's approach, based in his critique of three modes of argument, differs from approaches based in either foundationalism or coherentism about justification of claims, by a three-phase dynamic of inquiry:

1. Active, abductive genesis of theory, with no prior assurance of truth;
2. Deductive application of the contingent theory so as to clarify its practical implications;
3. Inductive testing and evaluation of the provisional theory's utility for the anticipation of future experience, in both senses: *prediction* and *control*.

Thereby he fleshed out an approach to inquiry far more solid than the flatter image of inductive generalization *simpliciter*, which is a mere relabeling of phenomenological patterns. Peirce's pragmatism was the first time the scientific method was proposed as an epistemology for philosophical questions.

A theory that succeeds better than its rivals in predicting and controlling our world is said to be nearer the truth. This is an operational notion of truth used by scientists.

Peirce extracted the pragmatic model or theory of inquiry from its raw materials in classical logic and refined it in parallel with the early development of symbolic logic to address problems about the nature of scientific reasoning.

Abduction, deduction, and induction make incomplete sense in isolation from one another but comprise a cycle understandable as a whole insofar as they collaborate toward inquiry's end. In the pragmatic way of thinking in terms of conceivable practical implications, every thing has a purpose, and its purpose is the first thing that we should try to note about it. Abduction hypothesizes an explanation for deduction to clarify into implications to be tested so that induction can evaluate the hypothesis, in the struggle to move from troublesome uncertainty to secure belief. No matter how traditional and needful it is to study the modes of inference in abstraction from one another, inquiry's integrity strongly limits the effective modularity of inquiry's principal components.

Peirce's outline of the scientific method in §III–IV of "A Neglected Argument"^[139] is summarized below (except as otherwise noted). There he also reviewed plausibility and inductive precision (issues of critique of arguments).

1. **Abductive** (or retroductive) phase. Guessing, inference to explanatory hypotheses for selection of those best worth trying. From abduction, Peirce distinguishes induction as inferring, on the basis of tests, the proportion of truth in the hypothesis. Every inquiry, whether into ideas, brute facts, or norms and laws, arises from surprising observations in one or more of those realms (and for example at any stage of an inquiry already underway). All explanatory content of theories comes from abduction, which guesses a new or outside idea so as to account in a simple, economical way for a surprising or complicated phenomenon. Oftenest even a well-prepared mind guesses wrong. But the modicum of success of our guesses far exceeds that of random luck, and seems born of attunement to nature by instincts developed or inherent, especially insofar as best guesses are optimally plausible and simple in the sense of the "facile and natural", as by Galileo's natural light of reason and as distinct from "logical simplicity".^[140]

Abduction is the most fertile but least secure mode of inference. Its general rationale is inductive: it succeeds often enough and it has no substitute in expediting us toward new truths.^[141] In 1903 Peirce called pragmatism "the logic of abduction".^[142] It points to efficiency. Coordinative method leads from abducing a plausible hypothesis to judging it for its testability^[143] and for how its trial would economize inquiry itself.^[144] The hypothesis, being insecure, needs to have practical implications leading at least to mental tests and, in science, lending themselves to scientific tests. A simple but unlikely guess, if uncostly to test for falsity, may belong first in line for testing. A guess

is intrinsically worth testing if it has instinctive plausibility or reasoned objective probability, while subjective likelihood, though reasoned, can be misleadingly seductive. Guesses can be chosen for trial strategically, for their caution (for which Peirce gave as example the game of Twenty Questions), breadth, or incomplexity.^[145] One can hope to discover only that which time would reveal through a learner's sufficient experience anyway, so the point is to expedite it; economy of research is what demands the leap, so to speak, of abduction and governs its art.

2. Deductive phase. Two stages:

- i. Explication. Unclearly premissed, but deductive, analysis of the hypothesis so as to render its parts as clear as possible.
- ii. Demonstration: Deductive Argumentation, Euclidean in procedure. Explicit deduction of hypothesis's consequences as predictions about evidence to be found. Corollarial or, if needed, Theorematic.

3. **Inductive** phase. Evaluation of the hypothesis, inferring from observational or experimental tests of its deduced consequences. The long-run validity of the rule of induction is deducible from the principle (presuppositional to reasoning in general) that the real "is only the object of the final opinion to which sufficient investigation would lead"; anything to which no such process would ever lead would not be real. Induction involving the ongoing accumulation of evidence follows "a method which, sufficiently persisted in," will "diminish the error below any predesignate degree." Three stages:

- i. Classification. Unclearly premissed, but inductive, classing of objects of experience under general ideas.
- ii. Probation: direct Inductive Argumentation. Crude or Gradual. Crude Induction, founded on experience in one mass (CP 2.759), presumes that future experience on a question will not differ utterly from all past experience (CP 2.756). Gradual Induction makes a new estimate of the proportion of truth in the hypothesis after each test, and is Qualitative or Quantitative. Qualitative Induction depends on estimating the relative evidential weights of the various qualities of the subject class under investigation (CP 2.759; see also CP 7.114–20). Quantitative Induction depends on how often, in a fair sample of instances of *S*, *S* is found actually accompanied by *P* that was predicted for *S* (CP 2.758). It depends on measurements, or statistics, or counting.
- iii. Sentential Induction. "...which, by Inductive reasonings, appraises the different Probations singly, then their combinations, then makes self-appraisal of these very appraisals themselves, and passes final judgment on the whole result".

Against Cartesianism

Peirce drew on the methodological implications of the four incapacities — no genuine introspection, no intuition in the sense of non-inferential cognition, no thought but in signs, and no conception of the absolutely incognizable — to attack philosophical Cartesianism, of which he said that:

1. "It teaches that philosophy must begin in universal doubt" — when, instead, we start with preconceptions, "prejudices [...] which it does not occur to us *can* be questioned", though we may find reason to question them later. "Let us not pretend to doubt in philosophy what we do not doubt in our hearts."
2. "It teaches that the ultimate test of certainty is...in the individual consciousness" — when, instead, in science a theory stays on probation till agreement is reached, then it has no actual doubters left. No lone individual can reasonably hope to fulfill philosophy's multi-generational dream. When "candid and disciplined minds" continue to disagree on a theoretical issue, even the theory's author should feel doubts about it.
3. It trusts to "a single thread of inference depending often upon inconspicuous premisses" — when, instead, philosophy should, "like the successful sciences", proceed only from tangible, scrutinizable premisses and trust not to any one argument but instead to "the multitude and variety of its arguments" as forming, not a chain at least as weak as its weakest link, but "a cable whose fibers", soever "slender, are sufficiently numerous and intimately connected".

4. It renders many facts "absolutely inexplicable, unless to say that 'God makes them so' is to be regarded as an explanation"^[146] — when, instead, philosophy should avoid being "unidealistic",^[147] misbelieving that something real can defy or evade all possible ideas, and supposing, inevitably, "some absolutely inexplicable, unanalyzable ultimate", which explanatory surmise explains nothing and so is inadmissible.

Philosophy: metaphysics

Some noted articles

- The *Monist* Metaphysical Series (1891–93)
 - The Architecture of Theories (1891)
 - The Doctrine of Necessity Examined (1892)
 - The Law of Mind (1892)
 - Man's Glassy Essence (1892)
 - Evolutionary Love (1893)
- Immortality in the Light of Synechism (1893 MS)

Peirce divided metaphysics into (1) ontology or general metaphysics, (2) psychical or religious metaphysics, and (3) physical metaphysics.

Ontology. Peirce was a Scholastic Realist, declaring for the reality of generals as early as 1868.^[148] Regarding modalities (possibility, necessity, etc.), he came in later years to regard himself as having wavered earlier as to just how positively real the modalities are. In his 1897 "The Logic of Relatives" he wrote:

I formerly defined the possible as that which in a given state of information (real or feigned) we do not know not to be true. But this definition today seems to me only a twisted phrase which, by means of two negatives, conceals an anacoluthon. We know in advance of experience that certain things are not true, because we see they are impossible.

Peirce retained, as useful for some purposes, the definitions in terms of information states, but insisted that the pragmaticist is committed to a strong modal realism by conceiving of objects in terms of predictive general conditional propositions about how they *would* behave under certain circumstances.^[149]

Psychical or Religious Metaphysics. Peirce believed in God, and characterized such belief as founded in an instinct exploratory in musing over the worlds of ideas, brute facts, and evolving habits — and it is a belief in God not as an *actual* or *existent* being (in Peirce's sense of those words), but all the same as a *real* being.^[150] In "A Neglected Argument for the Reality of God" (1908), Peirce sketches, for God's reality, an argument to a hypothesis of God as the Necessary Being, a hypothesis which he describes in terms of how it would tend to develop and become compelling in musement and inquiry by a normal person who is led, by the hypothesis, to consider as being purposed the features of the worlds of ideas, brute facts, and evolving habits (for example scientific progress), such that the thought of such purposefulness will "stand or fall with the hypothesis"; meanwhile, according to Peirce, the hypothesis, in supposing an "infinitely incomprehensible" being, starts off at odds with its own nature as a purportively true conception, and so, no matter how much the hypothesis grows, it both (A) inevitably regards itself as partly true, partly vague, and as continuing to define itself without limit, and (B) inevitably has God appearing likewise vague but growing, though God as the Necessary Being is not vague or growing; but the hypothesis will hold it to be *more* false to say the opposite, that God is purposeless. Peirce also argued that the will is free^[151] and (see Synechism) that there is at least an attenuated kind of immortality.

Physical Metaphysics. Peirce held the view, which he called objective idealism, that "matter is effete mind, inveterate habits becoming physical laws".^[152] Peirce asserted the reality of (1) absolute chance (his tychist view), (2) mechanical necessity (anancist view), and (3) that which he called the law of love (agapist view), echoing his categories Firstness, Secondness, and Thirdness, respectively. He held that fortuitous variation (which he also called "sporting"), mechanical necessity, and creative love are the three modes of evolution (modes called "tychasm",

"anancasm", and "agapasm")^[153] of the cosmos and its parts. He found his conception of agapasm embodied in Lamarckian evolution; the overall idea in any case is that of evolution tending toward an end or goal, and it could also be the evolution of a mind or a society; it is the kind of evolution which manifests workings of mind in some general sense. He said that overall he was a synechist, holding with reality of continuity,^[154] especially of space, time, and law.^[155]

Science of review

Peirce outlined two fields, "Cenoscropy" and "Science of Review", both of which he called philosophy. Both included philosophy about science. In 1903 he arranged them, from more to less theoretically basic, thus:

1. Science of Discovery.
 1. Mathematics.
 2. Cenoscropy (philosophy as discussed earlier in this article—categorial, normative, metaphysical), as First Philosophy, concerns positive phenomena in general, does not rely on findings from special sciences, and includes the *general* study of inquiry and scientific method.
 3. Idioscopy, or the Special Sciences (of nature and mind).
2. Science of Review, as Ultimate Philosophy, arranges "...the results of discovery, beginning with digests, and going on to endeavor to form a philosophy of science". His examples included Humboldt's *Cosmos*, Comte's *Philosophie positive*, and Spencer's *Synthetic Philosophy*.
3. Practical Science, or the Arts.

Peirce placed, within Science of Review, the work and theory of classifying the sciences (including mathematics and philosophy). His classifications, on which he worked for many years, draw on argument and wide knowledge, and are of interest both as a map for navigating his philosophy and as an accomplished polymath's survey of research in his time.

Notes

- [1] Hacking, Ian (1990), "A Universe of Chance", *The Taming of Chance*, pp. 200–215, Cambridge U. Pr.
- [2] Stigler, Stephen M. (1978), " Mathematical statistics in the early States (<http://projecteuclid.org/euclid-aos/1176344123>)", *Annals of Statistics*, v. 6, March, pp. 239–265, see p. 248..
- [3] Crease, Robert P. (2009), "Charles Sanders Peirce and the first absolute measurement standard: In his brilliant but troubled life, Peirce was a pioneer in both metrology and philosophy", *Physics Today* v. 62, issue 12, December, pp. 39–44. Eprint (http://ptonline.aip.org/journals/doc/PHTOAD-ft/vol_62/iss_12/39_1.shtml?bypassSSO).
- [4] Cadwallader, Thomas C. (1974), " Charles S. Peirce (1839-1914): The first American experimental psychologist ([http://onlinelibrary.wiley.com/doi/10.1002/1520-6696\(197407\)10:3<291::AID-JHBS2300100304>3.0.CO;2-N/abstract](http://onlinelibrary.wiley.com/doi/10.1002/1520-6696(197407)10:3<291::AID-JHBS2300100304>3.0.CO;2-N/abstract))", *Journal of the History of the Behavioral Sciences*, v. 10, issue 3, pp. 291–8, July.
- [5] Wible, James R. (2008), " The Economic Mind of Charles Sanders Peirce (<http://www.ingentaconnect.com/content/rodopi/cpm/2008/00000005/00000002/art00003>)", *Contemporary Pragmatism*, v. 5, n. 2, December, pp. 39-67
- [6] Nöth, Winfried (2000), " Charles Sanders Peirce, Pathfinder in Linguistics (<http://www.digitalpeirce.fee.unicamp.br/ling.htm>)", *Digital Encyclopedia of Charles S. Peirce* (<http://www.digitalpeirce.fee.unicamp.br/>).
- [7] "Peirce", in the case of C.S. Peirce, always rhymes with the English-language word "terse" and so, in most dialects, is pronounced exactly like the English-language word "". See " Note on the Pronunciation of 'Peirce' (http://www.iupui.edu/~peirce/news/1_3/13_4x.htm#pronunciation)", *Peirce Project Newsletter*, v. 1, nos. 3/4, Dec. 1994.
- [8] Weiss, Paul (1934), "Peirce, Charles Sanders" in the *Dictionary of American Biography. Arisbe* Eprint (<http://www.cspeirce.com/menu/library/aboutcsp/weissbio.htm>).
- [9] "Peirce, Benjamin", subheading "Charles Sanders", in *Webster's Biographical Dictionary* (1943/1960), Springfield, MA: Merriam-Webster.
- [10] Peirce, C. S., "Letter, Peirce to A. Marquand", dated 1886, W 5:541–3, Google Preview (http://books.google.com/books?id=DnvLHp919_wC&q=Marquand). See Burks, Arthur W., "Review: Charles S. Peirce, *The new elements of mathematics*", *Bulletin of the American Mathematical Society* v. 84, n. 5 (1978), pp. 913–18, see 917. PDF Eprint (http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body&id=pdf_1&handle=euclid.bams/1183541145). Also p. xliv in Houser, Nathan, Introduction, W 5.
- [11] Fisch, Max, " Introduction (<http://www.iupui.edu/~peirce/writings/v1/v1intro.htm>)", W 1:xvii, find phrase "One episode".
- [12] "Peirce, Charles Sanders" (1898), *The National Cyclopaedia of American Biography*, v. 8, p. 409 (<http://books.google.com/books?id=1uI-AAAAYAAJ&pg=PA409&dq=%22Peirce,%20Charles%22>).

- [13] B:54–6
- [14] B:363–4
- [15] B:19–20, 53, 75, 245
- [16] B:40
- [17] Burch, Robert (2001, 2010), " Charles Sanders Peirce (<http://plato.stanford.edu/entries/peirce/>)", *Stanford Encyclopedia of Philosophy*.
- [18] B:139
- [19] B:61–2
- [20] B:69
- [21] B:368
- [22] B:79–81
- [23] Moore, Edward C., and Robin, Richard S., eds., (1964), *Studies in the Philosophy of Charles Sanders Peirce, Second Series*, Amherst: U. of Massachusetts Press. On Peirce the astronomer, see Lenzen's chapter.
- [24] B:367
- [25] Fisch, Max (1983), "Peirce as Scientist, Mathematician, Historian, Logician, and Philosopher", *Studies in Logic* (new edition), see p. x.
- [26] See "Peirce Edition Project (UQÀM) - in short (<http://www.pep.uqam.ca/short.pep>)" from PEP-UQÀM.
- [27] Houser, Nathan, "Introduction (<http://www.iupui.edu/~peirce/writings/v5/v5intro.htm>)", W 5:xxviii–xxix, find "Allison".
- [28] B:202
- [29] Houser, Nathan (1989), "Introduction (<http://www.iupui.edu/~peirce/writings/v4/v4intro.htm>)", W 4:xxxviii, find "Eighty-nine".
- [30] B:150–4, 195, 279–80, 289
- [31] B:xv
- [32] B:98–101
- [33] B:141
- [34] B:148
- [35] Houser, Nathan, "Introduction (<http://www.iupui.edu/~peirce/writings/v6/v6intro.htm>)", W 6, first paragraph.
- [36] B:123, 368
- [37] B:150–1, 368
- [38] In 1885 (B:369); in 1890 and 1900 (B:215, 273); in 1891 (B:215–16); and in 1892 (B:151–2, 222).
- [39] B:77
- [40] B:191–2, 217, 270, 318, 321, 337.
- [41] B:13
- [42] B:369–74
- [43] B:191
- [44] B:246
- [45] B:242
- [46] B:271
- [47] B:249–55
- [48] B:371
- [49] B:189
- [50] B:370
- [51] B:205–6
- [52] B:374–6
- [53] B:279–89
- [54] B:261–4, 290–2, 324
- [55] B:306–7 & 315–6
- [56] Russell, Bertrand (1959), *Wisdom of the West*, p. 276.
- [57] Anellis, Irving H. (1995), "Peirce Rustled, Russell Pierced: How Charles Peirce and Bertrand Russell Viewed Each Other's Work in Logic, and an Assessment of Russell's Accuracy and Role in the Historiography of Logic", *Modern Logic* 5, 270–328. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/aboutcsp/anellis/csp&br.htm>).
- [58] Popper, Karl (1972), *Objective Knowledge: An Evolutionary Approach*, p. 212.
- [59] See Royce, Josiah, and Kernan, W. Fergus (1916), "Charles Sanders Peirce", *The Journal of Philosophy, Psychology, and Scientific Method* v. 13, pp. 701–9. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/aboutcsp/royce/cspobit.htm>).
- [60] Ketner et al. (1986), *Comprehensive Bibliography*, see p. iii.
- [61] Hookway, Christopher (2008), "Pragmatism (<http://plato.stanford.edu/entries/pragmatism/>)", *Stanford Encyclopedia of Philosophy*.
- [62] B:8
- [63] Fisch, Max (1986), *Peirce, Semeiotic, and Pragmatism*, Kenneth Laine Ketner and Christian J. W. Kloesel, eds., Bloomington, Indiana: Indiana U. Pr.
- [64] Theological Research Group in C.S. Peirce's Philosophy (Hermann Deuser, Justus-Liebig-Universität Gießen; Wilfred Härtle, Philipps-Universität Marburg, Germany).
- [65] Burks, Arthur, Introduction, CP 7, p. xi.

- [66] Robin, Richard S. (1967), *Annotated Catalogue of the Papers of Charles S. Peirce* (<http://www.iupui.edu/~peirce/robin/robin.htm>). Amherst MA: University of Massachusetts Press.
- [67] "The manuscript material now (1997) comes to more than a hundred thousand pages. These contain many pages of no philosophical interest, but the number of pages on philosophy certainly number much more than half of that. Also, a significant but unknown number of manuscripts have been lost." — Joseph Ransdell (1997), "Some Leading Ideas of Peirce's Semiotic", end note 2 (<http://www.cspeirce.com/menu/library/aboutcsp/ransdell/leading.htm#note2>), 1997 light revision of 1977 version in *Semiotica* 19:157–78.
- [68] Houser, Nathan, "The Fortunes and Misfortunes of the Peirce Papers", Fourth Congress of the IASS, Perpignan, France, 1989. *Signs of Humanity*, v. 3, 1992, pp. 1259–68. Eprint (<http://www.cspeirce.com/menu/library/aboutcsp/houser/fortunes.htm>)
- [69] Memorandum to the President of Charles S. Peirce Society by Ahti-Veikko Pietarinen, U. of Helsinki, March 29, 2012. Eprint (<http://www.helsinki.fi/~pietarin/Memorandum-Peirce-Society-Pietarinen-2012.pdf>).
- [70] See for example " Collections of Peirce's Writings (<http://www.helsinki.fi/science/commens/collections.html>)" at *Commens*, U. of Helsinki.
- [71] See 1987 review by B. Kuklick (of *Peirce* by Christopher Hookway), in *British Journal for the Philosophy of Science* v. 38, n. 1, pp. 117–19. First page (http://bjps.oxfordjournals.org/cgi/pdf_extract/38/1/117).
- [72] Auspitz, Josiah Lee (1994), "The Wasp Leaves the Bottle: Charles Sanders Peirce", *The American Scholar*, v. 63, n. 4, autumn, 602–18. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/aboutcsp/auspitz/escape.htm>).
- [73] Burks, Arthur W., "Review: Charles S. Peirce, *The new elements of mathematics*", *Bulletin of the American Mathematical Society* v. 84, n. 5 (1978), pp. 913–18 (PDF) (http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body&id=pdf_1&handle=euclid.bams/1183541145).
- [74] Peirce (1860 MS), "Orders of Infinity", *News from the Peirce Edition Project*, September 2010 (http://www.iupui.edu/~peirce/PEP_news_Sept2010.pdf) (PDF), p. 6, with the manuscript's text. Also see logic historian Irving Anellis's November 11, 2010 comment (<http://thread.gmane.org/gmane.science.philosophy.peirce/6621/focus=6626>) at *peirce-l*.
- [75] Peirce (MS, winter of 1880–81), "A Boolean Algebra with One Constant", CP 4.12–20, W 4:218–21. Google Preview (<http://books.google.com/books?id=E7ZUnx3FqrcC&q=378+Winter>). See Roberts, Don D. (1973), *The Existential Graphs of Charles S. Peirce*, p. 131.
- [76] Peirce (1881), "On the Logic of Number", *American Journal of Mathematics* v. 4, pp. 85 (<http://books.google.com/books?id=LQgPAAAIAAJ&pg=PA85>)-95. Reprinted (CP 3.252–88), (W 4:299–309). See See Shields, Paul (1997), "Peirce's Axiomatization of Arithmetic", in Houser *et al.*, eds., *Studies in the Logic of Charles S. Peirce*.
- [77] Peirce (1885), "On the Algebra of Logic: A Contribution to the Philosophy of Notation", *American Journal of Mathematics* 7, two parts, first part published 1885, pp. 180 (<http://books.google.com/books?id=lwYPAAAIAAJ&pg=PA180>)-202 (see Houser in linked paragraph (<http://www.iupui.edu/~peirce/writings/v4/v4introx.htm#21note>) in "Introduction" in W 4). Presented, National Academy of Sciences, Newport, RI, 14–17 October 1884 (see EP 1, Headnote 16 (<http://www.iupui.edu/~peirce/ep/ep1/heads/ep1heads.htm#16>)). 1885 is the year usually given for this work. Reprinted CP 3.359–403, W 5:162–90, EP 1:225–8, in part.
- [78] It was in Peirce's 1885 "On the Algebra of Logic". See Byrnes, John (1998), "Peirce's First-Order Logic of 1885", *Transactions of the Charles S. Peirce Society* v. 34, n. 4, pp. 949–76.
- [79] Brady, Geraldine (2000), *From Peirce to Skolem: A Neglected Chapter in the History of Logic*, North-Holland/Elsevier Science BV, Amsterdam, Netherlands.
- [80] See Peirce (1898), Lecture 3, "The Logic of Relatives" (not the 1897 *Monist* article), *Reasoning and the Logic of Things*, pp. 146–64, see 151.
- [81] Peirce (1898), "The Logic of Mathematics in Relation to Education" in *Educational Review* v. 15, pp. 209–16 (<http://www.archive.org/stream/educationalrevie15newyuoft#page/209/mode/1up>) (via *Internet Archive*). Reprinted CP 3.553–62. See also his "The Simplest Mathematics" (1902 MS), CP 4.227–323.
- [82] Lewis, Clarence Irving (1918), *A Survey of Symbolic Logic*, see ch. 1, §7 "Peirce", pp. 79–106, see p. 79 (<http://www.archive.org/stream/surveyofsymbolic00lewiiala#page/79/mode/1up>) (*Internet Archive*). Note that Lewis's bibliography lists works by Frege, tagged with asterisks as important.
- [83] Avery, John (2003) *Information theory and evolution*, p. 167; also Mitchell, Melanie, " My Scientific Ancestry (<http://web.cecs.pdx.edu/~mm/MMScientificAncestry.html>)".
- [84] Beil, Ralph G. and Ketner, Kenneth (2003), "Peirce, Clifford, and Quantum Theory", *International Journal of Theoretical Physics* v. 42, n. 9, pp. 1957–1972.
- [85] Houser, Roberts, and Van Evra, eds. (1997), *Studies in the Logic of Charles Sanders Peirce*, Indiana U., Bloomington, IN.
- [86] Misak, ed. (2004), *The Cambridge Companion to Peirce*, Cambridge U., UK.
- [87] Peirce (1903 MS), CP 6.176: "But I now define a *pseudo-continuum* as that which modern writers on the theory of functions call a *continuum*. But this is fully represented by [...] the totality of real values, rational and irrational [...]."
- [88] Peirce (1902 MS) and Ransdell, Joseph, ed. (1998), "Analysis of the Methods of Mathematical Demonstration", Memoir 4 (<http://www.cspeirce.com/menu/library/bycsp/l75/ver1/l75v1-02.htm#m4>), Draft C, MS L75.90–102, see 99–100. (Once there, scroll down).
- [89] See:
- Peirce (1908), "Some Amazing Mazes (Conclusion), Explanation of Curiosity the First", *The Monist*, v. 18, n. 3, pp. 416–64, see 463 (<http://books.google.com/books?id=CqsLAAAIAAJ&pg=PA463>)-4. Reprinted CP 4.594–642, see 642.
 - Havenel, Jérôme (2008), "Peirce's Clarifications on Continuity", *Transactions Winter 2008* pp. 68–133, see 119. Abstract (<http://www.jstor.org/pss/40321237>).

- [90] Peirce condemned the use of "certain likelihoods" (EP 2:108–9) even more strongly than he criticized Bayesian methods. Indeed Peirce used a bit of Bayesian inference in criticizing parapsychology (W 6:76).
- [91] Miller, Richard W. (1975), "Propensity: Popper or Peirce?", *British Journal for the Philosophy of Science* (site (<http://bjps.oxfordjournals.org>)), v. 26, n. 2, pp. 123–32.. Eprint (<http://bjps.oxfordjournals.org/cgi/reprint/26/2/123.pdf>).
- [92] Haack, Susan and Kolenda, Konstantin (1977), "Two Fallibilists in Search of the Truth", *Proceedings of the Aristotelian Society*, Supplementary Volumes, v. 51, pp. 63–104.
- [93] Peirce (1897) "Fallibilism, Continuity, and Evolution", CP 1.141–75 (Eprint (<http://www.textlog.de/4248.html>)), placed by the CP editors directly after "F.R.L." (1899, CP 1.135–40).
- [94] Peirce (1903), CP 1.180–202 Eprint (<http://web.archive.org/web/20111105121054/>) and (1906) "The Basis of Pragmaticism", EP 2:372–3, see " Philosophy (<http://www.helsinki.fi/science/commens/terms/philosophy.html>)" at CDPT.
- [95] <http://www.cspeirce.com/menu/library/bycsp/newlist/nl-frame.htm>
- [96] See in "Firstness", "Secondness", and "Thirdness" in CDPT (<http://www.helsinki.fi/science/commens/dictionary.html>).
- [97] Peirce (1893), "The Categories" MS 403. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/bycsp/bycsp.htm#NLOC-R>), edited by Joseph Ransdell, with information on the re-write, and interleaved with the 1867 "New List" for comparison.
- [98] "Minute Logic", CP 2.87, c.1902 and A Letter to Lady Welby, CP 8.329, 1904. See relevant quotes under " Categories, Cenopythagorean Categories (<http://www.helsinki.fi/science/commens/terms/categories.html>)" in *Commens Dictionary of Peirce's Terms* (CDPT), Bergman & Paalova, eds., U. of Helsinki.
- [99] See quotes under " Firstness, First [as a category] (<http://www.helsinki.fi/science/commens/terms/firstness.html>)" in CDPT.
- [100] The ground **blackness** is the pure abstraction of the quality **black**. Something **black** is something **embodying blackness**, pointing us back to the abstraction. The quality **black** amounts to reference to its own pure abstraction, the ground **blackness**. The question is not merely of *noun* (the ground) versus *adjective* (the quality), but rather of whether we are considering the black(ness) as abstracted away from application to an object, or instead as so applied (for instance to a stove). Yet note that Peirce's distinction here is not that between a property-general and a property-individual (a trope). See " On a New List of Categories (<http://www.cspeirce.com/menu/library/bycsp/newlist/nl-frame.htm>)" (1867), in the section appearing in CP 1.551. Regarding the ground, cf. the Scholastic conception of a relation's *foundation*, Google limited preview Deely 1982, p. 61 (http://books.google.com/books?id=fSzt6_ce-gC&pg=PA61&dq=%22Introducing+Semiotic%22+foundation+ground&sig=kgh62kOzOoFrCOYyAV04YxJ0SOo#PPA61)
- [101] A quale in this sense is a *such*, just as a quality is a suchness. Cf. under "Use of Letters" in §3 of Peirce's "Description of a Notation for the Logic of Relatives", *Memoirs of the American Academy*, v. 9, pp. 317–78 (1870), separately reprinted (1870), from which see p. 6 via Google books (<http://books.google.com/books?id=fFnWmf5oLaoC&pg=PA6>), also reprinted in CP 3.63:
- [102] See quotes under " Secondness, Second [as a category] (<http://www.helsinki.fi/science/commens/terms/secondness.html>)" in CDPT.
- [103] See quotes under " Thirdness, Third [as a category] (<http://www.helsinki.fi/science/commens/terms/thirdness.html>)" in CDPT.
- [104] " Charles S. Peirce on Esthetics and Ethics: A Bibliography (http://agora.phi.gvsu.edu/kap/CSP_Bibliography/CSP_norm_bib.pdf)" (PDF) by Kelly A. Parker in 1999.
- [105] Peirce (1902 MS), Carnegie Application, edited by Joseph Ransdell, Memoir 2 (<http://www.cspeirce.com/menu/library/bycsp/l75/ver1/l75v1-02.htm>), see table.
- [106] See Esthetics (<http://www.helsinki.fi/science/commens/terms/esthetics.html>) at CDPT.
- [107] Peirce (1899 MS), "F.R.L." [First Rule of Logic], CP 1.135–40, Eprint (http://web.archive.org/web/20120106071421/http://www.princeton.edu/~batke/peirce/frl_99.htm)
- [108] Peirce (1882), "Introductory Lecture on the Study of Logic" delivered September 1882, *Johns Hopkins University Circulars*, v. 2, n. 19, pp. 11 (<http://books.google.com/books?id=E0YFAAAAQAAJ&pg=PA11&dq=%22art+of+devising+methods+of+research%22>)–12 (via Google), November 1882. Reprinted (EP 1:210–14; W 4:378–82; CP 7.59–76). The definition of logic quoted by Peirce is by Peter of Spain.
- [109] Peirce (1878), "The Doctrine of Chances", *Popular Science Monthly*, v. 12, pp. 604–15 (CP 2.645–68, W 3:276–90, EP 1:142–54).
- [110] Peirce, CP 5.448 footnote, from "The Basis of Pragmaticism" in 1906.
- [111] Peirce, (1868), "Questions concerning certain Faculties claimed for Man", *Journal of Speculative Philosophy* v. 2, n. 2, pp. 103 (http://books.google.com/books?id=YHkqP2JHJ_IC&pg=RA1-PA103-14). On thought in signs, see p. 112. Reprinted CP 5.213–63 (on thought in signs, see 253), W 2:193–211, EP 2:11–27. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/bycsp/question/qu-frame.htm>).
- [112] Peirce (1902), The Carnegie Institute Application, Memoir 10, MS L75.361-2, *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/bycsp/l75/ver1/l75v1-04.htm#m10>).
- [113] Peirce (1868), "Some Consequences of Four Incapacities", *Journal of Speculative Philosophy* v. 2, n. 3, pp. 140 (http://books.google.com/books?id=YHkqP2JHJ_IC&pg=RA1-PA140-57). Reprinted CP 5.264–317, W 2:211–42, EP 1:28–55. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/bycsp/conseq/cn-frame.htm>).
- [114] Peirce, "Grounds of Validity of the Laws of Logic: Further Consequences of Four Incapacities", *Journal of Speculative Philosophy* v. II, n. 4, pp. 193 (http://books.google.com/books?id=YHkqP2JHJ_IC&pg=RA1-PA193-208). Reprinted CP 5.318–357, W 2:242–272 (PEP Eprint (http://www.iupui.edu/~peirce/writings/v2/w2_w2_23/v2_23.htm)), EP 1:56–82.
- [115] Peirce (1905), "What Pragmatism Is", *The Monist*, v. XV, n. 2, pp. 161–81, see 167 (<http://books.google.com/books?id=j6oLAAAAIAAJ&pg=PA167>). Reprinted CP 5.411–37, see 416. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/bycsp/whatis/whatpragis.htm>).
- [116] Peirce 1907, CP 5.484. Reprinted, EP 2:411 in "Pragmatism" (398–433).
- [117] See " Quasi-mind (<http://www.helsinki.fi/science/commens/terms/quasimind.html>)" in CDPT.

- [118] Peirce (1867), "Upon Logical Comprehension and Extension" (CP 2.391–426), (W 2:70–86 (http://www.iupui.edu/~peirce/writings/v2/w2_06/v2_06.htm)).
- [119] See pp. 404–9 in "Pragmatism" in EP 2. Ten quotes on collateral experience from Peirce provided by Joseph Ransdell can be viewed here (<http://lyris.ttu.edu/read/messages?id=57101>) at peirce-l's Lyris archive. Note: Ransdell's quotes from CP 8.178–9 are also in EP 2:493–4, which gives their date as 1909; and his quote from CP 8.183 is also in EP 2:495–6, which gives its date as 1909.
- [120] Peirce, letter to William James, dated 1909, see EP 2:492.
- [121] See " 76 definitions of the sign by C.S.Peirce (<http://perso.numericable.fr/robert.marty/semitique/76defeng.htm>)", collected by Robert Marty (U. of Perpignan, France).
- [122] Peirce, A Letter to Lady Welby (1908), *Semiotic and Significs*, pp. 80–1:
- [123] "Representamen", properly with the 'a' long and stressed (), was adopted (not coined) by Peirce as his technical term for the *sign* as covered in his theory, in case a divergence should come to light between his theoretical version and the popular senses of the word "sign". He eventually stopped using "representamen". See EP 2:272–3 and *Semiotic and Significs* p. 193, quotes in " Representamen (<http://www.helsinki.fi/science/commens/terms/representamen.html>)" at CDPT.
- [124] Peirce (1909), A Letter to William James, EP 2:492–502. Fictional object, 498. Object as universe of discourse, 492. See " Dynamical Object (<http://www.helsinki.fi/science/commens/terms/dynamicalobject.html>)" at CDPT.
- [125] See "Immediate Object", etc., at CDPT (<http://www.helsinki.fi/science/commens/dictionary.html>).
- [126] On the varying terminology, look up in CDPT (<http://www.helsinki.fi/science/commens/dictionary.html>).
- [127] *Popular Science Monthly*, v. 13, pp. 470–82, see 472 (<http://books.google.com/books?id=u8sWAQAAIAAJ&pg=PA472>) or the book at Wikisource. CP 2.619–44, see 623.
- [128] See, under " Abduction (<http://www.helsinki.fi/science/commens/terms/abduction.html>)" at CDPT, the following quotes:
- On correction of "A Theory of Probable Inference", see quotes from "Minute Logic", CP 2.102, c. 1902, and from the Carnegie Application (L75), 1902, *Historical Perspectives on Peirce's Logic of Science* v. 2, pp. 1031–1032.
 - On new logical form for abduction, see quote from Harvard Lectures on Pragmatism, 1903, CP 5.188–189.
- See also Santaella, Lucia (1997) "The Development of Peirce's Three Types of Reasoning: Abduction, Deduction, and Induction", 6th Congress of the IASS. Eprint (http://www.pucsp.br/~lbraga/epap_peir1.htm).
- [129] "Lectures on Pragmatism", 1903, CP 5.171.
- [130] A Letter to J. H. Kehler (dated 1911), *The New Elements of Mathematics* v. 3, pp. 203–4, see in " Retroduction (<http://www.helsinki.fi/science/commens/terms/retroduction.html>)" at CDPT.
- [131] James, William (1897), *The Will to Believe*, see p. 124.
- [132] See Pragmaticism#Pragmaticism's name for discussion and references.
- [133] "That the rule of induction will hold good in the long run may be deduced from the principle that reality is only the object of the final opinion to which sufficient investigation would lead", in Peirce (1878 April), "The Probability of Induction", p. 718 (<http://www.archive.org/stream/popscimonthly12youmiss#page/728/mode/1up>) (via Internet Archive) in *Popular Science Monthly*, v. 12, pp. 705–18. Reprinted in CP 2.669–93, W 3:290–305, EP 1:155–69, elsewhere.
- [134] Peirce (1902), CP 5.13 note 1.
- [135] See CP 1.34 Eprint (<http://www.textlog.de/4220.html>) (in "The Spirit of Scholasticism"), where Peirce ascribed the success of modern science less to a novel interest in verification than to the improvement of verification.
- [136] See Joseph Ransdell's comments and his tabular list of titles of Peirce's proposed list of memoirs in 1902 for his Carnegie application, Eprint (<http://www.cspeirce.com/menu/library/bycsp/l75/intro/l75intro.htm>)
- [137] Peirce (1905), "Issues of Pragmaticism", *The Monist*, v. XV, n. 4, pp. 481 (<http://books.google.com/books?id=j6oLAAAAIAAJ&pg=PA481-IA20-99>). Reprinted CP 5.497–525. Also important: CP 5.497–525.
- [138] Peirce, "Philosophy and the Conduct of Life", Lecture 1 of the 1898 Cambridge (MA) Conferences Lectures, CP 1.616–48 in part and *Reasoning and the Logic of Things*, 105–22, reprinted in EP 2:27–41.
- [139] Peirce (1908), "A Neglected Argument for the Reality of God", published in large part, *Hibbert Journal* v. 7, 90–112. Reprinted with an unpublished part, CP 6.452–85, *Selected Writings* pp. 358–79, EP 2:434–50, *Peirce on Signs* 260–78.
- [140] See also Nubiola, Jaime (2004), " Il Lume Naturale: Abduction and God (<http://www.unav.es/users/LumeNaturale.html>)", *Semiotiche* I/2, 91–102.
- [141] Peirce (c. 1906), "PAP (Prolegomena to an Apology for Pragmatism)" (MS 293), *The New Elements of Mathematics* v. 4, pp. 319–20, first quote under " Abduction (<http://www.helsinki.fi/science/commens/terms/abduction.html>)" at CDPT.
- [142] Peirce (1903), "Pragmatism – The Logic of Abduction", CP 5.195–205, especially 196. Eprint (<http://www.textlog.de/7663.html>).
- [143] Peirce, Carnegie application, MS L75.279–280: Memoir 27 (<http://www.cspeirce.com/menu/library/bycsp/l75/ver1/l75v1-08.htm#m27>), Draft B.
- [144] See MS L75.329–330, from Draft D of Memoir 27 (<http://www.cspeirce.com/menu/library/bycsp/l75/ver1/l75v1-08.htm#m27>) of Peirce's application to the Carnegie Institution:
- [145] Peirce, C. S., "On the Logic of Drawing Ancient History from Documents", EP 2, see 107–9. On Twenty Questions, see 109:
- [146] Peirce believed in God. See section #Philosophy: metaphysics.
- [147] However, Peirce disagreed with Hegelian absolute idealism. See for example CP 8.131.
- [148] Peirce (1868), "Nominalism versus Realism", *Journal of Speculative Philosophy* v. 2, n. 1, pp. 57 (http://books.google.com/books?id=YHkqP2JHJ_IC&pg=RA1-PA57-61). Reprinted (CP 6.619–24), (W 2:144–53 (<http://www.iupui.edu/~peirce/writings/v2/w2/>))

w2_14/v2_14.htm)).

[149] On developments in Peirce's realism, see:

- Peirce (1897), "The Logic of Relatives", *The Monist* v. VII, n. 2 pp. 161–217, see 206 (<http://books.google.com/books?id=pa0LAAAAIAAJ&pg=PA206>) (via Google). Reprinted CP 3.456–552.
- Peirce (1905), "Issues of Pragmaticism", *The Monist* v. XV, n. 4, pp. 481–99, see 495–6 (<http://books.google.com/books?id=j6oLAAAAIAAJ&pg=PA495>) (via Google). Reprinted (CP 5.438–63, see 453–7).
- Peirce (c. 1905), Letter to Signor Calderoni, CP 8.205–13, see 208.
- Lane, Robert (2007), "Peirce's Modal Shift: From Set Theory to Pragmaticism", *Journal of the History of Philosophy*, v. 45, n. 4.

[150] Peirce in his 1906 "Answers to Questions concerning my Belief in God", CP 6.495, Eprint (<http://users.xplornet.com/~gnox/CSPgod.htm>), reprinted in part as "The Concept of God" in *Philosophical Writings of Peirce*, J. Buchler, ed., 1940, pp. 375–8:

[151] See his "The Doctrine of Necessity Examined" (1892) and "Reply to the Necessitarians" (1893), to both of which editor Paul Carus responded.

[152] Peirce (1891), "The Architecture of Theories", *The Monist* v. 1, pp. 161 (<http://www.archive.org/stream/monistquart01hegeuoft#page/161/mode/1up>)–76, see p. 170 (<http://www.archive.org/stream/monistquart01hegeuoft#page/170/mode/1up>), via *Internet Archive*. Reprinted (CP 6.7–34) and (EP 1:285–97, see p. 293).

[153] See "tychism", "tychasm", "tychasticism", and the rest, at CDPT (<http://www.helsinki.fi/science/commens/dictionary.html>).

[154] Peirce (1893), "Evolutionary Love", *The Monist* v. 3, pp. 176–200. Reprinted CP 6.278–317, EP 1:352–72. *Arisbe* Eprint (<http://www.cspeirce.com/menu/library/bycsp/evolove/evolove.htm>)

[155] See p. 115 in *Reasoning and the Logic of Things* (Peirce's 1898 lectures).

External links

Charles Sanders Peirce bibliography has external links throughout to such materials as biographical and overview articles on Peirce at encyclopedias, study sites, etc.; individual works by Peirce; and collections, bibliographies, and Peirce's definitions in the Baldwin dictionary.

Other useful sets of links:

- Existential graph references and external links.
- Pragmatism external links.
- Semiotics external links.

Peirce sites

- Arisbe: The Peirce Gateway (<http://www.cspeirce.com/>), Joseph Ransdell, ed. Over 100 online writings by Peirce as of 11/24/10, with annotations. 100s of online papers on Peirce. The peirce-l e-forum. Much else.
- Center for Applied Semiotics (CAS) (<http://replay.web.archive.org/20030806032358/http://www.indiana.edu/~sign/>) (1998–2003), Donald Cunningham & Jean Umiker-Sebeok, Indiana U.
- Centro de Estudos Peirceanos (<http://www.pucsp.br/pos/cos/cepe/>) (CeneP) and Centro Internacional de Estudos Peirceanos (<http://estudospeirceanos.wordpress.com/>) (CIEP), Lucia Santaella *et al.*, Pontifical Catholic U. of São Paulo (PUC-SP), Brazil. In Portuguese, some English.
- Centro Studi Peirce (<http://www.filosofia.unimi.it/peirce/>), Carlo Sini, Rossella Fabbrichesi, *et al.*, U. of Milan, Italy. In Italian and English. Part of Pragma (<http://www.associazionepragma.com/>).
- Charles S. Peirce Foundation (<http://www.peirce-foundation.org/>). Co-sponsoring the 2014 Peirce International Centennial Congress (100th anniversary of Peirce's death).
- Charles S. Peirce Society (<http://www.peircesociety.org/>)
 - *Transactions of the Charles S. Peirce Society* (<http://www.peircesociety.org/transactions.html>). Quarterly journal of Peirce studies since spring 1965. Table of Contents (<http://www.peircesociety.org/contents.html>) of all issues.
- Charles S. Peirce Studies (<http://www.peirce.org/>), Brian Kariger, ed.
- Charles Sanders Peirce (<http://genealogy.math.ndsu.nodak.edu/id.php?id=24099>) at the Mathematics Genealogy Project
- Collegium for the Advanced Study of Picture Act and Embodiment (<http://translate.google.com/translate?hl=en&sl=de&u=http://bildakt-verkoerperung.de/forschungsschwerpunkte/>): The Peirce Archive. Humboldt U, Berlin, Germany. Cataloguing Peirce's innumerable drawings & graphic materials. More info (<http://translate.google.com/translate?hl=en&sl=de&u=http://bildakt-verkoerperung.de/forschungsschwerpunkte/>)

- /www.audsisselhoel.com/wordpress/?p=69) (Prof. Aud Sissel Hoel).
- Digital Encyclopedia of Charles S. Peirce (<http://www.digitalpeirce.fee.unicamp.br/>), João Queiroz (now at UFJF (<http://ufjf.academia.edu/JoaoQueiroz>) & Ricardo Gudwin (at Unicamp (<http://www.dca.fee.unicamp.br/~gudwin/>)), eds., U. of Campinas, Brazil, in English. 84 authors listed, 51 papers online & more listed, as of 1/31/09.
 - Existential Graphs (<http://www.existentialgraphs.com/>), Jay Zeman, ed., U. of Florida. Has 4 Peirce texts.
 - Grupo de Estudios Peirceanos (GEP) / Peirce Studies Group (<http://www.unav.es/cep/index-en.html>), Jaime Nubiola, ed., U. of Navarra, Spain. Big study site, Peirce & others in Spanish & English, bibliography, more.
 - Helsinki Peirce Research Center (<http://www.helsinki.fi/peirce/>) (HPRC), Ahti-Veikko Pietarinen *et al.*, U. of Helsinki, with Commens: Virtual Centre for Peirce Studies (<http://www.helsinki.fi/science/commens/>), Mats Bergman & Sami Paavola, eds. 23 papers by 11 authors as of 11/24/10.
 - *Commens Dictionary of Peirce's Terms* (<http://www.helsinki.fi/science/commens/dictionary.html>) (CDPT): Peirce's own definitions, often many per term across the decades.
 - His Glassy Essence (<http://www.wyttnys.net/>). Autobiographical Peirce. Kenneth Laine Ketner.
 - Institute for Studies in Pragmaticism (<http://www.pragmaticism.net/>), Kenneth Laine Ketner, Clyde Hendrick, *et al.*, Texas Tech U. Peirce's life and works.
 - International Research Group on Abductive Inference ([http://web.archive.org/web/*/<http://www.rz.uni-frankfurt.de/~wirth>](http://web.archive.org/web/*/http://www.rz.uni-frankfurt.de/~wirth)), Uwe Wirth *et al.*, eds., Goethe U., Frankfurt, Germany. Uses frames. Click on link at bottom of its home page for English. Moved to U. of Gießen, Germany, home page (<http://www.abduktionsforschung.de/>) not in English but see Artikel section there.
 - L'I.R.S.C.E. ([http://replay.web.archive.org/20070717060233/<http://webup.univ-perp.fr/lsh/rch/semiotics/irsce/irsce.html>](http://replay.web.archive.org/20070717060233/http://webup.univ-perp.fr/lsh/rch/semiotics/irsce/irsce.html)) (1974–2003)—Institut de Recherche en Sémiotique, Communication et Éducation, Gérard Deledalle, Joëlle Réthoré, U. of Perpignan, France.
 - Minute Semeiotic (<http://www.minutesemeiotic.org/?lang=en>), Vinicius Romanini, U. of São Paulo, Brazil. English, Portuguese.
 - Peirce (http://www.signosemio.com/peirce/a_peirce.asp) at *Signo: Theoretical Semiotics on the Web*, Louis Hébert, director, supported by U. of Québec. Theory, application, exercises of Peirce's Semiotics (http://www.signosemio.com/peirce/a_semiotique.asp) and Esthetics (http://www.signosemio.com/peirce/a_esthetique.asp). English, French.
 - Peirce Edition Project (PEP) (<http://www.iupui.edu/~peirce/>), Indiana U.-Purdue U. Indianapolis (IUPUI). André De Tienne, Nathan Houser, *et al.* Editors of the *Writings of Charles S. Peirce* (W) and *The Essential Peirce* (EP) v. 2. Many study aids such as the Robin Catalog of Peirce's manuscripts & letters and:
 - Biographical introductions to EP 1–2 (<http://www.iupui.edu/~peirce/ep/ep.htm>) and W 1–6 (<http://www.iupui.edu/~peirce/writings/crit.htm>) & 8 (<http://www.iupui.edu/~peirce/houserintro.html>)
 - Most of W 2 (<http://www.iupui.edu/~peirce/writings/v2/toc2.htm>) readable online.
 - PEP's branch at Université du Québec à Montréal (UQÀM) (<http://www.pep.uqam.ca/>). Working on W 7: Peirce's work on the *Century Dictionary*. Definition of the week (<http://www.pep.uqam.ca/definitionoftheweek.pep>).
 - Peirce's Existential Graphs (http://www.dr-dau.net/eg_readings.shtml), Frithjof Dau, Germany
 - Peirce's Theory of Semiosis: Toward a Logic of Mutual Affection (<http://www.chass.utoronto.ca/epc/srb/cyber/espout.html>), Joseph Esposito. Free online course.
 - Pragmatism Cybrary (<http://www.pragmatism.org/>), David Hildebrand & John Shook.
 - Research Group on Semiotic Epistemology and Mathematics Education ([http://replay.web.archive.org/19970519142208/<http://www.uni-bielefeld.de/idm/eng/arbeite/agsem.htm>](http://replay.web.archive.org/19970519142208/http://www.uni-bielefeld.de/idm/eng/arbeite/agsem.htm)) (late 1990s), Institut für Didaktik der Mathematik (Michael Hoffman, Michael Otte, Universität Bielefeld, Germany). See *Peirce Project Newsletter* v. 3, n. 1, p. 13 (http://www.iupui.edu/~peirce/news/3_1/3_1pdf/Page13.pdf).

- Semiotics according to Robert Marty (<http://perso.numericable.fr/robert.marty/semitotique/anglais.htm>), with 76 definitions of the sign by C. S. Peirce (<http://perso.numericable.fr/robert.marty/semitotique/access.htm>).

George Boole

George Boole



A portrait of George Boole, an English mathematician, philosopher, and logician. He is shown from the chest up, wearing a dark blue three-piece suit, a white shirt with a high collar, and a dark bow tie. His hair is dark and receding at the temples. He has a serious expression and is looking slightly to his left.

George Boole	
Born	2 November 1815 Lincoln, Lincolnshire, England
Died	8 December 1864 (aged 49) Ballintemple, County Cork, Ireland
Nationality	English
Era	19th-century philosophy
Region	Western Philosophy
Religion	Unitarian
School	Mathematical foundations of computer science
Main interests	Mathematics, Logic, Philosophy of mathematics
Notable ideas	Boolean algebra

George Boole (/ˈbuːl/; 2 November 1815 – 8 December 1864) was an English mathematician, philosopher and logician. He worked in the fields of differential equations and algebraic logic, and is now best known as the author of *The Laws of Thought*. As the inventor of the prototype of what is now called Boolean logic, which became the basis of the modern digital computer, Boole is regarded in hindsight as a founder of the field of computer science. Boole said,

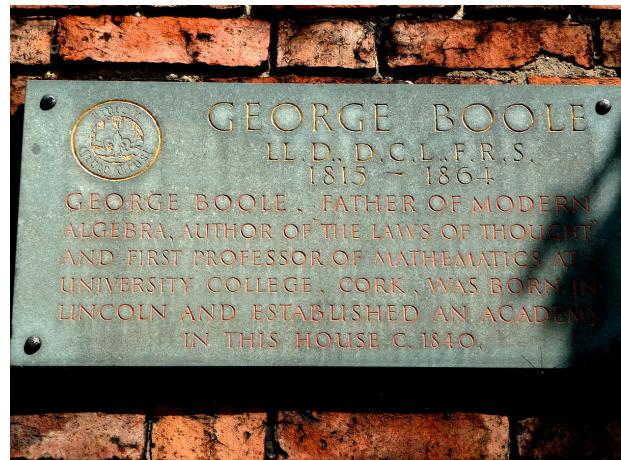
... no general method for the solution of questions in the theory of probabilities can be established which does not explicitly recognise ... those universal laws of thought which are the basis of all reasoning ...

Early life

George Boole's father, John Boole (1779–1848), was a tradesman in Lincoln and gave him lessons. He had an elementary school education, but little further formal and academic teaching. William Brooke, a bookseller in Lincoln, may have helped him with Latin; which he may also have learned at the school of Thomas Bainbridge. He was self-taught in modern languages.^[1] At age 16 Boole became the breadwinner for his parents and three younger siblings, taking up a junior teaching position in Doncaster, at Heigham's School.^[2] He taught briefly in Liverpool.

Boole participated in the local Mechanics Institute, the Lincoln Mechanics' Institution, which was founded in 1833.^[3] Edward Bromhead, who knew John Boole through the Institution, helped George Boole with mathematics books; and he was given the calculus text of Sylvestre François Lacroix by Rev. George Stevens Dickson, of St Swithin Lincoln. Without a teacher, it took him many years to master calculus.

Boole's Lincoln House



Plaque from the house in Lincoln.

At age 19 Boole successfully established his own school at Lincoln. Four years later he took over Hall's Academy, at Waddington, outside Lincoln, following the death of Robert Hall. In 1840 he moved back to Lincoln, where he ran a boarding school.

Boole became a prominent local figure, an admirer of John Kaye, the bishop.^[4] He took part in the local campaign for early closing. With E. R. Larken and others he set up a building society in 1847.^[5] He associated also with the Chartist Thomas Cooper, whose wife was a relation.^[6]

From 1838 onwards Boole was making contacts with sympathetic British academic mathematicians, and reading more widely. He studied algebra in the form of symbolic methods, as these were understood at the time, and began to publish research papers.

Professor at Cork

Boole's status as mathematician was recognised by his appointment in 1849 as the first professor of mathematics at Queen's College, Cork in Ireland. He met his future wife, Mary Everest, there in 1850 while she was visiting her uncle John Ryall who was Professor of Greek. They married some years later.^[7] He maintained his ties with Lincoln, working there with E. R. Larken in a campaign to reduce prostitution.^[8]

Boole was elected Fellow of the Royal Society in 1857; and received honorary degrees of LL.D. from the University of Dublin and Oxford University.^[citation needed]

Death

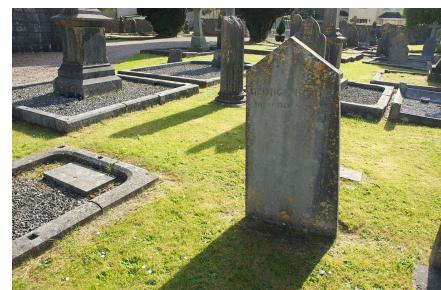
On 8 December 1864, Boole died of an attack of fever, ending in pleural effusion. He was buried in the Church of Ireland cemetery of St Michael's, Church Road, Blackrock (a suburb of Cork City). There is a commemorative plaque inside the adjoining church.^[citation needed]



The house in Cork in which Boole lived between 1849 and 1855.

Works

Boole's first published paper was *Researches in the theory of analytical transformations, with a special application to the reduction of the general equation of the second order*, printed in the *Cambridge Mathematical Journal* in February 1840 (Volume 2, no. 8, pp. 64–73), and it led to a friendship between Boole and Duncan Farquharson Gregory, the editor of the journal. His works are in about 50 articles and a few separate publications.^[9]



Boole's gravestone, Cork, Ireland.

In 1841 Boole published an influential paper in early invariant theory.

He received a medal from the Royal Society for his memoir of 1844, *On A General Method of Analysis*. It was a contribution to the theory of linear differential equations, moving from the case of constant coefficients on which he had already published, to variable coefficients.^[10] The innovation in operational methods is to admit that operations may not commute.^[11] In 1847 Boole published *The Mathematical Analysis of Logic*, the first of his works on symbolic logic.^[12]

Differential equations

Two systematic treatises on mathematical subjects were completed by Boole during his lifetime. The *Treatise on Differential Equations* appeared in 1859, and was followed, the next year, by a *Treatise on the Calculus of Finite Differences*, a sequel to the former work. In the

sixteenth and seventeenth chapters of the *Differential Equations* is an account of the general symbolic method, and of a general method in analysis, originally described in his memoir printed in the *Philosophical Transactions* for 1844.^[citation needed]

During the last few years of his life Boole worked on a second edition of his *Differential Equations*, and part of his last vacation was spent in the libraries of the Royal Society and the British Museum; but it was left incomplete. Isaac Todhunter printed the manuscripts in 1865, in a supplementary volume.^[citation needed]

Analysis

In 1857, Boole published the treatise *On the Comparison of Transcendents, with Certain Applications to the Theory of Definite Integrals*, in which he studied the sum of residues of a rational function. Among other results, he proved what is now called Boole's identity:



Detail of stained glass window in Lincoln Cathedral dedicated to George Boole.



Plaque beneath Boole's window in Lincoln Cathedral.

$$\text{mes} \left\{ x \in \mathbb{R} \mid \Re \frac{1}{\pi} \sum \frac{a_k}{x - b_k} \geq t \right\} = \frac{\sum a_k}{\pi t}$$

for any real numbers $a_k > 0$, b_k , and $t > 0$. Generalisations of this identity play an important role in the theory of the Hilbert transform.

Symbolic logic

In 1847 Boole published the pamphlet *Mathematical Analysis of Logic*. He later regarded it as a flawed exposition of his logical system, and wanted *An Investigation of the Laws of Thought (1854)*, on Which are Founded the Mathematical Theories of Logic and Probabilities to be seen as the mature statement of his views. Boole's initial involvement in logic was prompted by a current debate on quantification, between Sir William Hamilton who supported the theory of "quantification of the predicate", and Boole's supporter Augustus De Morgan who advanced a version of De Morgan duality, as it is now called. Boole's approach was ultimately much further reaching than either sides' in the controversy. It founded what was first known as the "algebra of logic" tradition.^[13]

Boole did not regard logic as a branch of mathematics, but he provided a general symbolic method of logical inference. Boole proposed that logical propositions should be expressed by means of algebraic equations. Algebraic manipulation of the symbols in the equations would provide a fail-safe method of logical deduction: i.e. logic is reduced to a type of algebra.^[citation needed]

By 1 (unity) Boole denoted the "universe of thinkable objects"; literal symbols, such as x, y, z, v, u , etc., were used with the "elective" meaning attaching to adjectives and nouns of natural language. Thus, if x = horned and y = sheep, then the successive acts of election (i.e. choice) represented by x and y , if performed on unity, give the class "horned sheep". Thus, $(1 - x)$ would represent the operation of selecting all things in the world except horned things, that is, all not horned things, and $(1 - x)(1 - y)$ would give all things neither horned nor sheep.^[citation needed]

Treatment of addition in logic

Boole conceived of "elective symbols" of his kind as an algebraic structure. But this general concept was not available to him: he did not have the segregation standard in abstract algebra of postulated (axiomatic) properties of operations, and deduced properties.^[14] His work was a beginning to the algebra of sets, again not a concept available to Boole as a familiar model. His pioneering efforts encountered specific difficulties, and the treatment of addition was an obvious difficulty in the early days.

Boole replaced the operation of multiplication by the word 'and' and addition by the word 'or'. But in Boole's original system, + was a partial operation: in the language of set theory it would correspond only to disjoint union of subsets. Later authors changed the interpretation, commonly reading it as exclusive or, or in set theory terms symmetric difference; this step means that addition is always defined.

In fact there is the other possibility, that + should be read as disjunction. This other possibility extends from the disjoint union case, where exclusive or and non-exclusive or both give the same answer. Handling this ambiguity was an early problem of the theory, reflecting the modern use of both Boolean rings and Boolean algebras (which are simply different aspects of one type of structure). Boole and Jevons struggled over just this issue in 1863, in the form of the correct evaluation of $x + x$. Jevons argued for the result x , which is correct for + as disjunction. Boole kept the result as something undefined. He argued against the result 0, which is correct for exclusive or, because he saw the equation $x + x = 0$ as implying $x = 0$, a false analogy with ordinary algebra.

Probability theory

The second part of the *Laws of Thought* contained a corresponding attempt to discover a general method in probabilities. Here the goal was algorithmic: from the given probabilities of any system of events, to determine the consequent probability of any other event logically connected with those events.

Legacy

Boolean algebra is named after him, as is the crater Boole on the Moon. The keyword *Bool* represents a Boolean datatype in many programming languages, though Pascal and Java, among others, both use the full name *Boolean*.^[15] The library, underground lecture theatre complex and the Boole Centre for Research in Informatics^[16] at University College Cork are named in his honour.

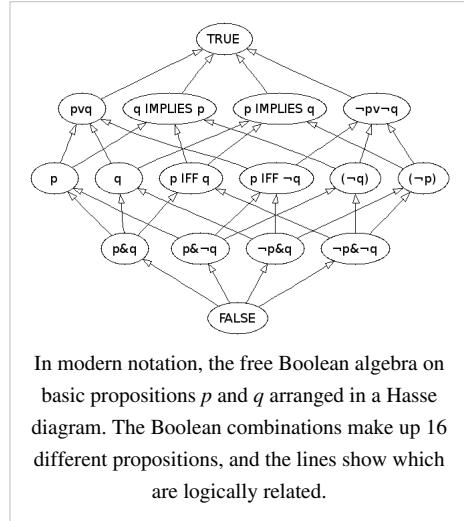
19th-century development

Boole's work was extended and refined by a number of writers, beginning with William Stanley Jevons. Augustus De Morgan had worked on the logic of relations, and Charles Sanders Peirce integrated his work with Boole's during the 1870s.^[17] Other significant figures were Platon Sergeevich Poretskii, and William Ernest Johnson. The conception of a Boolean algebra structure on equivalent statements of a propositional calculus is credited to Hugh MacColl (1877), in work surveyed 15 years later by Johnson. Surveys of these developments were published by Ernst Schröder, Louis Couturat, and Clarence Irving Lewis.

20th-century development

In 1921 the economist John Maynard Keynes published a book on probability theory, *A Treatise of Probability*. Keynes believed that Boole had made a fundamental error in his definition of independence which vitiated much of his analysis.^[18] In his book *The Last Challenge Problem*, David Miller provides a general method in accord with Boole's system and attempts to solve the problems recognised earlier by Keynes and others. Theodore Hailperin showed much earlier that Boole had used the correct mathematical definition of independence in his worked out problems

Boole's work and that of later logicians initially appeared to have no engineering uses. Claude Shannon attended a philosophy class at the University of Michigan which introduced him to Boole's studies. Shannon recognised that Boole's work could form the basis of mechanisms and processes in the real world and that it was therefore highly relevant. In 1937 Shannon went on to write a master's thesis, at the Massachusetts Institute of Technology, in which he showed how Boolean algebra could optimise the design of systems of electromechanical relays then used in telephone routing switches. He also proved that circuits with relays could solve Boolean algebra problems. Employing the properties of electrical switches to process logic is the basic concept that underlies all modern electronic digital computers. Victor Shestakov at Moscow State University (1907–1987) proposed a theory of electric switches based on Boolean logic even earlier than Claude Shannon in 1935 on the testimony of Soviet logicians and mathematicians Yanovskaya, Gaaze-Rapoport, Dobrushin, Lupalov, Medvedev and Uspensky, though they presented their academic theses in the same year, 1938. Wikipedia:Please clarify But the first publication of Shestakov's result took place only in 1941 (in Russian). Hence Boolean algebra became the foundation of practical digital circuit design; and Boole, via Shannon and Shestakov, provided the theoretical grounding for the Digital Age.^[19]



In modern notation, the free Boolean algebra on basic propositions p and q arranged in a Hasse diagram. The Boolean combinations make up 16 different propositions, and the lines show which are logically related.

Views

Boole's views were given in four published addresses: *The Genius of Sir Isaac Newton*; *The Right Use of Leisure*; *The Claims of Science*; and *The Social Aspect of Intellectual Culture*.^[20] The first of these was from 1835, when Charles Anderson-Pelham, 2nd Baron Yarborough gave a bust of Newton to the Mechanics' Institute in Lincoln.^[21] The second justified and celebrated in 1847 the outcome of the successful campaign for early closing in Lincoln, headed by Alexander Leslie-Melville, of Branston Hall.^[22] *The Claims of Science* was given in 1851 at Queen's College, Cork. *The Social Aspect of Intellectual Culture* was also given in Cork, in 1855 to the Cuvierian Society.

Boole read a wide variety of Christian theology. Combining his interests in mathematics and theology, he compared the Christian trinity of Father, Son, and Holy Ghost with the three dimensions of space, and was attracted to the Hebrew conception of God as an absolute unity. Boole considered converting to Judaism but in the end was said to have chosen Unitarianism. However, his biographer, Des MacHale, describes him as an "agnostic deist".

Two influences on Boole were later claimed by his wife, Mary Everest Boole: a universal mysticism tempered by Jewish thought, and Indian logic.^[23] Mary Boole stated that an adolescent mystical experience provided for his life's work:

My husband told me that when he was a lad of seventeen a thought struck him suddenly, which became the foundation of all his future discoveries. It was a flash of psychological insight into the conditions under which a mind most readily accumulates knowledge [...] For a few years he supposed himself to be convinced of the truth of "the Bible" as a whole, and even intended to take orders as a clergyman of the

English Church. But by the help of a learned Jew in Lincoln he found out the true nature of the discovery which had dawned on him. This was that man's mind works by means of some mechanism which "functions normally towards Monism."^[24]

In Ch. 13 of *Laws of Thought* Boole used examples of propositions from Benedict Spinoza and Samuel Clarke. The work contains some remarks on the relationship of logic to religion, but they are slight and cryptic.^[25] Boole was apparently disconcerted at the book's reception just as a mathematical toolset:

George afterwards learned, to his great joy, that the same conception of the basis of Logic was held by Leibnitz, the contemporary of Newton. De Morgan, of course, understood the formula in its true sense; he was Boole's collaborator all along. Herbert Spencer, Jowett, and Leslie Ellis understood, I feel sure; and a few others, but nearly all the logicians and mathematicians ignored [953] the statement that the book was meant to throw light on the nature of the human mind; and treated the formula entirely as a wonderful new method of reducing to logical order masses of evidence about external fact.

Mary Boole claimed that there was profound influence — via her uncle George Everest — of Indian thought on George Boole, as well as on Augustus De Morgan and Charles Babbage:

Think what must have been the effect of the intense Hinduizing of three such men as Babbage, De Morgan, and George Boole on the mathematical atmosphere of 1830–65. What share had it in generating the Vector Analysis and the mathematics by which investigations in physical science are now conducted?

Family

In 1855 he married Mary Everest (niece of George Everest), who later wrote several educational works on her husband's principles.

The Booles had five daughters:

- Mary Lucy Margret (1856–1908)^[26] who married the mathematician and author Charles Howard Hinton and had four children: George (1882–1943), Eric (*1884), William (1886–1909)^[27] and Sebastian (1887–1923) inventor of the Jungle gym. Sebastian had three children:
 - William H. Hinton (1919–2004) visited China in the 1930s and 40s and wrote an influential account of the Communist land reform.
 - Joan Hinton (1921–2010) worked for the Manhattan Project and lived in China from 1948 until her death on 8 June 2010; she was married to Sid Engst.
 - Jean Hinton (married name Rosner) (1917–2002) peace activist.
- Margaret, (1858 – ?) married Edward Ingram Taylor, an artist.
 - Their elder son Geoffrey Ingram Taylor became a mathematician and a Fellow of the Royal Society.
 - Their younger son Julian was a professor of surgery.
- Alicia (1860–1940), who made important contributions to four-dimensional geometry
- Lucy Everest (1862–1905), who was first female professor of chemistry in England
- Ethel Lilian (1864–1960), who married the Polish scientist and revolutionary Wilfrid Michael Voynich and was the author of the novel *The Gadfly*.

References

-  Chisholm, Hugh, ed. (1911). "Boole, George". *Encyclopædia Britannica* (11th ed.). Cambridge University Press
- Ivor Grattan-Guinness, *The Search for Mathematical Roots 1870–1940*. Princeton University Press. 2000.
- Francis Hill (1974), *Victorian Lincoln*; Google Books ^[28].
- Des MacHale, *George Boole: His Life and Work*. Boole Press ^[29]. 1985.
- Stephen Hawking, *God Created the Integers*. Running Press, Philadelphia. 2007.

Notes

- [1] Hill, p. 149; Google Books (<http://books.google.co.uk/books?id=-A89AAAAIAAJ&pg=PA149>).
- [2] Rhee, Rush. (1954) "George Boole as Student and Teacher. By Some of His Friends and Pupils." *Proceedings of the Royal Irish Academy. Section A: Mathematical and Physical Sciences*. Vol. 57. Royal Irish Academy
- [3] Society for the History of Astronomy, *Lincolnshire*. (<http://www.freewebs.com/sochistastro/lincolnshire.htm>)
- [4] Hill, p. 172 note 2; Google Books (<http://books.google.co.uk/books?id=-A89AAAAIAAJ&pg=PA172>).
- [5] Hill, p. 130 note 1; Google Books (<http://books.google.co.uk/books?id=-A89AAAAIAAJ&pg=PA130>).
- [6] Hill, p. 148; Google Books (<http://books.google.co.uk/books?id=-A89AAAAIAAJ&pg=PA148>).
- [7] Ronald Calinger, *Vita mathematica: historical research and integration with teaching* (1996), p. 292; Google Books (<http://books.google.co.uk/books?id=D21wKHoYGg0C&pg=PA292>).
- [8] Hill, p. 138 note 4; Google Books (<http://books.google.co.uk/books?id=-A89AAAAIAAJ&pg=PA138>).
- [9] A list of Boole's memoirs and papers is in the *Catalogue of Scientific Memoirs* published by the Royal Society, and in the supplementary volume on differential equations, edited by Isaac Todhunter. To the *Cambridge Mathematical Journal* and its successor, the *Cambridge and Dublin Mathematical Journal*, Boole contributed 22 articles in all. In the third and fourth series of the *Philosophical Magazine* are found 16 papers. The Royal Society printed six memoirs in the *Philosophical Transactions*, and a few other memoirs are to be found in the *Transactions of the Royal Society of Edinburgh* and of the Royal Irish Academy, in the *Bulletin de l'Académie de St-Pétersbourg* for 1862 (under the name G. Boldt, vol. iv. pp. 198–215), and in *Crell's Journal*. Also included is a paper on the mathematical basis of logic, published in the *Mechanic's Magazine* in 1848.
- [10] Andrei Nikolaevich Kolmogorov, Adolf Pavlovich Yushkevich (editors), *Mathematics of the 19th Century: function theory according to Chebyshev, ordinary differential equations, calculus of variations, theory of finite differences* (1998), pp. 130–2; Google Books (<http://books.google.co.uk/books?id=Mw6JMdZQO-wC&pg=PA130>).
- [11] Jeremy Gray, Karen Hunger Parshall, *Episodes in the History of Modern Algebra (1800–1950)* (2007), p. 66; Google Books (<http://books.google.co.uk/books?id=zMSI6QLIJZsC&pg=PA66>).
- [12] George Boole, *The Mathematical Analysis of Logic, Being an Essay towards a Calculus of Deductive Reasoning* (<http://books.google.com/books?id=zv4YAQAAIAAJ&pg=PP9#v=onepage&q&f=false>) (London, England: Macmillan, Barclay, & Macmillan, 1847).
- [13] Witold Marciszewski (editor), *Dictionary of Logic as Applied in the Study of Language* (1981), pp. 194–5.
- [14] Andrei Nikolaevich Kolmogorov, Adolf Pavlovich Yushkevich, *Mathematics of the 19th Century: mathematical logic, algebra, number theory, probability theory* (2001), pp. 15 (note 15)–16; Google Books (<http://books.google.co.uk/books?id=X3u5hJCkobYC&pg=PA15>).
- [15] P. J. Brown, *Pascal from Basic*, Addison-Wesley, 1982. ISBN 0-201-13789-5, page 72
- [16] Boole Centre for Research in Informatics (<http://www.bcri.ucc.ie>)
- [17] Ivor Grattan-Guinness, Gérard Bornet, *George Boole: Selected manuscripts on logic and its philosophy* (1997), p. xlvi; Google Books (<http://books.google.co.uk/books?id=pzg7UFsIVJIC&pg=PR46>).
- [18] Chapter XVI, p. 167, section 6 of *A treatise on probability*, volume 4: "The central error in his system of probability arises out of his giving two inconsistent definitions of 'independence' (2) He first wins the reader's acquiescence by giving a perfectly correct definition: "Two events are said to be independent when the probability of either of them is unaffected by our *expectation* of the occurrence or failure of the other." (3) But a moment later he interprets the term in quite a different sense; for, according to Boole's second definition, we must regard the events as independent unless we are told either that they *must* concur or that they *cannot* concur. That is to say, they are independent unless we know for certain that there is, in fact, an invariable connection between them. "The simple events, x , y , z , will be said to be *conditioned* when they are not free to occur in every possible combination; in other words, when some compound event depending upon them is precluded from occurring. ... Simple unconditioned events are by definition independent." (1) In fact as long as xz is *possible*, x and z are independent. This is plainly inconsistent with Boole's first definition, with which he makes no attempt to reconcile it. The consequences of his employing the term independence in a double sense are far-reaching. For he uses a method of reduction which is only valid when the arguments to which it is applied are independent in the first sense, and assumes that it is valid if they are independent in second sense. While his theorems are true if all propositions or events involved are independent in the first sense, they are not true, as he supposes them to be, if the events are independent only in the second sense."
- [19] "That dissertation has since been hailed as one of the most significant master's theses of the 20th century. To all intents and purposes, its use of binary code and Boolean algebra paved the way for the digital circuitry that is crucial to the operation of modern computers and telecommunications equipment."

- [20] 1902 *Britannica* article by Jevons; online text. (<http://www.1902encyclopedia.com/B/BOO/george-boole.html>)
- [21] James Gasser, *A Boole Anthology: recent and classical studies in the logic of George Boole* (2000), p. 5; Google Books (<http://books.google.co.uk/books?id=A2Q5Yghl000C&pg=PA5>).
- [22] Gasser, p. 10; Google Books (<http://books.google.co.uk/books?id=A2Q5Yghl000C&pg=PA10>).
- [23] Jonardon Ganeri (2001), *Indian Logic: a reader*, Routledge, p. 7, ISBN 0-7007-1306-9; Google Books (http://books.google.co.uk/books?id=t_nOiqFmxOIC&pg=PA7).
- [24] Boole, Mary Everest *Indian Thought and Western Science in the Nineteenth Century*, Boole, Mary Everest *Collected Works* eds. E. M. Cobham and E. S. Dummer, London, Daniel 1931 pp.947–967
- [25] Grattan-Guinness and Bornet, p. 16; Google Books (<http://books.google.co.uk/books?id=pgz7UFsIVJIC&pg=PR16>).
- [26] 'My Right To Die', *Woman Kills Self* in *The Washington Times* v. 28 May 1908 (PDF (<http://chroniclingamerica.loc.gov/lccn/sn84026749/1908-05-28/ed-1/seq-1.pdf>)); *Mrs. Mary Hinton A Suicide* in *The New York Times* v. 29 May 1908 (PDF (<http://query.nytimes.com/mem/archive-free/pdf?res=9E02E5DB1631E233A2575AC2A9639C946997D6CF>)).
- [27] *Smothers In Orchard* in *The Los Angeles Times* v. 27 February 1909.
- [28] <http://books.google.co.uk/books?id=-A89AAAAIAAJ&pg=PA149>
- [29] <http://boolepress.com/>

External links

- Roger Parsons' article on Boole (<http://www.rogerparsons.info/george/boole.html>)
- Works by George Boole (http://www.gutenberg.org/author/George_Boole) at Project Gutenberg
- George Boole's work as first Professor of Mathematics in University College, Cork, Ireland (http://www.ucc.ie/academic/undersci/pages/sci_georgeboole.htm)

Ivan Ivanovich Zhegalkin

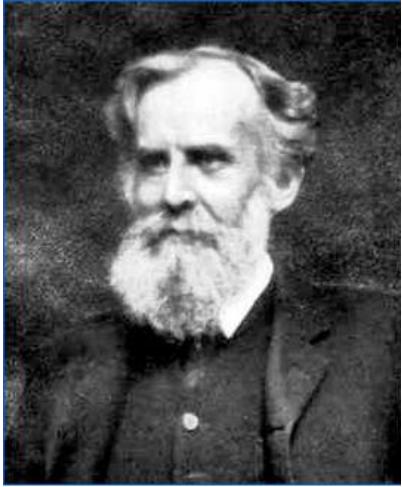
Ivan Ivanovich Zhegalkin (Russian Ива́н Ива́нович Жега́лкин; alternative romanizations: Žegalkin, Gégaline) (August 3, 1869, Mtsensk – March 28, 1947, Moscow) was a Russian mathematician. He is best known for his formulation of Boolean algebra as the theory of the ring of integers mod 2, via what are now called Zhegalkin polynomials.

Zhegalkin was professor of mathematics at Moscow State University. He helped found the thriving mathematical logic group there, which became the Department of Mathematical Logic established by Sofia Janovskaja in 1959. Reminiscing on his student days, Nikolai Luzin recalls Zhegalkin as the only professor he was not afraid of.

References

- Volkov, V.A. (1997). "Two letters of N.N. Luzin to M.Ya. Vygodskii". *Historico-Mathematical Investigation* 2: 133–152.
- Vladimirov, D.A. (1969). *булевы алгебры* (Boolean algebras, in Russian, German translation *Boolesche Algebren* 1974). Nauka (German translation Akademie-Verlag).
- Zhegalkin, Ivan Ivanovich (1927). "On the Technique of Calculating Propositions in Symbolic Logic". *Matematicheskii Sbornik* 43: 9–28.

John Venn

John Venn	
	
Born	4 August 1834 Kingston upon Hull, Yorkshire, England
Died	4 April 1923 (aged 88) Cambridge, England
Nationality	English
Fields	Mathematics Logic Philosophy
Institutions	University of Cambridge
Alma mater	University of Cambridge
Notable awards	Fellow of the Royal Society

John Venn FRS (4 August 1834 – 4 April 1923), was a British logician and philosopher. He is famous for introducing the Venn diagram, which is used in many fields, including set theory, probability, logic, statistics, and computer science.

Life and work

John Venn was born in 1834 in Hull, Yorkshire. His mother, Martha Sykes, came from Swanland, near Hull, and died while John was only three. His father was the Rev. Henry Venn who, at the time of John's birth, was the rector of the parish of Drypool near Hull. Henry Venn, a fellow of Queens', was from a family of distinction. His own father, John's grandfather, was the Reverend John Venn who had been the rector of Clapham in south London. He was a leader of the Clapham Sect, a group of evangelical Christians centered on his church who campaigned for prison reform and the abolition of slavery and cruel sports.

His son described him thus: "*Of spare build, he was throughout his life a fine walker and mountain climber, a keen botanist, and an excellent talker and linguist.*"

John Venn's father (Henry) also played a prominent role in the evangelical Christian movement. The Society for Missions to Africa and the East was founded by evangelical clergy of the Church of England in 1799, and in 1812 it was renamed the Church Missionary Society for Africa and the East. Henry Venn was secretary to this Society from 1841. He moved to Highgate near London in order to carry out his duties and held this position until his death in 1873.

John Venn was brought up strictly. It was expected that he would follow the family tradition into the Christian ministry. After Highgate School, he entered Gonville and Caius College, Cambridge, in 1853. He was graduated in 1857 and shortly afterward was elected a fellow of the college. He was ordained as a deacon at Ely in 1858 and became a priest in 1859. In 1862 he returned to Cambridge as a lecturer in moral sciences.

Venn also had a rare skill in building machines. He used his skill to build a machine for bowling cricket balls, which was so good that when the Australian Cricket team visited Cambridge in 1909, Venn's machine clean bowled one of its top stars four times.

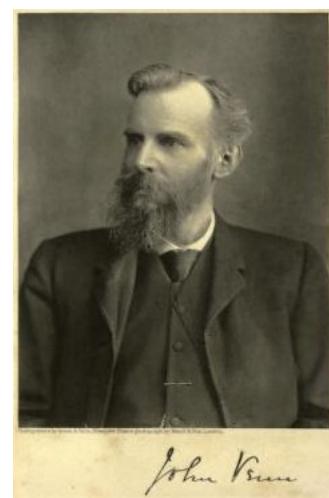
Venn's main area of interest was logic and he published three texts on the subject. He wrote *The Logic of Chance* which introduced the frequency interpretation or frequency theory of probability in 1866, *Symbolic Logic* which introduced the Venn diagrams in 1881, and *The Principles of Empirical Logic* in 1889.

In 1883, Venn was elected to the Royal Society. In 1897, he wrote a history of his college, called *The Biographical History of Gonville and Caius College, 1849–1897*. He began a compilation of biographical notes of Cambridge University alumni, a work which was continued by his son, John Archibald Venn (1883–1958) and published as *Alumni Cantabrigienses* in 10 volumes from 1922 to 1953.

John Venn died in 1923 at Cambridge, and was buried nearby at Trumpington Churchyard (Extension).

In 1867 Venn married Susanna Carnegie Edmonstone, the daughter of the Rev. Charles Edmonstone. They had one child, a son, John Archibald Venn, who in 1932 became president of Queens' College, Cambridge. The Venns, father and son, wrote *Alumni Cantabrigienses* together (see below).

The annals of a clerical family (1904) traces the Venn family history back to the seventeenth century – he was the eighth generation of his family to have a university education. In 1910 he published a work on historical biography, namely a treatise on John Caius, one of the founders of his College. Three years later he published *Early Collegiate Life* which collected many of his writings describing what life was like in the early days of Cambridge University. Working with his son he then started the immense task of compiling a history of Cambridge University *Alumni Cantabrigienses*. The first volume, published in 1922, contained 76,000 names and covered the period up to 1751. It was "nothing less than a 'biographical list of all known students, graduates, and holders of office at the University of Cambridge from the earliest times to 1900'. ... The Venns, father and son, spared no industry in building up these



Signature of John Venn.

records, which are of extraordinary value to historians and genealogists ..."'

Memorials

- Venn is commemorated at the University of Hull by the Venn Building, built in 1928.
- A stained glass window in the dining hall of Gonville and Caius College, Cambridge, commemorates Venn's work.

Selected works

- Venn, John (January 1876). "Consistency and Real Inference" [1]. *Mind* 1 (1).
- Venn, John (1881). *Symbolic Logic* [2]. London: Macmillan and Company. ISBN 1-4212-6044-1.
- Venn, John (1880). "On the Employment of Geometrical Diagrams for the Sensible Representation of Logical Propositions" [3]. *Proceedings of the Cambridge Philosophical Society* 4: 47–59.
- *The Logic of Chance: An Essay on the Foundations and Province of the Theory of Probability*
 - First Edition (1866): Google Book Search [4]
 - Second Edition (1876): Google Book Search [5] or Internet Archive [6] or University of Göttingen [7]
 - Third Edition (1888): Google Book Search [8] or Internet Archive [9]
- Venn, John (1901). *Caius College* [10]. London: F. E. Robinson & Co.
- John Caius, John Venn (1904). *The Annals of Gonville and Caius College* [11]. Printed for the Cambridge Antiquarian Society, sold by Deighton, Bell & Co. (1904) – by John Caius, edited by John Venn



The Venn Building, University of Hull.



Stained glass window at Gonville and Caius College, Cambridge, commemorating Venn and the Venn diagram.

References

- [1] <http://fair-use.org/mind/1876/01/consistency-and-real-inference>
- [2] <http://books.google.com/?id=nisCAAAAQAAJ&printsec=frontcover&dq=john+venn>
- [3] <http://books.google.com/?id=GpI1AAAAIAAJ&pg=PA47&dq=john+venn>
- [4] http://books.google.com/books?id=vMBVEVc_M3MC&dq=editions:09V3JZZfabqkCAexyAoexy&lr=&source=gbz_summary_s&cad=0
- [5] http://books.google.com/books?id=TaMZAAMAAJ&dq=editions:09V3JZZfabqkCAexyAoexy&lr=&source=gbz_summary_s&cad=0
- [6] <http://www.archive.org/details/50424309>
- [7] http://gdz.sub.uni-goettingen.de/no_cache/dms/load/img/?IDDOC=59523
- [8] http://books.google.com/books?id=RR0jgsRmVZsC&dq=john+venn&as_brr=1&source=gbz_summary_s&cad=0
- [9] <http://www.archive.org/details/logicofchance029416mbp>
- [10] <http://books.google.com/?id=1A4BAAAAYAAJ&printsec=frontcover&dq=john+venn>
- [11] <http://books.google.com/?id=qJmUDQaAHBwC&printsec=frontcover&dq=john+venn>

External links

- The Venn archives (<http://www.mundus.ac.uk/cats/44/1206.htm>) clarify the confusing timeline of the various Venns.
- Obituary of John Venn (<http://www-groups.dcs.st-and.ac.uk/~history/Obits/Venn.html>) (New York Times)
- Portrait of Venn (<http://www.theory.cs.uvic.ca/~cos/venn/VennPaintEJC.html>) by Charles Brock, and a link to a site about Venn
- Another (clearer) view of the Venn stained glass window (<http://www.theory.cs.uvic.ca/~cos/venn/gifs/VennStain.gif>)
- John Venn (<http://www.findagrave.com/cgi-bin/fg.cgi?page=gr&GRid=14054755>) at *Find a Grave*

Marshall Harvey Stone

Marshall Harvey Stone	
Born	April 8, 1903 New York City
Died	January 9, 1989 Madras
Citizenship	American
Fields	Real analysis, Functional analysis, Boolean algebra
Institutions	Harvard, U. Chicago, U. of Massachusetts
Alma mater	Harvard
Doctoral advisor	G. D. Birkhoff
Doctoral students	Holbrook MacNeille, Harvard, 1935 John Williams Calkin, Harvard, 1937 William Frederick Eberlein, Harvard, 1942 Edwin Hewitt, Harvard, 1942 George Mackey, Harvard, 1942 Michael Joseph Norris, Harvard, 1944 Richard V. Kadison, U. Chicago, 1950 John Verner Finch, U. Chicago, 1951 Matthew P. Gaffney, Jr., U. Chicago, 1951 Bernard A. Galler, U. Chicago, 1955 John J. McKibben, U. Chicago, 1957 Royal Bruce Kellogg, U. Chicago, 1958 Adam Koranyi, U. Chicago, 1959 Christopher Ian Byrnes, U. of Massachusetts, 1975
Known for	Stone–von Neumann theorem, Stone–Čech compactification, Stone–Weierstrass theorem

Marshall Harvey Stone (April 8, 1903, New York City – January 9, 1989, Madras, India) was an American mathematician who contributed to real analysis, functional analysis, and the study of Boolean algebras.

Biography

Stone was the son of Harlan Fiske Stone, who was the Chief Justice of the United States in 1941–1946. Marshall Stone's family expected him to become a lawyer like his father, but he became enamored of mathematics while he was a Harvard University undergraduate. He completed a Harvard Ph.D. in 1926, with a thesis on differential equations that was supervised by George David Birkhoff. Between 1925 and 1937, he taught at Harvard, Yale University, and Columbia University. Stone was promoted to a full Professor at Harvard in 1937.

During World War II, Stone did classified research as part of the "Office of Naval Operations" and the "Office of the Chief of Staff" of the United States Department of War. In 1946, he became the chairman of the Mathematics Department at the University of Chicago, a position that he held until 1952. He remained on the faculty at this university until 1968, after which he taught at the University of Massachusetts Amherst until 1980.

The department he joined in 1946 was in the doldrums, after having been at the turn of the 20th century arguably the best American mathematics department, thanks to the leadership of Eliakim Hastings Moore. Stone did an outstanding job of making the Chicago department eminent again, mainly by hiring Paul Halmos, André Weil, Saunders Mac Lane, Antoni Zygmund, and Shiing-Shen Chern.

Accomplishments

During the 1930s, Stone did much important work:

- In 1930, he proved the celebrated Stone–von Neumann uniqueness theorem.
- In 1932, he published a classic monograph 662 pages long titled *Linear transformations in Hilbert space and their applications to analysis*, a presentation about self-adjoint operators. Much of its content is now deemed to be part of functional analysis.
- In 1932, he proved conjectures by Hermann Weyl on spectral theory, arising from the application of group theory to quantum mechanics.
- In 1934, he published two papers setting out what is now called Stone–Čech compactification theory. This theory grew out of his attempts to understand more deeply his results on spectral theory.
- In 1936, he published a long paper that included Stone's representation theorem for Boolean algebras, an important result in mathematical logic and universal algebra.
- The Stone–Weierstrass theorem substantially generalized Weierstrass's theorem on the uniform approximation of continuous functions by polynomials.

Stone was elected to the National Academy of Sciences (United States) in 1938. He presided over the American Mathematical Society, 1943–44, and the International Mathematical Union, 1952–54. In 1982, he was awarded the National Medal of Science.^[1]

Selected publications

- "A comparison of the series of Fourier and Birkhoff". *Trans. Amer. Math. Soc.* **28** (4): 695–761. 1926. MR 1501372^[2].
- *Linear transformations in Hilbert space and their applications to analysis*. New York: American Mathematical Society. 1932.
- "Boolean algebras and their applications to topology". *Proc Natl Acad Sci U S A* **20** (3): 197–202. 1934. PMC 1076376^[3].
- *The theory of real functions*. Ann Arbor: Edwards Brothers. 1940.
- "Mathematics and the future of science". *Bull. Amer. Math. Soc.* **63** (2): 61–76. 1957. MR 0086013^[4].
- *Lectures on preliminaries to functional analysis*. Madras: Institute of Mathematical Sciences. 1963. Notes by B. Ramachandran (50 pages)

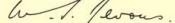
References

- [1] National Science Foundation - The President's National Medal of Science (http://www.nsf.gov/od/nms/recip_details.cfm?recip_id=348)
- [2] <http://www.ams.org/mathscinet-getitem?mr=1501372>
- [3] <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1076376>
- [4] <http://www.ams.org/mathscinet-getitem?mr=0086013>

External links

- O'Connor, John J.; Robertson, Edmund F., "Marshall Harvey Stone" (<http://www-history.mcs.st-andrews.ac.uk/Biographies/Stone.html>), *MacTutor History of Mathematics archive*, University of St Andrews.
- Marshall Harvey Stone (<http://genealogy.math.ndsu.nodak.edu/id.php?id=4946>) at the Mathematics Genealogy Project
- Johnstone, Peter (1982). *Stone Spaces*. Cambridge: Cambridge University Press. ISBN 0-521-23893-5.

William Stanley Jevons

William Stanley Jevons	
	
Born	1 September 1835 Liverpool, UK
Died	13 August 1882 (aged 46) Bexhill near Hastings, UK
Fields	Economics Logic
Institutions	University College London 1876–80 Owens College (now University of Manchester) 1863–1875
Alma mater	University College London
Academic advisors	Augustus De Morgan
Known for	Marginal utility theory
Influences	Jeremy Bentham
Influenced	Frank Fetter Alfred Marshall Irving Fisher Lionel Robbins
Signature	
	
Notes	
While not a formal advisor (Jevons never acquired a PhD), De Morgan was his most influential professor. ^[1]	

William Stanley Jevons, LL.D., M.A., F.R.S. (/dʒɛvənz/,^[2] 1 September 1835 – 13 August 1882) was a British economist and logician.

Irving Fisher described his book *A General Mathematical Theory of Political Economy* (1862) as the start of the mathematical method in economics.^[3] It made the case that economics as a science concerned with quantities is necessarily mathematical.^[4] In so doing, it expounded upon the "final" (marginal) utility theory of value. Jevons' work, along with similar discoveries made by Carl Menger in Vienna (1871) and by Léon Walras in Switzerland (1874), marked the opening of a new period in the history of economic thought. Jevons' contribution to the marginal

revolution in economics in the late 19th century established his reputation as a leading political economist and logician of the time.

Jevons broke off his studies of the natural sciences in London in 1854 to work as an assayer in Sydney, where he acquired an interest in political economy. Returning to the UK in 1859, he published *General Mathematical Theory of Political Economy* in 1862, outlining the marginal utility theory of value, and *A Serious Fall in the Value of Gold* in 1863. For Jevons, the utility or value to a consumer of an additional unit of a product is inversely related to the number of units of that product he already owns, at least beyond some critical quantity.

It was for *The Coal Question* (1865), in which he called attention to the gradual exhaustion of the UK's coal supplies, that he received public recognition, in which he put forth what is now known as Jevon's paradox, i.e. that increases in energy production efficiency leads to more not less consumption. The most important of his works on logic and scientific methods is his *Principles of Science* (1874),^[5] as well as *The Theory of Political Economy* (1871) and *The State in Relation to Labour* (1882).

Background

Jevons was born in Liverpool, in the UK. His father, Thomas Jevons, a man of strong scientific tastes and a writer on legal and economic subjects, was an iron merchant. His mother Mary Anne Jevons was the daughter of William Roscoe. At the age of fifteen he was sent to London to attend University College School. He appears at this time to have already formed the belief that important achievements as a thinker were possible to him, and at more than one critical period in his career this belief was the decisive factor in determining his conduct. Towards the end of 1853, after having spent two years at University College, where his favourite subjects were chemistry and botany, he unexpectedly received the offer of the assayership to the new mint in Australia. The idea of leaving the UK was distasteful, but pecuniary considerations had, in consequence of the failure of his father's firm in 1847, become of vital importance, and he accepted the post.

He left the UK for Sydney in June 1854, and remained there for five years.^[6] At the end of that period he resigned his appointment, and in the autumn of 1859 entered again as a student at University College London, proceeding in due course to the B.A. and M.A. degrees of the University of London. He now gave his principal attention to the moral sciences, but his interest in natural science was by no means exhausted: throughout his life he continued to write occasional papers on scientific subjects, and his intimate knowledge of the physical sciences greatly contributed to the success of his chief logical work, *The Principles of Science*. Not long after taking his M.A. degree Jevons obtained a post as tutor at Owens College, Manchester.

In 1866 he was elected professor of logic and mental and moral philosophy and Cobden professor of political economy in Owens college. Next year he married Harriet Ann Taylor, whose father, John Edward Taylor, had been the founder and proprietor of the Manchester Guardian. Jevons suffered a good deal from ill health and sleeplessness, and found the delivery of lectures covering so wide a range of subjects very burdensome. In 1876 he was glad to exchange the Owens professorship for the professorship of political economy in University College, London. Travelling and music were the principal recreations of his life; but his health continued to be bad, and he suffered from depression. He found his professorial duties increasingly irksome, and feeling that the pressure of literary work left him no spare energy, he decided in 1880 to resign the post. On 13 August 1882 he was drowned whilst bathing near Hastings.

Jevons was a prolific writer, and at the time of his death was a leader in the UK both as a logician and as an economist. Alfred Marshall said of his work in economics that it "will probably be found to have more constructive force than any, save that of Ricardo, that has been done during the last hundred years."

Stanley Jevons was raised a Christian Unitarian,^[7] and excerpts from his journals indicate he remained committed to his Christian beliefs until death. After dying in 1882 from accidental drowning, he was buried in the Hampstead Cemetery.^[8]

Theory of utility



Portrait of W. Stanley Jevons at 42, by G. F. Stodart

Jevons arrived quite early in his career at the doctrines that constituted his most characteristic and original contributions to economics and logic. The theory of utility, which became the keynote of his general theory of political economy, was practically formulated in a letter written in 1860; and the germ of his logical principles of the substitution of similars may be found in the view which he propounded in another letter written in 1861, that "philosophy would be found to consist solely in pointing out the likeness of things." The theory of utility above referred to, namely, that the degree of utility of a commodity is some continuous mathematical function of the quantity of the commodity available, together with the implied doctrine that economics is essentially a mathematical science, took more definite form in a paper on "A General Mathematical Theory of Political Economy", written for the British Association in 1862. This paper does not appear to have attracted much attention either in 1862 or on its publication four years later in the *Journal of the Statistical Society*; and

it was not till 1871, when the *Theory of Political Economy* appeared, that Jevons set forth his doctrines in a fully developed form.

It was not till after the publication of this work that Jevons became acquainted with the applications of mathematics to political economy made by earlier writers, notably Antoine Augustin Cournot and HH Gossen. The theory of utility was at about 1870 being independently developed on somewhat similar lines by Carl Menger in Austria and Léon Walras in Switzerland. As regards the discovery of the connection between value in exchange and final (or marginal) utility, the priority belongs to Gossen, but this in no way detracts from the great importance of the service which Jevons rendered to British economics by his fresh discovery of the principle, and by the way in which he ultimately forced it into notice. In his reaction from the prevailing view he sometimes expressed himself without due qualification: the declaration, for instance, made at the commencement of the *Theory of Political Economy*, that value depends entirely upon utility, lent itself to misinterpretation. But a certain exaggeration of emphasis may be pardoned in a writer seeking to attract the attention of an indifferent public. The Neoclassical Revolution, which would reshape economics, had been started.

Jevons did not explicitly distinguish between the concepts of ordinal and cardinal utility. Cardinal utility allows the relative magnitude of utilities to be discussed, while ordinal utility only implies that goods can be compared and ranked according to which good provided the most utility. Although Jevons predated the debate about ordinality or cardinality of utility, his mathematics required the use of cardinal utility functions. For example, in The Theory of Political Economy, Chapter II, the subsection on "Theory of Dimensions of Economic Quantities", Jevons makes the statement that "In the first place, pleasure and pain must be regarded as measured upon the same scale, and as having, therefore, the same dimensions, being quantities of the same kind, which can be added and subtracted...." Speaking of measurement, addition, and subtraction requires cardinality, as does Jevons' heavy use of integral calculus. Note that cardinality does not imply direct measurability, in which Jevons did not believe.

Practical economics

It was not, however, as a theorist dealing with the fundamental data of economic science, but as a brilliant writer on practical economic questions, that Jevons first received general recognition. *A Serious Fall in the Value of Gold* (1863) and *The Coal Question* (1865) placed him in the front rank as a writer on applied economics and statistics; and he would be remembered as one of the leading economists of the 19th century even had his *Theory of Political Economy* never been written. Amongst his economic works may be mentioned *Money and the Mechanism of Exchange* (1875), written in a popular style, and descriptive rather than theoretical, but wonderfully fresh and original in treatment and full of suggestiveness, a *Primer on Political Economy* (1878), *The State in Relation to Labour* (1882), and two works published after his death, namely, *Methods of Social Reform and Investigations in Currency and Finance*, containing papers that had appeared separately during his lifetime. The last-named volume contains Jevons' speculations on the connection between commercial crises and sunspots. He was engaged at the time of his death upon the preparation of a large treatise on economics and had drawn up a table of contents and completed some chapters and parts of chapters. This fragment was published in 1905 under the title of *The Principles of Economics: a fragment of a treatise on the industrial mechanism of society, and other papers*.

In *The Coal Question*, Jevons covered a breadth of concepts on energy depletion that have recently been revisited by writers covering the subject of peak oil. For example, Jevons explained that improving energy efficiency typically reduced energy costs and thereby increased rather than decreased energy use, an effect now known as the Jevons paradox. *The Coal Question* remains a paradigmatic study of resource depletion theory. Jevons's son, H. Stanley Jevons, published an 800-page follow-up study in 1915 in which the difficulties of estimating recoverable reserves of a theoretically finite resource are discussed in detail.^[9]

In a relatively minor work, "Commercial Crises and Sun-Spots",^[10] Jevons analyzed business cycles, proposing that crises in the economy might not be random events, but might be based on discernible prior causes. To clarify the concept, he presented a statistical study relating business cycles with sunspots. His reasoning was that sunspots affected the weather, which, in turn, affected crops. Crop changes could then be expected to cause economic changes. Subsequent studies have found that sunny weather has a small but significant positive impact on stock returns, probably due to its impact on traders' moods.^[11]



Portrait of Jevons published in the *Popular Science Monthly* in 1877

Logic



The Logic Piano

Jevons' work in logic went on *pari passu* with his work in political economy. In 1864 he published a small volume, entitled *Pure Logic; or, the Logic of Quality apart from Quantity*, which was based on Boole's system of logic, but freed from what he considered the false mathematical dress of that system. In the years immediately following he devoted considerable attention to the construction of a logical machine, exhibited before the Royal Society in 1870, by means of which the conclusion derivable from any given set of premisses could be mechanically obtained. In 1866 what he regarded as the great and universal principle of all reasoning dawned upon him; and in 1869 he published a sketch of this fundamental doctrine under the title of *The Substitution of Similars*. He expressed the principle in its simplest form as follows: "Whatever is true of a thing is true of its like", and he worked out in detail its various applications including the "Logic Piano", a mechanical computer he designed and had built in 1869.

In the following year appeared the *Elementary Lessons on Logic*, which soon became the most widely read elementary textbook on logic in the English language. In the meantime he was engaged upon a much more important logical treatise, which appeared in 1874 under the title of *The Principles of Science*. In this work Jevons embodied the substance of his earlier works on pure logic and the substitution of similars; he also enunciated and developed the view that induction is simply an inverse employment of deduction; he treated in a luminous manner the general theory of probability, and the relation between probability and induction; and his knowledge of the various natural sciences enabled him throughout to relieve the abstract character of logical doctrine by concrete scientific illustrations, often worked out in great detail. An example is his discussion of the use of one-way functions in cryptography, including remarks on the integer factorization problem that foreshadowed its use in public key cryptography. Jevons' general theory of induction was a revival of the theory laid down by Whewell and criticized by John Stuart Mill; but it was put in a new form, and was free from some of the non-essential adjuncts which rendered Whewell's exposition open to attack. The work as a whole was one of the most notable contributions to logical doctrine that appeared in the UK in the 19th century. His *Studies in Deductive Logic*, consisting mainly of exercises and problems for the use of students, was published in 1880. In 1877 and the following years Jevons contributed to the *Contemporary Review* some articles on Mill, which he had intended to supplement by further articles, and eventually publish in a volume as a criticism of Mill's philosophy. These articles and one other were republished after Jevons' death, together with his earlier logical treatises, in a volume, entitled *Pure Logic, and other Minor Works*. The criticisms on Mill contain much that is ingenious and much that is forcible, but on the whole they cannot be regarded as taking rank with Jevons's other work. His strength lay in his power as an original thinker rather than as a critic; and he will be remembered by his constructive work as logician, economist and statistician.

On Jevons as logician, see Grattan-Guinness (2000).



The Logic Piano keyboard

Number theory

Jevons had written in his *Principles of Science*:^[12] "Can the reader say what two numbers multiplied together will produce the number 8616460799 ? I think it unlikely that anyone but myself will ever know." This became known as Jevons' Number and was factored by Derrick Norman Lehmer in 1903^[13] and later on a pocket calculator by Solomon W. Golomb.^[14]

Works

- 1862. *A General Mathematical Theory of Political Economy*
- 1863. *A Serious Fall in the Value of Gold*,^[15] Edward Stanford.
- 1864. *Pure Logic; or, the Logic of Quality apart from Quantity*
- 1865. *The Coal Question*, Macmillan and Co.^[16]
- 1869. *The Substitution of Similars, The True Principle of Reasoning*,^[17] Macmillan & Co.
- 1870. *Elementary Lessons on Logic*
- 1871. *The Match Tax: A Problem in Finance*, Edward Stanford.
- 1871. *The Theory of Political Economy*, Macmillan & Co.
- "Theory of Political Economy."^[18] In James R. Newman, ed., *The World of Mathematics*, Vol. 2, Part IV, 1956.
- 1874. *Principles of Science*, Macmillan & Co.
- 1875. *Money and the Mechanism of Exchange*, D. Appleton and Co.
- 1878. *A Primer on Political Economy*
- 1880. *Studies in Deductive Logic*
- 1882. *The State in Relation to Labour*
- 1883. *Methods of Social Reform and Other Papers*, Macmillan and Co.
 - *Methods of Social Reform, and Other Papers*, Kelley, 1965.
- 1886. *Letters and Journal of W. Stanley Jevons*,^[19] Ed. by Harriet A. Jevons, Macmillan & Co.
- 1972–81. *Papers and Correspondence*, edited by R. D. Collison Black, Macmillan & the Royal Economic Society (7 vol.)

Articles

- "On the Variation of Prices and the Value of the Currency since 1782,"^[20] *Journal of the Statistical Society of London*, Vol. 28, No. 2, Jun., 1865.
- "On the Frequent Autumnal Pressure in the Money Market, and the Action of the Bank of England,"^[21] *Journal of the Statistical Society of London*, Vol. 29, No. 2, Jun., 1866.
- "On the Condition of the Metallic Currency of the United Kingdom, with Reference to the Question of International Coinage,"^[22] *Journal of the Statistical Society of London*, Vol. 31, No. 4, Dec., 1868.
- "Who Discovered the Quantification of the Predicate?,"^[23] *The Contemporary Review*, Vol. XXI, December 1872/May 1873.
- "The Railways and the State."^[24] In *Essays and Addresses*, Macmillan & Co., 1874.
- "The Silver Question,"^[25] *Journal of Social Science*, No. IX, January 1878.
- "John Stuart Mill's Philosophy Tested,"^[26] Part II,^[27] *The Contemporary Review*, Vol. XXXI, December 1877/January 1878; Part III^[28], Vol. XXXII, April 1878.^{[29][30]}
- "Methods of Social Reform, I: Amusements of the People,"^[31] *The Contemporary Review*, Vol. XXXIII, October 1878.
- "Methods of Social Reform, II: A State Parcel Post,"^[32] *The Contemporary Review*, Vol. XXXIV, January 1879.
- "The Periodicity of Commercial Crises, and its Physical Explanation,"^[33] *Journal of The Statistical and Social Inquiry Society of Ireland*, Vol. VII, Part 54, 1878/1879.

- "Experimental Legislation and the Drink Traffic,"^[34] *The Contemporary Review*, Vol. XXXVII, January/June 1880.
- "Recent Mathematico-Logical Memoirs,"^[35] *Nature*, Vol. XXIII, March 24, 1881.
- "Richard Cantillon and the Nationality of Political Economy,"^[36] *The Contemporary Review*, Vol. XXXIX, January/June 1881.
- "The Rationale of Free Public Libraries,"^[37] *The Contemporary Review*, Vol. XXXIX, January/June 1881.
- "Bimetallism,"^[38] *The Contemporary Review*, Vol. XXXIX, January/June 1881.
- "Married Women in Factories,"^[39] *The Contemporary Review*, Vol. XLI, January/June 1882.

Miscellany

- Luigi Cossa, *Guide to the Study of Political Economy*,^[40] with a Preface by W. Stanley Jevons, Macmillan & Co., 1880.

Notes

- [1] R. D. Collison Black (1972). "Jevons, Bentham and De Morgan", *Economica*, New Series, Vol. 39, No. 154, pp. 119–134
- [2] Daniel Jones, *Everyman's English Pronouncing Dictionary* (Dent, Dutton: 13th ed., 1967), p. 266.
- [3] Irving Fisher, 1892. *Mathematical Investigations in the Theory of Value and Prices*, Appendix III, "The Utility and History of Mathematical Method in Economics", p. 109. (http://books.google.com/books?id=zMNGyKoJxv8C&pg=PA109&lpg=PA109&dq=%22The+mathematical+method+really+began+with+Jevons+in+1871.%22&source=bl&ots=i-5Sxjja0B&sig=z2zoAyDtx4qm0LwuJhyB16UjmQ&hl=en&ei=dzgFTKLMNcH7lwePrJHXBg&sa=X&oi=book_result&ct=result&resnum=1&ved=0CBYQ6AEwAA#v=onepage&q=%22The+mathematical+method+really+began+with+Jevons+in+1871.%22&f=false)
- [4] W. Stanley Jevons, 1871. *The Principles of Political Economy*, p. 4.
- [5] Jevons, William Stanley, *The Principles of Science: A Treatise on Logic and Scientific Method*, Macmillan & Co., London, 1874, 2nd ed. 1877, 3rd ed. 1879. Reprinted with a foreword by Ernst Nagel, Dover Publications, New York, 1958.
- [6] Anecdotal History of Annandale (<http://ramin.com.au/annandale/story2-1.shtml#jevons>)
- [7] Mosselmans, Bert, "William Stanley Jevons" (<http://plato.stanford.edu/entries/william-jevons/>), *The Stanford Encyclopedia of Philosophy*
- [8] UVic.ca - University of Victoria (http://web.uvic.ca/~rutherford/mr_grvs1.html)
- [9] Jevons, H. Stanley Jevons, (1915) *The British Coal Trade*. London: Kegan Paul, Trench and Trübner; (complete text available at Google Books) see especially pp. 718 ff.
- [10] Jevons, William Stanley (14 November 1878). "Commercial crises and sun-spots", *Nature* xix, pp. 33–37.
- [11] Hirshleifer, David, and Tyler Shumway (2003). "Good day sunshine: stock returns and the weather", *Journal of Finance* 58 (3), pp. 1009–1032.
- [12] *Principles of Science* (<http://www.archive.org/stream/principlesofscie00jevorich#page/n165/mode/2up>), Macmillan & Co., 1874, p. 141.
- [13] Lehmer, D.N., "A Theorem in the Theory of Numbers" (http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body&id=pdf_1&handle=euclid.bams/1183419373), read before the San Francisco Section of the American Mathematical Society, 19 December 1903.
- [14] Golomb, Solomon. "On Factoring Jevons' Number", *Cryptologia*, vol. XX, no. 3, July, 1996, PP. 243–244.
- [15] <http://www.archive.org/stream/aseriousfallinv00jevogoo>
- [16] Missemér, Antoine. "William Stanley Jevons' The Coal Question (1865), Beyond the Rebound Effect," *Ecological Economics*, Volume 82, October 2012.
- [17] <http://www.archive.org/stream/substitutionofsi00jevorich>
- [18] <http://www.unz.org/Pub/NewmanJames-1957v02-01217>
- [19] <http://www.archive.org/stream/lettersjournalof00jevoiala#page/n7/mode/2up>
- [20] <http://www.jstor.org/stable/2338419>
- [21] <http://www.jstor.org/stable/2338585>
- [22] <http://www.jstor.org/stable/2338797>
- [23] <http://archive.org/stream/contemporaryrev45unkngoog#page/n830/mode/2up>
- [24] <http://archive.org/stream/essaysaddresses00oweniala#page/464/mode/2up>
- [25] <http://archive.org/stream/journalsocialsc03russgoog#page/n38/mode/2up>
- [26] <http://archive.org/stream/contemporaryrev00unkngoog#page/n186/mode/2up>
- [27] <http://archive.org/stream/contemporaryrev00unkngoog#page/n276/mode/2up>
- [28] <http://archive.org/stream/contemporaryrev33unkngoog#page/n98/mode/2up>
- [29] "J. S. Mill's Philosophy Tested by Prof. Jevons," *Mind*, Vol. 3, No. 10, Apr., 1878.
- [30] Jackson, Reginald. "Mill's Treatment of Geometry: A Reply to Jevons," *Mind*, New Series, Vol. 50, No. 197, Jan., 1941.

- [31] <http://archive.org/stream/contemporaryrev36unkngoog#page/n510/mode/2up>
- [32] <http://www.archive.org/stream/contemporaryrev56unkngoog#page/n218/mode/2up>
- [33] http://www.tara.tcd.ie/bitstream/2262/8261/1/jssisiVolVIIPartLIV_334342.pdf
- [34] <http://archive.org/stream/contemporaryrev24unkngoog#page/n188/mode/2up>
- [35] <http://www.archive.org/stream/nature01grougoog#page/n640/mode/2up>
- [36] <http://archive.org/stream/contemporaryrev58unkngoog#page/n62/mode/2up>
- [37] <http://archive.org/stream/contemporaryrev58unkngoog#page/n384/mode/2up>
- [38] <http://archive.org/stream/contemporaryrev58unkngoog#page/n750/mode/2up>
- [39] <http://archive.org/stream/contemporaryrev15unkngoog#page/n46/mode/2up>
- [40] <http://www.archive.org/stream/guidetostudyofpo00cossuoft#page/n5/mode/2up>

References

-  This article incorporates text from a publication now in the public domain: Chisholm, Hugh, ed. (1911). *Encyclopædia Britannica* (11th ed.). Cambridge University Press
- R.D. Collison Black (1987). "Jevons, William Stanley", *The New Palgrave: A Dictionary of Economics*, v. 2, pp. 1008–1014.
- Ivor Grattan-Guinness, 2000. *The Search for Mathematical Roots 1870–1940*. Princeton University Press.
- Terry Peach (1987). "Jevons as an economic theorist", *The New Palgrave: A Dictionary of Economics*, v. 2, pp. 1014–1019.
- The first part of this article was based on an article in the Encyclopedia of Marxism at www.marxists.org (<http://www.marxists.org/>).

Further reading

- Bam, Vincent, et al. "Hypothetical Fallibilism in Peirce and Jevons," *Transactions of the Charles S. Peirce Society*, Vol. 15, No. 2, Spring, 1979.
- Barrett, Lindsay and Connell, Matthew. "Jevons and the Logic 'Piano'," (<http://www.rutherfordjournal.org/article010103.html>) *The Rutherford Journal*, Vol. 1, Issue 1, 2006.
- Collison Black, R. D. "Jevons and Cairnes," *Economica*, New Series, Vol. 27, No. 107, Aug., 1960.
- Collison Black, R. D. "Jevons, Bentham and De Morgan," *Economica*, New Series, Vol. 39, No. 154, May, 1972.
- De Marchi, N. B. "The Noxious Influence of Authority: A Correction of Jevons' Charge," *Journal of Law and Economics*, Vol. 16, No. 1, Apr., 1973.
- Grattan-Guinness, I. "'In Some Parts Rather Rough': A Recently Discovered Manuscript Version of William Stanley Jevons's 'General Mathematical Theory of Political Economy' (1862)," *History of Political Economy*, Vol. 34, Number 4, Winter 2002.
- Jevons, H. Winefrid. "William Stanley Jevons: His Life," *Econometrica*, Vol. 2, No. 3, Jul., 1934.
- Keynes, J. M. "William Stanley Jevons 1835–1882: A Centenary Allocation on his Life and Work as Economist and Statistician," *Journal of the Royal Statistical Society*, Vol. 99, No. 3, 1936.
- Könekamp, Rosamund. "William Stanley Jevons (1835–1882). Some Biographical Notes," *Manchester School of Economic and Social Studies*, Vol. 30, No. 3, Sept. 1962.
- Konvitz, Milton R. "An Empirical Theory of the Labor Movement: W. Stanley Jevons," *The Philosophical Review*, Vol. 57, No. 1, Jan., 1948.
- La Nauze, J. A. "The Conception of Jevon's Utility Theory," *Economica*, New Series, Vol. 20, No. 80, Nov., 1953.
- Maas, Harro. *William Stanley Jevons and the Making of Modern Economics*, Cambridge University Press, 2005.
- Madureira, Nuno Luis. "The Anxiety of Abundance: William Stanley Jevons and Coal Scarcity in the Nineteenth Century," *Environment and History*, Volume 18, Number 3, August 2012.
- Mays, W. and Henry, D. P. "Jevons and Logic," *Mind*, New Series, Vol. 62, No. 248, Oct., 1953.

- Mosselmans, Bert. "William Stanley Jevons and the Extent of Meaning in Logic and Economics," (http://www.eshet.net/public/-1_mosselmans.pdf) *History and Philosophy of Logic*, Volume 19, Issue 2, 1998.
- Mosselmans, Bert. *William Stanley Jevons and the Cutting Edge of Economics*, Routledge, 2007.
- Noller, Carl W. "Jevons on Cost," *Southern Economic Journal*, Vol. 39, No. 1, Jul., 1972.
- Paul, Ellen Frankel. "W. Stanley Jevons: Economic Revolutionary, Political Utilitarian," *Journal of the History of Ideas*, Vol. 40, No. 2, Apr./Jun., 1979.
- Peart, Sandra. "'Disturbing Causes,' 'Noxious Errors,' and the Theory-Practice Distinction in the Economics of J.S. Mill and W.S. Jevons," *The Canadian Journal of Economics*, Vol. 28, No. 4b, Nov., 1995.
- Peart, Sandra. *The Economics of W. S. Jevons*, Routledge, 1996.
- Peart, Sandra. "Jevons and Menger Re-Homogenized?: Jaffé after 20 Years," *The American Journal of Economics and Sociology*, Vol. 57, No. 3, Jul., 1998.
- Peart, Sandra. "Facts Carefully Marshalled' in the Empirical Studies of William Stanley Jevons," *History of Political Economy*, Vol. 33, Annual Supplement, 2001.
- Robertson, Ross M. "Jevons and His Precursors," *Econometrica*, Vol. 19, No. 3, Jul., 1951.
- Schabas, Margaret. "The 'Worldly Philosophy' of William Stanley Jevons," *Victorian Studies*, Vol. 28, No. 1, Autumn, 1984.
- Schabas, Margaret. "Alfred Marshall, W. Stanley Jevons, and the Mathematization of Economics," *Isis*, Vol. 80, No. 1, Mar., 1989.
- Schabas, Margaret. *A World Ruled by Number: William Stanley Jevons and the Rise of Mathematical Economics*, Princeton University Press, 1990.
- Strong, John V. "The Infinite Ballot Box of Nature: De Morgan, Boole, and Jevons on Probability and the Logic of Induction," *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, Vol. 1976, Volume One: Contributed Papers, 1976.
- Wood, John C. *William Stanley Jevons: Critical Assessments*, 2 vol., Routledge, 1988.
- York, Richard. "Ecological Paradoxes: William Stanley Jevons and the Paperless Office," (<http://www.humanecologyreview.org/pastissues/her132/york.pdf>) *Human Ecology Review*, Vol. 13, No. 2, 2006.
- Young, Allyn A. "Jevons' 'Theory of Political Economy'," (<http://www.jstor.org/stable/1804593>) *The American Economic Review*, Vol. 2, No. 3, Sep., 1912.

External links

- O'Connor, John J.; Robertson, Edmund F., "William Stanley Jevons" (<http://www-history.mcs.st-andrews.ac.uk/Biographies/Jevons.html>), *MacTutor History of Mathematics archive*, University of St Andrews.
- New School: William Stanley Jevons (<http://cepa.newschool.edu/het/profiles/jevons.htm>)
- Royal Society certificate of election 1872 (<http://www.royalsoc.ac.uk/DServe/dserve.exe?dsqIni=Dserve.ini&dsqApp=Archive&dsqCmd>Show.tcl&dsqSearch=RefNo==EC/1872/01'&dsqDb=Catalog>)
- Jevons and the Logic 'Piano' (<http://www.rutherfordjournal.org/article010103.html>) at *The Rutherford Journal*
- The Coal Question – Encyclopedia of Earth ([http://www.eoearth.org/article/The_Coal_Question_\(e-book\)](http://www.eoearth.org/article/The_Coal_Question_(e-book)))
- Letters and Journal of W. Stanley Jevons (http://oll.libertyfund.org/index.php?option=com_staticxt&staticfile=show.php?title=2079&Itemid=27), edited by his wife (1886). This work contains a bibliography of Jevons' writings.
- Powerhouse Museum, Sydney. "The curious economist: William Stanley Jevons in Sydney" (<http://www.powerhousemuseum.com/exhibitions/jevons.asp>). Retrieved 2007-05-09.
- "William Stanley Jevons (1835–1883)" (<http://www.econlib.org/library/Enc/bios/Jevons.html>). *The Concise Encyclopedia of Economics*. Library of Economics and Liberty (2nd ed.) (Liberty Fund). 2008.

Works available online

- The Coal Question (<http://www.econlib.org/library/YPDBooks/Jevons/jvnCQ.html>) (also available here (<http://oll.libertyfund.org/Home3/Book.php?recordID=0546>))
- The Theory of Political Economy (<http://www.econlib.org/library/YPDBooks/Jevons/jvnPE.html>)
- Money and the Mechanism of Exchange (<http://www.econlib.org/library/YPDBooks/Jevons/jvnMME.html>)
- Richard Cantillon and the Nationality of Political Economy (<http://www.econlib.org/library/NPDBooks/Cantillon/cntNT8.html>)
- *Elementary Lessons in Logic* (http://mises.org/books/lessons_in_logic_jevons.pdf)
- *Money and the Mechanism of Exchange* (http://mises.org/books/money_mechanism_jevons.pdf)
- *The Theory of Political Economy* (http://mises.org/books/political_economy_jevons.pdf)
- William Stanley Jevons (<http://socserv2.socsci.mcmaster.ca/~econ/ugcm/3li3/jevons/index.html>)
- Works by W. Stanley Jevons ([http://archive.org/search.php?query=creator:"Jevons,+William+Stanley,+1835-1882"](http://archive.org/search.php?query=creator:)) at Internet Archive
- Works by W. Stanley Jevons ([http://catalog.hathitrust.org/Search/Home?lookfor="Jevons, William Stanley, 1835-1882."&type=author&inst=all](http://catalog.hathitrust.org/Search/Home?lookfor=)) at Hathi Trust
- Works by W. Stanley Jevons ([http://www.europeana.eu/portal/search.html?query=who:\(Jevons+William+Stanley+\)&rows=12](http://www.europeana.eu/portal/search.html?query=who:(Jevons+William+Stanley+)&rows=12)) at Europeana
- Brief Account of a General Mathematical Theory of Political Economy (<http://www.marxists.org/reference/subject/economics/jevons/mathem.htm>)

From Google.com books via Contents tab or [page] box at top or scroll:

- The Principles of Political Economy (http://books.google.com/books?id=aYcBAAAAQAAJ&printsec=frontcover&source=gbv2_summary_r&cad=0#v=onepage&q&f=false), 1871,
- The Theory of Political Economy (http://books.google.com/books?id=aYcBAAAAQAAJ&printsec=frontcover&source=gbv2_summary_r&cad=0#v=onepage&q&f=false), 1879, 2nd ed.
- The Theory of Political Economy ([http://books.google.com/books?id=uigbAAAAYAAJ&dq="The+Theory+of+Political+Economy"+jevons&printsec=frontcover&source=bn&hl=en&ei=TvUETLiBDsGclgf4krTXBg&sa=X&oi=book_result&ct=result&resnum=4&ved=0CCQQ6AEwAw#v=onepage&q&f=false](http://books.google.com/books?id=uigbAAAAYAAJ&dq=)), 1888, 3rd ed. (1879 ed. + 3rd Preface by Harriet A. Jevons & adds to bibliographic 1st Appendix).

Propositional Calculus

Clause (logic)

In logic, a **clause** is a finite disjunction of literals. Clauses are usually written as follows, where the symbols l_i are literals:

$$l_1 \vee \dots \vee l_n$$

In some cases, clauses are written (or defined) as sets of literals, so that clause above would be written as $\{l_1, \dots, l_n\}$. That this set is to be interpreted as the disjunction of its elements is implied by the context. A clause can be empty; in this case, it is an empty set of literals. The empty clause is denoted by various symbols such as \emptyset , \perp , or \square . The truth evaluation of an empty clause is always *false*.

In first-order logic, a clause is interpreted as the universal closure of the disjunction of literals.^[citation needed] Formally, a first-order *atom* is a formula of the kind of $P(t_1, \dots, t_n)$, where P is a predicate of arity n and each t_i is an arbitrary term, possibly containing variables. A first-order *literal* is either an atom $P(t_1, \dots, t_n)$ or a negated atom $\neg P(t_1, \dots, t_n)$. If L_1, \dots, L_m are literals, and x_1, \dots, x_k are their (free) variables, then L_1, \dots, L_m is a clause, implicitly read as the closed first-order formula $\forall x_1, \dots, x_k. L_1, \dots, L_m$. The usual definition of satisfiability assumes free variables to be existentially quantified, so the omission of a quantifier is to be taken as a convention and not as a consequence of how the semantics deal with free variables. In logic programming, clauses are usually written as the implication of a head from a body. In the simplest case, the body is a conjunction of literals while the head is a single literal. More generally, the head may be a disjunction of literals. If b_1, \dots, b_m are the literals in the body of a clause and h_1, \dots, h_n are those of its head, the clause is usually written as follows:

$$h_1, \dots, h_n \leftarrow b_1, \dots, b_m$$

- If $m=0$ and $n=1$, the clause is called a (Prolog) fact.
- If $m>0$ and $n=1$, the clause is called a (Prolog) rule.
- If $m>0$ and $n=0$, the clause is called a (Prolog) query.
- If $n>1$, the clause is no longer Horn.

References

External links

- Clause logic related terminology ([http://www.articleworld.org/index.php/Clause_\(logic\)](http://www.articleworld.org/index.php/Clause_(logic)))
- Clause simultaneously translated in several languages and meanings (<http://www.free-dictionary-translation.com/Clause.html>)

Contradiction

In classical logic, a **contradiction** consists of a logical incompatibility between two or more propositions. It occurs when the propositions, taken together, yield two conclusions which form the logical, usually opposite inversions of each other. Illustrating a general tendency in applied logic, Aristotle's law of noncontradiction states that "One cannot say of something that it is and that it is not in the same respect and at the same time."

By extension, outside of classical logic, one can speak of contradictions between actions when one presumes that their motives contradict each other.

History

By creation of a paradox, Plato's *Euthydemus* dialogue demonstrates the need for the notion of *contradiction*. In the ensuing dialogue Dionysodorus denies the existence of "contradiction", all the while that Socrates is contradicting him:

". . . I in my astonishment said: What do you mean Dionysodorus? I have often heard, and have been amazed to hear, this thesis of yours, which is maintained and employed by the disciples of Protagoras and others before them, and which to me appears to be quite wonderful, and suicidal as well as destructive, and I think that I am most likely to hear the truth about it from you. The dictum is that there is no such thing as a falsehood; a man must either say what is true or say nothing. Is not that your position?"

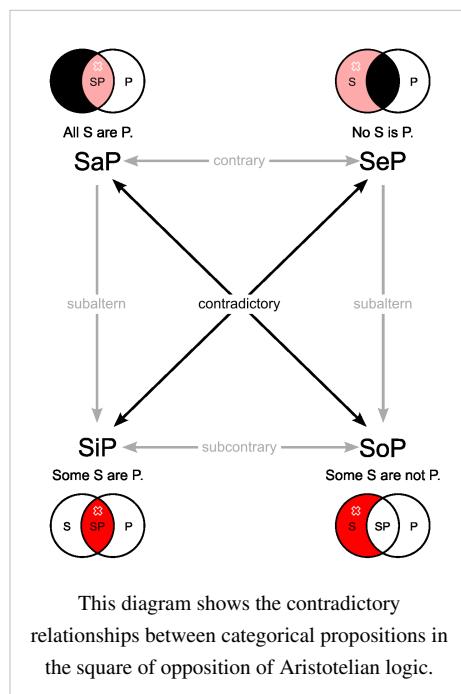
Indeed, Dionysodorus agrees that "there is no such thing as false opinion . . . there is no such thing as ignorance" and demands of Socrates to "Refute me." Socrates responds "But how can I refute you, if, as you say, to tell a falsehood is impossible?".^[1]

Contradiction in formal logic

Note: The symbol \perp (falsum) represents an arbitrary contradiction, with the dual tee symbol \top used to denote an arbitrary tautology. Contradiction is sometimes symbolized by " Opq ", and tautology by " Vpq ". The turnstile symbol, \vdash is often read as "yields" or "proves".

In classical logic, particularly in propositional and first-order logic, a proposition φ is a contradiction if and only if $\varphi \vdash \perp$. Since for contradictory φ it is true that $\vdash \varphi \rightarrow \psi$ for all ψ (because $\perp \rightarrow \psi$), one may prove any proposition from a set of axioms which contains contradictions. This is called the "principle of explosion" or "ex falso quodlibet" ("from falsity, whatever you like").

In a complete logic, a formula is contradictory if and only if it is unsatisfiable.



Proof by contradiction

For a proposition φ it is true that $\vdash \varphi$, i. e. that φ is a tautology, i. e. that it is always true, if and only if $\neg\varphi \vdash \perp$, i. e. if the negation of φ is a contradiction. Therefore, a proof that $\neg\varphi \vdash \perp$ also proves that φ is true. The use of this fact constitutes the technique of the proof by contradiction, which mathematicians use extensively. This applies only in a logic using the excluded middle $A \vee \neg A$ as an axiom.

Symbolic representation

In mathematics, the symbol used to represent a contradiction within a proof varies. [2] Some symbols that may be used to represent a contradiction include \square , Opq , $\Rightarrow \Leftarrow$, \perp , \Box , and \ast . It is not uncommon to see Q.E.D. or some variant immediately after a contradiction symbol; this occurs in a proof by contradiction, to indicate that the original assumption was false and that the theorem must therefore be true.

The notion of contradiction in an axiomatic system and a proof of its consistency

A Consistency proof requires (i) an axiomatic system (ii) a demonstration that it is not the case that both the formula p and its negation $\neg p$ can be derived in the system. But by whatever method one goes about it, all consistency proofs would *seem* to necessitate the primitive notion of *contradiction*; moreover, it *seems* as if this notion would simultaneously have to be "outside" the formal system in the definition of tautology.

When Emil Post in his 1921 *Introduction to a general theory of elementary propositions* extended his proof of the consistency of the propositional calculus (i.e. the logic) beyond that of *Principia Mathematica* (PM) he observed that with respect to a *generalized* set of postulates (i.e. axioms) he would no longer be able to automatically invoke the notion of "contradiction" – such a notion might not be contained in the postulates:

"The prime requisite of a set of postulates is that it be consistent. Since the ordinary notion of consistency involves that of contradiction, which again involves negation, and since this function does not appear in general as a primitive in [the *generalized* set of postulates] a new definition must be given".^[3]

Post's solution to the problem is described in the demonstration *An Example of a Successful Absolute Proof of Consistency* offered by Ernest Nagel and James R. Newman in their 1958 *Gödel's Proof*. They too observe a problem with respect to the notion of "contradiction" with its usual "truth values" of "truth" and "falsity". They observe that:

"The property of being a tautology has been defined in notions of truth and falsity. Yet these notions obviously involve a reference to something *outside* the formula calculus. Therefore, the procedure mentioned in the text in effect offers an *interpretation* of the calculus, by supplying a model for the system. This being so, the authors have not done what they promised, namely, '**to define a property of formulas in terms of purely structural features of the formulas themselves**'. [Indeed] . . . proofs of consistency which are based on models, and which argue from the truth of axioms to their consistency, merely shift the problem."^[4]

Given some "primitive formulas" such as PM's primitives $S_1 \vee S_2$ [inclusive OR], $\neg S$ (negation) one is forced to define the axioms in terms of these primitive notions. In a thorough manner Post demonstrates in PM, and defines (as do Nagel and Newman, see below), that the property of *tautologous* – as yet to be defined – is "inherited": if one begins with a set of tautologous axioms (postulates) and a deduction system that contains substitution and modus ponens then a *consistent* system will yield only tautologous formulas.

So what will be the definition of *tautologous*?

Nagel and Newman create two mutually exclusive and exhaustive classes K_1 and K_2 into which fall (the outcome of) the axioms when their variables e.g. S_1 and S_2 are assigned from these classes. This also applies to the primitive formulas. For example: "A formula having the form $S_1 \vee S_2$ is placed into class K_2 if both S_1 and S_2 are in K_2 ; otherwise it is placed in K_1 ", and "A formula having the form $\neg S$ is placed in K_2 , if S is in K_1 ; otherwise it is placed in K_1 ".^[5]

Nagel and Newman can now define the notion of *tautologous*: "a formula is a tautology if, and only if, it falls in the class K_1 no matter in which of the two classes its elements are placed".^[6] Now the property of "being tautologous" is described without reference to a model or an interpretation.

For example, given a formula such as $\sim S_1 \vee S_2$ and an assignment of K_1 to S_1 and K_2 to S_2 one can evaluate the formula and place its outcome in one or the other of the classes. The assignment of K_1 to $\sim S_1$ places $\sim S_1$ in K_2 , and now we can see that our assignment causes the formula to fall into class K_2 . Thus by definition our formula is not a tautology.

Post observed that, if the system were inconsistent, a deduction in it (that is, the last formula in a sequence of formulas derived from the tautologies) could ultimately yield S itself. As as an assignment to variable S can come from either class K_1 or K_2 , the deduction violates the inheritance characteristic of tautology, i.e. the derivation must yield an (evaluation of a formula) that will fall into class K_1 . From this, Post was able to derive the following definition of inconsistency *without the use of the notion of contradiction*:

Definition. A system will be said to be inconsistent if it yields the assertion of the unmodified variable p [S in the Newman and Nagel examples].

In other words, the notion of "contradiction" can be dispensed when constructing a proof of consistency; what replaces it is the notion of "mutually exclusive and exhaustive" classes. More interestingly,^[citation needed] an axiomatic system need not include the notion of "contradiction".

Contradictions and philosophy

Adherents of the epistemological theory of coherentism typically claim that as a necessary condition of the justification of a belief, that belief must form a part of a logically non-contradictory (consistent) system of beliefs. Some dialetheists, including Graham Priest, have argued that coherence may not require consistency.^[7]

Pragmatic contradictions

A pragmatic contradiction occurs when the very statement of the argument contradicts the claims it purports. An inconsistency arises, in this case, because the act of utterance, rather than the content of what is said, undermines its conclusion. For examples, arguably, Nietzsche's statement that one should not obey others, or Moore's paradox. Within the analytic tradition, these are seen as self-refuting statements and performative contradictions. Other traditions may read them more like zen koans, in which the author purposes makes a contradiction using the traditional meaning, but then implies a new meaning of the word which does not contradict the statement.

Dialectical materialism

In dialectical materialism, contradiction, as derived by Karl Marx from Hegelianism, usually refers to an opposition inherently existing within one realm, one unified force or object. This contradiction, as opposed to metaphysical thinking, is not an objectively impossible thing, because these contradicting forces exist in objective reality, not cancelling each other out, but actually defining each other's existence. According to Marxist theory, such a contradiction can be found, for example, in the fact that:

- (a) enormous wealth and productive powers coexist alongside:
- (b) extreme poverty and misery;
- (c) the existence of (a) being contrary to the existence of (b).

Hegelian and Marxist theory stipulates that the dialectic nature of history will lead to the sublation, or synthesis, of its contradictions. Marx therefore postulated that history would logically make capitalism evolve into a socialist society where the means of production would equally serve the exploited and suffering class of society, thus resolving the prior contradiction between (a) and (b).^[citation needed]

Mao Zedong's philosophical essay furthered Marx and Lenin's thesis and suggested that all existence is the result of contradiction.

Contradiction outside formal logic

Colloquial usage can label actions and/or statements as contradicting each other when due (or perceived as due) to presuppositions which are contradictory in the logical sense.

Proof by contradiction is used in mathematics to construct proofs.

Footnotes

[1] Dialog *Euthydemus* from *The Dialogs of Plato* translated by Benjamin Jowett appearing in: BK 7 *Plato*: Robert Maynard Hutchins, editor in chief, 1952, *Great Books of the Western World*, Encyclopædia Britannica, Inc., Chicago.

[2] <http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf>

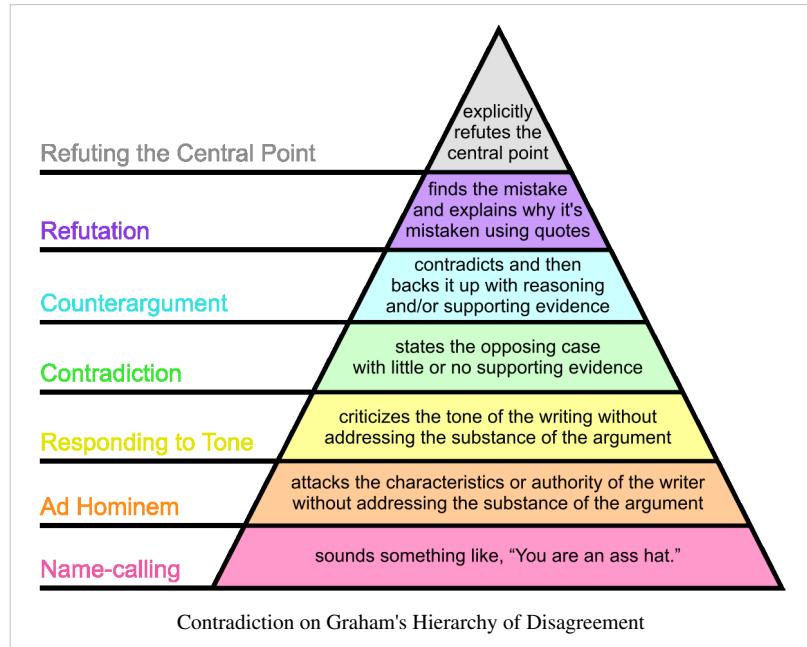
[3] Post 1921 *Introduction to a general theory of elementary propositions* in van Heijenoort 1967:272.

[4] boldface italics added, Nagel and Newman:109-110.

[5] Nagel and Newman:110-111

[6] Nagel and Newman:111

[7] In Contradiction: A Study of the Transconsistent (http://books.google.com/books?hl=en&id=bId-cRNUWdAC&dq=contradiction&printsec=frontcover&source=web&ots=7GWw7Qk_h3&sig=Od0aTst-tCcnbHDbOzpwBHssjEs&sa=X&oi=book_result&resnum=10&ct=result#PPP1,M1) By Graham Priest



References

- Józef Maria Bocheński 1960 *Précis of Mathematical Logic*, translated from the French and German editions by Otto Bird, D. Reidel, Dordrecht, South Holland.
- Jean van Heijenoort 1967 *From Frege to Gödel: A Source Book in Mathematical Logic 1879-1931*, Harvard University Press, Cambridge, MA, ISBN 0-674-32449-8 (pbk.)
- Ernest Nagel and James R. Newman 1958 *Gödel's Proof*, New York University Press, Card Catalog Number: 58-5610.

External links

- Hazewinkel, Michiel, ed. (2001), "Contradiction (inconsistency)" (<http://www.encyclopediaofmath.org/index.php?title=p/c025900>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Hazewinkel, Michiel, ed. (2001), "Contradiction, law of" (<http://www.encyclopediaofmath.org/index.php?title=p/c025910>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Contradiction (<http://plato.stanford.edu/entries/contradiction>) entry by Laurence R. Horn in the *Stanford Encyclopedia of Philosophy*

Deductive closure

Deductive closure is a property of a set of objects (usually the objects in question are statements). A set of objects, \circ , is said to exhibit *closure* or to be *closed* under a given operation, R , provided that for every object, x , if x is a member of \circ and x is R -related to any object, y , then y is a member of \circ .^[1] In the context of statements, a deductive closure is the set of all the statements that can be deduced from a given set of statements.

In propositional logic, the set of all true propositions exhibits **deductive closure**: if set \circ is the set of true propositions, and operation R is logical consequence (" \vdash "), then provided that proposition p is a member of \circ and p is R -related to q (i.e., $p \vdash q$), q is also a member of \circ .

Epistemic closure

In epistemology, many philosophers have and continue to debate whether particular subsets of propositions—especially ones ascribing knowledge or justification of a belief to a subject—are closed under deduction.

References

[1] Peter D. Klein, *Closure*, *The Cambridge Dictionary of Philosophy* (second edition)

Formation rule

In mathematical logic, **formation rules** are rules for describing which strings of symbols formed from the alphabet of a formal language are syntactically valid within the language. These rules only address the location and manipulation of the strings of the language. It does not describe anything else about a language, such as its semantics (i.e. what the strings mean). (See also formal grammar).

Formal language

A *formal language* is an organized set of symbols the essential feature being that it can be precisely defined in terms of just the shapes and locations of those symbols. Such a language can be defined, then, without any reference to any meanings of any of its expressions; it can exist before any interpretation is assigned to it—that is, before it has any meaning. A formal grammar determines which symbols and sets of symbols are formulas in a formal language.

Formal systems

A *formal system* (also called a *logical calculus*, or a *logical system*) consists of a formal language together with a deductive apparatus (also called a *deductive system*). The deductive apparatus may consist of a set of transformation rules (also called *inference rules*) or a set of axioms, or have both. A formal system is used to derive one expression from one or more other expressions. Propositional and predicate calculi are examples of formal systems.

Propositional and predicate logic

The formation rules of a propositional calculus may, for instance, take a form such that;

- if we take Φ to be a propositional formula we can also take $\neg\Phi$ to be a formula;
- if we take Φ and Ψ to be a propositional formulas we can also take $(\Phi \& \Psi)$, $(\Phi \rightarrow \Psi)$, $(\Phi \vee \Psi)$ and $(\Phi \leftrightarrow \Psi)$ to also be formulas.

A predicate calculus will usually include all the same rules as a propositional calculus, with the addition of quantifiers such that if we take Φ to be a formula of propositional logic and α as a variable then we can take $(\forall \alpha)\Phi$ and $(\exists \alpha)\Phi$ each to be formulas of our predicate calculus.

Frege system

In proof complexity, a **Frege system** is a propositional proof system whose proofs are sequences of formulas derived using a finite set of sound and implicationally complete inference rules. Frege systems (more often known as Hilbert systems in general proof theory) are named after Gottlob Frege.

Formal definition

Let K be a finite functionally complete set of Boolean connectives, and consider propositional formulas built from variables p_0, p_1, p_2, \dots using K -connectives. A Frege rule is an inference rule of the form

$$r = \frac{B_1, \dots, B_n}{B},$$

where B_1, \dots, B_n, B are formulas. If R is a finite set of Frege rules, then $F = (K, R)$ defines a derivation system in the following way. If X is a set of formulas, and A is a formula, then an F -derivation of A from axioms X is a sequence of formulas A_1, \dots, A_m such that $A_m = A$, and every A_k is a member of X , or it is derived from some of the formulas A_i , $i < k$, by a substitution instance of a rule from R . An F -proof of a formula A is an F -derivation of A from the empty set of axioms ($X = \emptyset$). F is called a Frege system if

- F is sound: every F -provable formula is a tautology.
- F is implicationally complete: for every formula A and a set of formulas X , if X entails A , then there is an F -derivation of A from X .

The length (number of lines) in a proof A_1, \dots, A_m is m . The size of the proof is the total number of symbols.

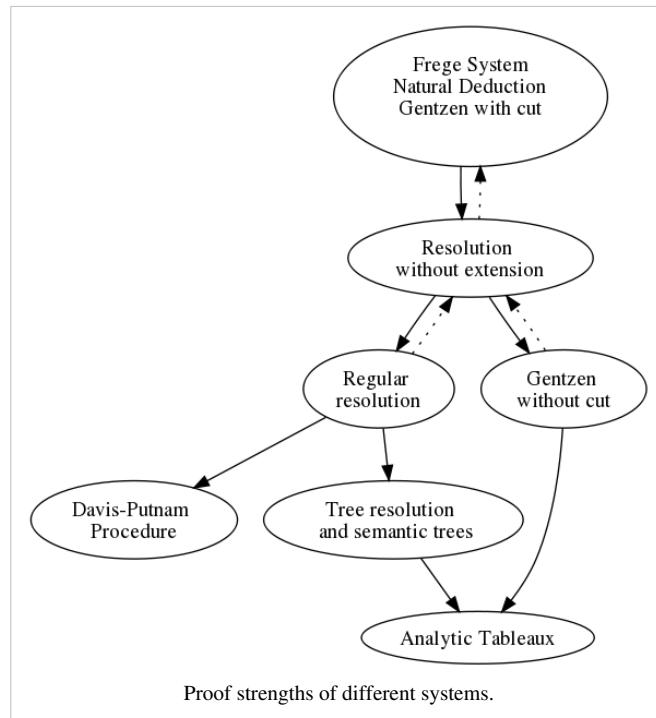
A derivation system F as above is refutationally complete, if for every inconsistent set of formulas X , there is an F -derivation of a fixed contradiction from X .

Examples

- Frege's propositional calculus is a Frege system.
- There are many examples of sound Frege rules on the Propositional calculus page.
- Resolution is not a Frege system because it only operates on clauses, not on formulas built in an arbitrary way by a functionally complete set of connectives. Moreover, it is not implicationally complete, i.e. we cannot conclude $A \vee B$ from A . However, adding the *weakening* rule: $\frac{A}{A \vee B}$ makes it implicationally complete. Resolution is also refutationally complete.

Properties

- Reckhow's theorem (1979) states that all Frege systems are p-equivalent.
- Natural deduction and sequent calculus (Gentzen system with cut) are also p-equivalent to Frege systems.
- There are polynomial-size Frege proofs of the pigeonhole principle (Buss 1987).
- Frege systems are considered to be fairly strong systems. Unlike, say, resolution, there are no known superlinear lower bounds on the number of lines in Frege proofs, and the best known lower bounds on the size of the proofs are quadratic.
- The minimal number of rounds in the *prover-adversary* game needed to prove a tautology ϕ is proportional to the logarithm of the minimal number of steps in a Frege proof of ϕ .



References

- Krajíček, Jan (1995). "Bounded Arithmetic, Propositional Logic, and Complexity Theory", Cambridge University Press.
- Cook, Stephen; Reckhow, Robert A. (1979). "The Relative Efficiency of Propositional Proof Systems". *Journal of Symbolic Logic* 44 (1). pp. 36–50. JSTOR 2273702 [1].
- Buss, S. R. (1987). "Polynomial size proofs of the propositional pigeonhole principle", Journal of Symbolic Logic 52, pp. 916–927.
- Pudlák, P., Buss, S. R. (1995). "How to lie without being (easily) convicted and the lengths of proofs in propositional calculus", in: Computer Science Logic'94 (Pacholski and Tiuryn eds.), Springer LNCS 933, 1995, pp. 151–162.

Further reading

- MacKay, D. J. (2008). "Information Theory, Inference, and Learning Algorithms"

References

[1] <http://www.jstor.org/stable/2273702>

Frege's propositional calculus

In mathematical logic **Frege's propositional calculus** was the first axiomatization of propositional calculus. It was invented by Gottlob Frege, who also invented predicate calculus, in 1879 as part of his second-order predicate calculus (although Charles Peirce was the first to use the term "second-order" and developed his own version of the predicate calculus independently of Frege).

It makes use of just two logical operators: implication and negation, and it is constituted by six axioms and one inference rule: modus ponens.

Axioms

THEN-1: $A \rightarrow (B \rightarrow A)$

THEN-2: $(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$

THEN-3: $(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$

FRG-1: $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$

FRG-2: $\neg\neg A \rightarrow A$

FRG-3: $A \rightarrow \neg\neg A$

Inference Rule

MP: $P, P \rightarrow Q \vdash Q$

Frege's propositional calculus is equivalent to any other classical propositional calculus, such as the "standard PC" with 11 axioms. Frege's PC and standard PC share two common axioms: THEN-1 and THEN-2. Notice that axioms THEN-1 through THEN-3 only make use of (and define) the implication operator, whereas axioms FRG-1 through FRG-3 define the negation operator.

The following theorems will aim to find the remaining nine axioms of standard PC within the "theorem-space" of Frege's PC, showing that the theory of standard PC is contained within the theory of Frege's PC.

(A theory, also called here, for figurative purposes, a "theorem-space", is a set of theorems which are a subset of a universal set of well-formed formulas. The theorems are linked to each other in a directed manner by inference rules, forming a sort of dendritic network. At the roots of the theorem-space are found the axioms, which "generate" the theorem-space much like a generating set generates a group.)

Rule THEN-1*: $A \vdash B \rightarrow A$

#	wff	reason
1.	A	premise
2.	$A \rightarrow (B \rightarrow A)$	THEN-1
3.	$B \rightarrow A$	MP 1,2.

Rule THEN-2*: $A \rightarrow (B \rightarrow C) \vdash (A \rightarrow B) \rightarrow (A \rightarrow C)$

#	wff	reason
1.	$A \rightarrow (B \rightarrow C)$	premise
2.	$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$	THEN-2
3.	$(A \rightarrow B) \rightarrow (A \rightarrow C)$	MP 1,2.

Rule THEN-3*: $A \rightarrow (B \rightarrow C) \vdash B \rightarrow (A \rightarrow C)$

#	wff	reason
1.	$A \rightarrow (B \rightarrow C)$	premise
2.	$(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$	THEN-3
3.	$B \rightarrow (A \rightarrow C)$	MP 1,2.

Rule FRG-1*: $A \rightarrow B \vdash \neg B \rightarrow \neg A$

#	wff	reason
1.	$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$	FRG-1
2.	$A \rightarrow B$	premise
3.	$\neg B \rightarrow \neg A$	MP 2,1.

Rule TH1*: $A \rightarrow B, B \rightarrow C \vdash A \rightarrow C$

#	wff	reason
1.	$B \rightarrow C$	premise
2.	$(B \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$	THEN-1
3.	$A \rightarrow (B \rightarrow C)$	MP 1,2
4.	$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$	THEN-2
5.	$(A \rightarrow B) \rightarrow (A \rightarrow C)$	MP 3,4
6.	$A \rightarrow B$	premise
7.	$A \rightarrow C$	MP 6,5.

Theorem TH1: $(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$

#	wff	reason
1.	$(B \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$	THEN-1
2.	$(A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$	THEN-2
3.	$(B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$	TH1* 1,2
4.	$((B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))) \rightarrow ((A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)))$	THEN-3
5.	$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$	MP 3,4.

Theorem TH2: $A \rightarrow (\neg A \rightarrow \neg B)$

#	wff	reason
1.	$A \rightarrow (B \rightarrow A)$	THEN-1
2.	$(B \rightarrow A) \rightarrow (\neg A \rightarrow \neg B)$	FRG-1
3.	$A \rightarrow (\neg A \rightarrow \neg B)$	TH1* 1,2.

Theorem TH3: $\neg A \rightarrow (A \rightarrow \neg B)$

#	wff	reason
1.	$A \rightarrow (\neg A \rightarrow \neg B)$	TH 2
2.	$(A \rightarrow (\neg A \rightarrow \neg B)) \rightarrow (\neg A \rightarrow (A \rightarrow \neg B))$	THEN-3
3.	$\neg A \rightarrow (A \rightarrow \neg B)$	MP 1,2.

Theorem TH4: $\neg(A \rightarrow \neg B) \rightarrow A$

#	wff	reason
1.	$\neg A \rightarrow (A \rightarrow \neg B)$	TH3
2.	$(\neg A \rightarrow (A \rightarrow \neg B)) \rightarrow (\neg(A \rightarrow \neg B) \rightarrow \neg \neg A)$	FRG-1
3.	$\neg(A \rightarrow \neg B) \rightarrow \neg \neg A$	MP 1,2
4.	$\neg \neg A \rightarrow A$	FRG-2
5.	$\neg(A \rightarrow \neg B) \rightarrow A$	TH1* 3,4.

Theorem TH5: $(A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A)$

#	wff	reason
1.	$(A \rightarrow \neg B) \rightarrow (\neg \neg B \rightarrow \neg A)$	FRG-1
2.	$((A \rightarrow \neg B) \rightarrow (\neg \neg B \rightarrow \neg A)) \rightarrow (\neg \neg B \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A))$	THEN-3
3.	$\neg \neg B \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$	MP 1,2
4.	$B \rightarrow \neg \neg B$	FRG-3, with $A := B$
5.	$B \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$	TH1* 4,3
6.	$(B \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)) \rightarrow ((A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A))$	FRG-1
7.	$(A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A)$	MP 5,6.

Theorem TH6: $\neg(A \rightarrow \neg B) \rightarrow B$

#	wff	reason
1.	$\neg(B \rightarrow \neg A) \rightarrow B$	TH4, with $A := B$, $B := A$
2.	$(B \rightarrow \neg A) \rightarrow (A \rightarrow \neg B)$	TH5, with $A := B$, $B := A$
3.	$((B \rightarrow \neg A) \rightarrow (A \rightarrow \neg B)) \rightarrow (\neg(A \rightarrow \neg B) \rightarrow \neg(B \rightarrow \neg A))$	FRG-1
4.	$\neg(A \rightarrow \neg B) \rightarrow \neg(B \rightarrow \neg A)$	MP 2,3
5.	$\neg(A \rightarrow \neg B) \rightarrow B$	TH1* 4,1.

Theorem TH7: $A \rightarrow A$

#	wff	reason
1.	$A \rightarrow \neg \neg A$	FRG-3
2.	$\neg \neg A \rightarrow A$	FRG-2
3.	$A \rightarrow A$	TH1* 1,2.

Theorem TH8: $A \rightarrow ((A \rightarrow B) \rightarrow B)$

#	wff	reason
1.	$(A \rightarrow B) \rightarrow (A \rightarrow B)$	TH7, with $A := A \rightarrow B$
2.	$((A \rightarrow B) \rightarrow (A \rightarrow B)) \rightarrow (A \rightarrow ((A \rightarrow B) \rightarrow B))$	THEN-3
3.	$A \rightarrow ((A \rightarrow B) \rightarrow B)$	MP 1,2.

Theorem TH9: $B \rightarrow ((A \rightarrow B) \rightarrow B)$

#	wff	reason
1.	$B \rightarrow ((A \rightarrow B) \rightarrow B)$	THEN-1, with $A := B$, $B := A \rightarrow B$.

Theorem TH10: $A \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))$

#	wff	reason
1.	$(A \rightarrow \neg B) \rightarrow (A \rightarrow \neg B)$	TH7
2.	$((A \rightarrow \neg B) \rightarrow (A \rightarrow \neg B)) \rightarrow (A \rightarrow ((A \rightarrow \neg B) \rightarrow \neg B))$	THEN-3
3.	$A \rightarrow ((A \rightarrow \neg B) \rightarrow \neg B)$	MP 1,2
4.	$((A \rightarrow \neg B) \rightarrow \neg B) \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))$	TH5
5.	$A \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))$	TH1* 3,4.

Note: $\neg(A \rightarrow \neg B) \rightarrow A$ (TH4), $\neg(A \rightarrow \neg B) \rightarrow B$ (TH6), and $A \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))$ (TH10), so $\neg(A \rightarrow \neg B)$ behaves just like $A \wedge B$ (compare with axioms AND-1, AND-2, and AND-3).

Theorem TH11: $(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$

#	wff	reason
1.	$A \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))$	TH10
2.	$(A \rightarrow (B \rightarrow \neg(A \rightarrow \neg B))) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow \neg(A \rightarrow \neg B)))$	THEN-2
3.	$(A \rightarrow B) \rightarrow (A \rightarrow \neg(A \rightarrow \neg B))$	MP 1,2
4.	$(A \rightarrow \neg(A \rightarrow \neg B)) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$	TH5
5.	$(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$	TH1* 3,4.

TH11 is axiom NOT-1 of standard PC, called "reductio ad absurdum".

Theorem TH12: $((A \rightarrow B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$

#	wff	reason
1.	$B \rightarrow (A \rightarrow B)$	THEN-1
2.	$(B \rightarrow (A \rightarrow B)) \rightarrow (((A \rightarrow B) \rightarrow C) \rightarrow (B \rightarrow C))$	TH1
3.	$((A \rightarrow B) \rightarrow C) \rightarrow (B \rightarrow C)$	MP 1,2
4.	$(B \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$	THEN-1
5.	$((A \rightarrow B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$	TH1* 3,4.

Theorem TH13: $(B \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow C)$

#	wff	reason
1.	$(B \rightarrow (B \rightarrow C)) \rightarrow ((B \rightarrow B) \rightarrow (B \rightarrow C))$	THEN-2
2.	$(B \rightarrow B) \rightarrow ((B \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow C))$	THEN-3* 1
3.	$B \rightarrow B$	TH7
4.	$(B \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow C)$	MP 3,2.

Rule TH14*: $A \rightarrow (B \rightarrow P), P \rightarrow Q \vdash A \rightarrow (B \rightarrow Q)$

#	wff	reason
1.	$P \rightarrow Q$	premise
2.	$(P \rightarrow Q) \rightarrow (B \rightarrow (P \rightarrow Q))$	THEN-1
3.	$B \rightarrow (P \rightarrow Q)$	MP 1,2
4.	$(B \rightarrow (P \rightarrow Q)) \rightarrow ((B \rightarrow P) \rightarrow (B \rightarrow Q))$	THEN-2
5.	$(B \rightarrow P) \rightarrow (B \rightarrow Q)$	MP 3,4
6.	$((B \rightarrow P) \rightarrow (B \rightarrow Q)) \rightarrow (A \rightarrow ((B \rightarrow P) \rightarrow (B \rightarrow Q)))$	THEN-1
7.	$A \rightarrow ((B \rightarrow P) \rightarrow (B \rightarrow Q))$	MP 5,6
8.	$(A \rightarrow (B \rightarrow P)) \rightarrow (A \rightarrow (B \rightarrow Q))$	THEN-2* 7
9.	$A \rightarrow (B \rightarrow P)$	premise
10.	$A \rightarrow (B \rightarrow Q)$	MP 9,8.

Theorem TH15: $((A \rightarrow B) \rightarrow (A \rightarrow C)) \rightarrow (A \rightarrow (B \rightarrow C))$

#	wff	reason
1.	$((A \rightarrow B) \rightarrow (A \rightarrow C)) \rightarrow (((A \rightarrow B) \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow C))$	THEN-2
2.	$((A \rightarrow B) \rightarrow C) \rightarrow (A \rightarrow (B \rightarrow C))$	TH12
3.	$((A \rightarrow B) \rightarrow (A \rightarrow C)) \rightarrow (((A \rightarrow B) \rightarrow A) \rightarrow (A \rightarrow (B \rightarrow C)))$	TH14* 1,2
4.	$((A \rightarrow B) \rightarrow A) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow (B \rightarrow C)))$	THEN-3* 3
5.	$A \rightarrow ((A \rightarrow B) \rightarrow A)$	THEN-1
6.	$A \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow (B \rightarrow C)))$	TH1* 5,4
7.	$((A \rightarrow B) \rightarrow (A \rightarrow C)) \rightarrow (A \rightarrow (A \rightarrow (B \rightarrow C)))$	THEN-3* 6
8.	$(A \rightarrow (A \rightarrow (B \rightarrow C))) \rightarrow (A \rightarrow (B \rightarrow C))$	TH13
9.	$((A \rightarrow B) \rightarrow (A \rightarrow C)) \rightarrow (A \rightarrow (B \rightarrow C))$	TH1* 7,8.

Theorem TH15 is the converse of axiom THEN-2.

Theorem TH16: $(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$

#	wff	reason
1.	$(\neg A \rightarrow \neg B) \rightarrow (\neg \neg B \rightarrow \neg \neg A)$	FRG-1
2.	$\neg \neg B \rightarrow ((\neg A \rightarrow \neg B) \rightarrow \neg \neg A)$	THEN-3* 1
3.	$B \rightarrow \neg \neg B$	FRG-3
4.	$B \rightarrow ((\neg A \rightarrow \neg B) \rightarrow \neg \neg A)$	TH1* 3,2
5.	$(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow \neg \neg A)$	THEN-3* 4
6.	$\neg \neg A \rightarrow A$	FRG-2
7.	$(\neg \neg A \rightarrow A) \rightarrow (B \rightarrow (\neg \neg A \rightarrow A))$	THEN-1
8.	$B \rightarrow (\neg \neg A \rightarrow A)$	MP 6,7
9.	$(B \rightarrow (\neg \neg A \rightarrow A)) \rightarrow ((B \rightarrow \neg \neg A) \rightarrow (B \rightarrow A))$	THEN-2
10.	$(B \rightarrow \neg \neg A) \rightarrow (B \rightarrow A)$	MP 8,9
11.	$(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$	TH1* 5,10.

Theorem TH17: $(\neg A \rightarrow B) \rightarrow (\neg B \rightarrow A)$

#	wff	reason
1.	$(\neg A \rightarrow \neg \neg B) \rightarrow (\neg B \rightarrow A)$	TH16, with $B := \neg B$
2.	$B \rightarrow \neg \neg B$	FRG-3
3.	$(B \rightarrow \neg \neg B) \rightarrow (\neg A \rightarrow (B \rightarrow \neg \neg B))$	THEN-1
4.	$\neg A \rightarrow (B \rightarrow \neg \neg B)$	MP 2,3
5.	$(\neg A \rightarrow (B \rightarrow \neg \neg B)) \rightarrow ((\neg A \rightarrow B) \rightarrow (\neg A \rightarrow \neg \neg B))$	THEN-2
6.	$(\neg A \rightarrow B) \rightarrow (\neg A \rightarrow \neg \neg B)$	MP 4,5
7.	$(\neg A \rightarrow B) \rightarrow (\neg B \rightarrow A)$	TH1* 6,1.

Compare TH17 with theorem TH5.

Theorem TH18: $((A \rightarrow B) \rightarrow B) \rightarrow (\neg A \rightarrow B)$

#	wff	reason
1.	$(A \rightarrow B) \rightarrow (\neg B \rightarrow (A \rightarrow B))$	THEN-1
2.	$(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$	TH16
3.	$(\neg B \rightarrow \neg A) \rightarrow (\neg B \rightarrow (A \rightarrow B))$	TH1* 2,1
4.	$((\neg B \rightarrow \neg A) \rightarrow (\neg B \rightarrow (A \rightarrow B))) \rightarrow (\neg B \rightarrow (\neg A \rightarrow (A \rightarrow B)))$	TH15
5.	$\neg B \rightarrow (\neg A \rightarrow (A \rightarrow B))$	MP 3,4
6.	$(\neg A \rightarrow (A \rightarrow B)) \rightarrow (\neg (A \rightarrow B) \rightarrow A)$	TH17
7.	$\neg B \rightarrow (\neg (A \rightarrow B) \rightarrow A)$	TH1* 5,6
8.	$(\neg B \rightarrow (\neg (A \rightarrow B) \rightarrow A)) \rightarrow ((\neg B \rightarrow \neg (A \rightarrow B)) \rightarrow (\neg B \rightarrow A))$	THEN-2
9.	$(\neg B \rightarrow \neg (A \rightarrow B)) \rightarrow (\neg B \rightarrow A)$	MP 7,8
10.	$((A \rightarrow B) \rightarrow B) \rightarrow (\neg B \rightarrow \neg (A \rightarrow B))$	FRG-1
11.	$((A \rightarrow B) \rightarrow B) \rightarrow (\neg B \rightarrow A)$	TH1* 10,9
12.	$(\neg B \rightarrow A) \rightarrow (\neg A \rightarrow B)$	TH17
13.	$((A \rightarrow B) \rightarrow B) \rightarrow (\neg A \rightarrow B)$	TH1* 11,12.

Theorem TH19: $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow C))$

#	wff	reason
1.	$\neg A \rightarrow (\neg B \rightarrow \neg(\neg A \rightarrow \neg B))$	TH10
2.	$B \rightarrow \neg B$	FRG-3
3.	$(B \rightarrow \neg B) \rightarrow (\neg A \rightarrow (B \rightarrow \neg B))$	THEN-1
4.	$\neg A \rightarrow (B \rightarrow \neg B)$	MP 2,3
5.	$(\neg A \rightarrow (B \rightarrow \neg B)) \rightarrow ((\neg A \rightarrow B) \rightarrow (\neg A \rightarrow \neg B))$	THEN-2
6.	$(\neg A \rightarrow B) \rightarrow (\neg A \rightarrow \neg B)$	MP 4,5
7.	$\neg(\neg A \rightarrow \neg B) \rightarrow \neg(\neg A \rightarrow B)$	FRG-1* 6
8.	$\neg A \rightarrow (\neg B \rightarrow \neg(\neg A \rightarrow B))$	TH14* 1,7
9.	$((A \rightarrow B) \rightarrow B) \rightarrow (\neg A \rightarrow B)$	TH18
10.	$\neg(\neg A \rightarrow B) \rightarrow ((A \rightarrow B) \rightarrow B)$	FRG-1* 9
11.	$\neg A \rightarrow (\neg B \rightarrow \neg((A \rightarrow B) \rightarrow B))$	TH14* 8,10
12.	$\neg C \rightarrow (\neg A \rightarrow (\neg B \rightarrow \neg((A \rightarrow B) \rightarrow B)))$	THEN-1* 11
13.	$(\neg C \rightarrow \neg A) \rightarrow (\neg C \rightarrow (\neg B \rightarrow \neg((A \rightarrow B) \rightarrow B)))$	THEN-2* 12
14.	$(\neg C \rightarrow (\neg B \rightarrow \neg((A \rightarrow B) \rightarrow B))) \rightarrow ((\neg C \rightarrow B) \rightarrow (\neg C \rightarrow \neg((A \rightarrow B) \rightarrow B)))$	THEN-2
15.	$(\neg C \rightarrow \neg A) \rightarrow ((\neg C \rightarrow \neg B) \rightarrow (\neg C \rightarrow \neg((A \rightarrow B) \rightarrow B)))$	TH1* 13,14
16.	$(A \rightarrow C) \rightarrow (\neg C \rightarrow \neg A)$	FRG-1
17.	$(A \rightarrow C) \rightarrow ((\neg C \rightarrow \neg B) \rightarrow (\neg C \rightarrow \neg((A \rightarrow B) \rightarrow B)))$	TH1* 16,15
18.	$(\neg C \rightarrow \neg((A \rightarrow B) \rightarrow B)) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow C)$	TH16
19.	$(A \rightarrow C) \rightarrow ((\neg C \rightarrow \neg B) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow C))$	TH14* 17,18
20.	$(B \rightarrow C) \rightarrow (\neg C \rightarrow \neg B)$	FRG-1
21.	$((B \rightarrow C) \rightarrow (\neg C \rightarrow \neg B)) \rightarrow ((\neg C \rightarrow B) \rightarrow ((A \rightarrow B) \rightarrow C))$	TH1
22.	$((\neg C \rightarrow \neg B) \rightarrow ((A \rightarrow B) \rightarrow C)) \rightarrow ((B \rightarrow C) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow C))$	MP 20,21
23.	$(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow C))$	TH1* 19,22.

Note: $A \rightarrow ((A \rightarrow B) \rightarrow B)$ (TH8), $B \rightarrow ((A \rightarrow B) \rightarrow B)$ (TH9), and $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (((A \rightarrow B) \rightarrow B) \rightarrow C))$ (TH19), so $((A \rightarrow B) \rightarrow B)$ behaves just like $A \vee B$. (Compare with axioms OR-1, OR-2, and OR-3.)

Theorem TH20: $(A \rightarrow \neg A) \rightarrow \neg A$

#	wff	reason
1.	$(A \rightarrow A) \rightarrow ((A \rightarrow \neg A) \rightarrow \neg A)$	TH11
2.	$A \rightarrow A$	TH7
3.	$(A \rightarrow \neg A) \rightarrow \neg A$	MP 2,1.

TH20 corresponds to axiom NOT-3 of standard PC, called "tertium non datur".

Theorem TH21: $A \rightarrow (\neg A \rightarrow B)$

#	wff	reason
1.	$A \rightarrow (\neg A \rightarrow \neg\neg B)$	TH2, with $B := \neg B$
2.	$\neg\neg B \rightarrow B$	FRG-2
3.	$A \rightarrow (\neg A \rightarrow B)$	TH14* 1,2.

TH21 corresponds to axiom NOT-2 of standard PC, called "ex contradictione quodlibet".

All the axioms of standard PC have been derived from Frege's PC, after having let

$A \wedge B := \neg(A \rightarrow \neg B)$ and $A \vee B := (A \rightarrow B) \rightarrow B$. These expressions are not unique, e.g. $A \vee B$ could also have been defined as $(B \rightarrow A) \rightarrow A$, $\neg A \rightarrow B$, or $\neg B \rightarrow A$. Notice, though, that the definition $A \vee B := (A \rightarrow B) \rightarrow B$ contains no negations. On the other hand, $A \wedge B$ cannot be defined in terms of implication alone, without using negation.

In a sense, the expressions $A \wedge B$ and $A \vee B$ can be thought of as "black boxes". Inside, these black boxes contain formulas made up only of implication and negation. The black boxes can contain anything, as long as when plugged into the AND-1 through AND-3 and OR-1 through OR-3 axioms of standard PC the axioms remain true. These axioms provide complete syntactic definitions of the conjunction and disjunction operators.

The next set of theorems will aim to find the remaining four axioms of Frege's PC within the "theorem-space" of standard PC, showing that the theory of Frege's PC is contained within the theory of standard PC.

Theorem ST1: $A \rightarrow A$

#	wff	reason
1.	$(A \rightarrow ((A \rightarrow A) \rightarrow A)) \rightarrow ((A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A))$	THEN-2
2.	$A \rightarrow ((A \rightarrow A) \rightarrow A)$	THEN-1
3.	$(A \rightarrow (A \rightarrow A)) \rightarrow (A \rightarrow A)$	MP 2,1
4.	$A \rightarrow (A \rightarrow A)$	THEN-1
5.	$A \rightarrow A$	MP 4,3.

Theorem ST2: $A \rightarrow \neg\neg A$

#	wff	reason
1.	A	hypothesis
3.	$A \rightarrow (\neg A \rightarrow A)$	THEN-1
4.	$\neg A \rightarrow A$	MP 1,3
6.	$\neg A \rightarrow \neg A$	ST1
7.	$(\neg A \rightarrow A) \rightarrow ((\neg A \rightarrow \neg A) \rightarrow \neg\neg A)$	NOT-1
8.	$(\neg A \rightarrow \neg A) \rightarrow \neg\neg A$	MP 4,7
9.	$\neg\neg A$	MP 6,8
10.	$A \vdash \neg\neg A$	summary 1-9
11.	$A \rightarrow \neg\neg A$	DT 10.

ST2 is axiom FRG-3 of Frege's PC.

Theorem ST3: $B \vee C \rightarrow (\neg C \rightarrow B)$

#	wff	reason
1.	$C \rightarrow (\neg C \rightarrow B)$	NOT-2
2.	$B \rightarrow (\neg C \rightarrow B)$	THEN-1
3.	$(B \rightarrow (\neg C \rightarrow B)) \rightarrow ((C \rightarrow (\neg C \rightarrow B)) \rightarrow ((B \vee C) \rightarrow (\neg C \rightarrow B)))$	OR-3
4.	$(C \rightarrow (\neg C \rightarrow B)) \rightarrow ((B \vee C) \rightarrow (\neg C \rightarrow B))$	MP 2,3
5.	$B \vee C \rightarrow (\neg C \rightarrow B)$	MP 1,4.

Theorem ST4: $\neg\neg A \rightarrow A$

#	wff	reason
1.	$A \vee \neg A$	NOT-3
2.	$(A \vee \neg A) \rightarrow (\neg\neg A \rightarrow A)$	ST3
3.	$\neg\neg A \rightarrow A$	MP 1,2.

ST4 is axiom FRG-2 of Frege's PC.

Prove ST5: $(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$

#	wff	reason
1.	$A \rightarrow (B \rightarrow C)$	hypothesis
2.	B	hypothesis
3.	A	hypothesis
4.	$B \rightarrow C$	MP 3,1
5.	C	MP 2,4
6.	$A \rightarrow (B \rightarrow C), B, A \vdash C$	summary 1-5
7.	$A \rightarrow (B \rightarrow C), B \vdash A \rightarrow C$	DT 6
8.	$A \rightarrow (B \rightarrow C) \vdash B \rightarrow (A \rightarrow C)$	DT 7
9.	$(A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C))$	DT 8.

ST5 is axiom THEN-3 of Frege's PC.

Theorem ST6: $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$

#	wff	reason
1.	$A \rightarrow B$	hypothesis
2.	$\neg B$	hypothesis
3.	$\neg B \rightarrow (A \rightarrow \neg B)$	THEN-1
4.	$A \rightarrow \neg B$	MP 2,3
5.	$(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$	NOT-1
6.	$(A \rightarrow \neg B) \rightarrow \neg A$	MP 1,5
7.	$\neg A$	MP 4,6
8.	$A \rightarrow B, \neg B \vdash \neg A$	summary 1-7
9.	$A \rightarrow B \vdash \neg B \rightarrow \neg A$	DT 8
10.	$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$	DT 9.

ST6 is axiom FRG-1 of Frege's PC.

Each of Frege's axioms can be derived from the standard axioms, and each of the standard axioms can be derived from Frege's axioms. This means that the two sets of axioms are interdependent and there is no axiom in one set which is independent from the other set. Therefore the two sets of axioms generate the same theory: Frege's PC is equivalent to standard PC.

(For if the theories should be different, then one of them should contain theorems not contained by the other theory. These theorems can be derived from their own theory's axiom set: but as has been shown this entire axiom set can be derived from the other theory's axiom set, which means that the given theorems can actually be derived solely from the other theory's axiom set, so that the given theorems also belong to the other theory. Contradiction: thus the two axiom sets span the same theorem-space. By construction: Any theorem derived from the standard axioms can be derived from Frege's axioms, and vice versa, by first proving as theorems the axioms of the other theory as shown above and then using those theorems as lemmas to derive the desired theorem.)

References

- Buss, Samuel (1998). "An introduction to proof theory". *Handbook of proof theory*. Elsevier. pp. 1–78. ISBN 0-444-89840-9.

Implicational propositional calculus

In mathematical logic, the **implicational propositional calculus** is a version of classical propositional calculus which uses only one connective, called implication or conditional. In formulas, this binary operation is indicated by "implies", "if ..., then ...", " \rightarrow ", " \rightarrow ", etc..

Virtual completeness as an operator

Implication alone is not functionally complete as a logical operator because one cannot form all other two-valued truth functions from it. However, if one has a propositional formula which is known to be false and uses that as if it were a nullary connective for falsity, then one can define all other truth functions. So implication is virtually complete as an operator. If P, Q , and F are propositions and F is known to be false, then:

- $\neg P$ is equivalent to $P \rightarrow F$
- $P \wedge Q$ is equivalent to $(P \rightarrow (Q \rightarrow F)) \rightarrow F$
- $P \vee Q$ is equivalent to $(P \rightarrow Q) \rightarrow Q$
- $P \leftrightarrow Q$ is equivalent to $((P \rightarrow Q) \rightarrow ((Q \rightarrow P) \rightarrow F)) \rightarrow F$

More generally, since the above operators are known to be functionally complete, it follows that any truth function can be expressed in terms of " \rightarrow " and " F ", if we have a proposition F which is known to be false.

It is worth noting that F is not definable from \rightarrow and arbitrary sentence variables: any formula constructed from \rightarrow and propositional variables must receive the value true when all of its variables are evaluated to true. It follows as a corollary that $\{\rightarrow\}$ is not functionally complete. It cannot, for example, be used to define the two-place truth function that always returns *false*.

Axiom system

- Axiom schema 1 is $P \rightarrow (Q \rightarrow P)$.
- Axiom schema 2 is $(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$.
- Axiom schema 3 (Peirce's law) is $((P \rightarrow Q) \rightarrow P) \rightarrow P$.
- The one non-nullary rule of inference (modus ponens) is: from P and $P \rightarrow Q$ infer Q .

Where in each case, P , Q , and R may be replaced by any formulas which contain only " \rightarrow " as a connective. If Γ is a set of formulas and A a formula, then $\Gamma \vdash A$ means that A is derivable using the axioms and rules above and formulas from Γ as additional hypotheses.

Łukasiewicz (1948) found a simpler axiom system for the implicational calculus, which replaces the schemata 1–3 above with a single schema

- $((P \rightarrow Q) \rightarrow R) \rightarrow ((R \rightarrow P) \rightarrow (S \rightarrow P))$.

He also argued that there is no shorter axiom system.

Basis properties

Since all axioms and rules of the calculus are schemata, derivation is closed under substitution:

If $\Gamma \vdash A$, then $\sigma(\Gamma) \vdash \sigma(A)$,

where σ is any substitution (of formulas using only implication).

The implicational propositional calculus also satisfies the deduction theorem:

If $\Gamma, A \vdash B$, then $\Gamma \vdash A \rightarrow B$.

As explained in the deduction theorem article, this holds for any axiomatic extension of the system consisting of axioms 1 and 2 above and modus ponens.

Completeness

The implicational propositional calculus is semantically complete with respect to the usual two-valued semantics of classical propositional logic. That is, if Γ is a set of implicational formulas, and A is an implicational formula entailed by Γ , then $\Gamma \vdash A$.

Proof

A proof of the completeness theorem is outlined below. First, using the compactness theorem and the deduction theorem, we may reduce the completeness theorem to its special case with empty Γ , i.e., we only need to show that every tautology is derivable in the system.

The proof is similar to completeness of full propositional logic, but it also uses the following idea to overcome the functional incompleteness of implication. If A and F are formulas, then $A \rightarrow F$ is equivalent to $(\neg A^*) \vee F$, where A^* is the result of replacing in A all, some, or none of the occurrences of F by falsity. Similarly, $(A \rightarrow F) \rightarrow F$ is equivalent to $A^* \vee F$. So under some conditions, one can use them as substitutes for saying A^* is false or A^* is true respectively.

We first observe some basic facts about derivability:

$$\boxed{A \rightarrow B, B \rightarrow C \vdash A \rightarrow C} \quad (1)$$

Indeed, we can derive $A \rightarrow (B \rightarrow C)$ using Axiom 1, and then derive $A \rightarrow C$ by modus ponens (twice) from Ax. 2.

$$\boxed{A \rightarrow B \vdash (B \rightarrow C) \rightarrow (A \rightarrow C)} \quad (2)$$

This follows from (1) by the deduction theorem.

$$\boxed{A \rightarrow C, (A \rightarrow B) \rightarrow C \vdash C} \quad (3)$$

If we further assume $C \rightarrow B$, we can derive $A \rightarrow B$ using (1), then we derive C by modus ponens. This shows $A \rightarrow C, (A \rightarrow B) \rightarrow C, C \rightarrow B \vdash C$, and the deduction theorem gives $A \rightarrow C, (A \rightarrow B) \rightarrow C \vdash (C \rightarrow B) \rightarrow C$. We apply Ax. 3 to obtain (3).

Let F be an arbitrary fixed formula. For any formula A , we define $A^0 = (A \rightarrow F)$ and $A^1 = ((A \rightarrow F) \rightarrow F)$. Let us consider only formulas in propositional variables p_1, \dots, p_n . We claim that for every formula A in these variables and every truth assignment e ,

$$\boxed{p_1^{e(p_1)}, \dots, p_n^{e(p_n)} \vdash A^{e(A)}} \quad (4)$$

We prove (4) by induction on A . The base case $A = p_i$ is trivial. Let $A = (B \rightarrow C)$. We distinguish three cases:

1. $e(C) = 1$. Then also $e(A) = 1$. We have

$$(C \rightarrow F) \rightarrow F \vdash ((B \rightarrow C) \rightarrow F) \rightarrow F$$

by applying (2) twice to the axiom $C \rightarrow (B \rightarrow C)$. Since we have derived $(C \rightarrow F) \rightarrow F$ by the induction hypothesis, we can infer $((B \rightarrow C) \rightarrow F) \rightarrow F$.

2. $e(B) = 0$. Then again $e(A) = 1$. The deduction theorem applied to (3) gives

$$B \rightarrow F \vdash ((B \rightarrow C) \rightarrow F) \rightarrow F.$$

Since we have derived $B \rightarrow F$ by the induction hypothesis, we can infer $((B \rightarrow C) \rightarrow F) \rightarrow F$.

3. $e(B) = 1$ and $e(C) = 0$. Then $e(A) = 0$. We have

$$(B \rightarrow F) \rightarrow F, C \rightarrow F, B \rightarrow C \vdash B \rightarrow F \quad \text{by (1)}$$

$\vdash F \quad \text{by modus ponens,}$

thus $(B \rightarrow F) \rightarrow F, C \rightarrow F \vdash (B \rightarrow C) \rightarrow F$ by the deduction theorem. We have derived $(B \rightarrow F) \rightarrow F$ and $C \rightarrow F$ by the induction hypothesis, hence we can infer $(B \rightarrow C) \rightarrow F$. This completes the proof of (4).

Now let A be a tautology in variables p_1, \dots, p_n . We will prove by reverse induction on $k = n, \dots, 0$ that for every assignment e ,

$$\boxed{p_1^{e(p_1)}, \dots, p_k^{e(p_k)} \vdash A^1} \quad (5)$$

The base case $k = n$ is a special case of (4). Assume that (5) holds for $k + 1$, we will show it for k . By applying deduction theorem to the induction hypothesis, we obtain

$$\begin{aligned} p_1^{e(p_1)}, \dots, p_k^{e(p_k)} &\vdash (p_{k+1} \rightarrow F) \rightarrow A^1, \\ p_1^{e(p_1)}, \dots, p_k^{e(p_k)} &\vdash ((p_{k+1} \rightarrow F) \rightarrow F) \rightarrow A^1, \end{aligned}$$

by first setting $e(p_{k+1}) = 0$ and second setting $e(p_{k+1}) = 1$. From this we derive (5) using (3).

For $k = 0$ we obtain that the formula A^1 , i.e., $(A \rightarrow F) \rightarrow F$, is provable without assumptions. Recall that F was an arbitrary formula, thus we can choose $F = A$, which gives us provability of the formula $(A \rightarrow A) \rightarrow A$. Since $A \rightarrow A$ is provable by the deduction theorem, we can infer A .

This proof is constructive. That is, given a tautology, one could actually follow the instructions and create a proof of it from the axioms. However, the length of such a proof increases exponentially with the number of propositional variables in the tautology, hence it is not a practical method for any but the very shortest tautologies.

References

- Mendelson, Elliot (1997) *Introduction to Mathematical Logic*, 4th ed. ^[1] London: Chapman & Hall.
- Łukasiewicz, Jan (1948) *The shortest axiom of the implicational calculus of propositions* ^[2], Proc. Royal Irish Academy, vol. 52, sec. A, no. 3, pp. 25–33.

References

- [1] <http://worldcat.org/oclc/259359>
[2] <http://www.jstor.org/stable/20488489>

Intermediate logic

In mathematical logic, a **superintuitionistic logic** is a propositional logic extending intuitionistic logic. Classical logic is the strongest consistent superintuitionistic logic, thus consistent superintuitionistic logics are called **intermediate logics** (the logics are intermediate between intuitionistic logic and classical logic).^[citation needed]

Definition

A superintuitionistic logic is a set L of propositional formulas in a countable set of variables p_i satisfying the following properties:

1. all axioms of intuitionistic logic belong to L ;
2. if F and G are formulas such that F and $F \rightarrow G$ both belong to L , then G also belongs to L (closure under modus ponens);
3. if $F(p_1, p_2, \dots, p_n)$ is a formula of L , and G_1, G_2, \dots, G_n are any formulas, then $F(G_1, G_2, \dots, G_n)$ belongs to L (closure under substitution).

Such a logic is intermediate if furthermore

1. L is not the set of all formulas.

Properties and examples

There exists a continuum of different intermediate logics. Specific intermediate logics are often constructed by adding one or more axioms to intuitionistic logic, or by a semantical description. Examples of intermediate logics include:

- intuitionistic logic (**IPC**, **Int**, **IL**, **H**)
- classical logic (**CPC**, **Cl**, **CL**): **IPC** + $p \vee \neg p = \text{IPC} + \neg\neg p \rightarrow p = \text{IPC} + ((p \rightarrow q) \rightarrow p) \rightarrow p$
- the logic of the weak excluded middle (**KC**, Jankov's logic, De Morgan logic^[1]): **IPC** + $\neg\neg p \vee \neg p$
- Gödel–Dummett logic (**LC**, **G**): **IPC** + $(p \rightarrow q) \vee (q \rightarrow p)$
- Kreisel–Putnam logic (**KP**): **IPC** + $(\neg p \rightarrow (q \vee r)) \rightarrow ((\neg p \rightarrow q) \vee (\neg p \rightarrow r))$
- Medvedev's logic of finite problems (**LM**, **ML**): defined semantically as the logic of all frames of the form $\langle \mathcal{P}(X) \setminus \{X\}, \subseteq \rangle$ for finite sets X ("Boolean hypercubes without top"), as of 2010^[2] not known to be recursively axiomatizable
- realizability logics
- Scott's logic (**SL**): **IPC** + $((\neg\neg p \rightarrow p) \rightarrow (p \vee \neg p)) \rightarrow (\neg\neg p \vee \neg p)$
- Smetanich's logic (**SmL**): **IPC** + $(\neg q \rightarrow p) \rightarrow (((p \rightarrow q) \rightarrow p) \rightarrow p)$
- logics of bounded cardinality (**BC** _{n}): **IPC** + $\bigvee_{i=0}^n (\bigwedge_{j < i} p_j \rightarrow p_i)$
- logics of bounded width, also known as the logic of bounded anti-chains (**BW** _{n} , **BA** _{n}):

$$\text{IPC} + \bigvee_{i=0}^n (\bigwedge_{j \neq i} p_j \rightarrow p_i)$$

- logics of bounded depth (\mathbf{BD}_n): $\mathbf{IPC} + p_n \vee (p_n \rightarrow (p_{n-1} \vee (p_{n-1} \rightarrow \dots \rightarrow (p_2 \vee (p_2 \rightarrow (p_1 \vee \neg p_1)))\dots)))$
- logics of bounded top width (\mathbf{BTW}_n): $\mathbf{IPC} + \bigvee_{i=0}^n (\bigwedge_{j < i} p_j \rightarrow \neg\neg p_i)$
- logics of bounded branching ($\mathbf{T}_n, \mathbf{BB}_n$): $\mathbf{IPC} + \bigwedge_{i=0}^n ((p_i \rightarrow \bigvee_{j \neq i} p_j) \rightarrow \bigvee_{j \neq i} p_j) \rightarrow \bigvee_{i=0}^n p_i$
- Gödel n -valued logics (\mathbf{G}_n): $\mathbf{LC} + \mathbf{BC}_{n-1} = \mathbf{LC} + \mathbf{BD}_{n-1}$

Superintuitionistic or intermediate logics form a complete lattice with intuitionistic logic as the bottom and the inconsistent logic (in the case of superintuitionistic logics) or classical logic (in the case of intermediate logics) as the top. Classical logic is the only coatom in the lattice of superintuitionistic logics; the lattice of intermediate logics also has a unique coatom, namely **SmL**.

The tools for studying intermediate logics are similar to those used for intuitionistic logic, such as Kripke semantics. For example, Gödel–Dummett logic has a simple semantic characterization in terms of total orders.

Semantics

Given a Heyting algebra H , the set of propositional formulas that are valid in H is an intermediate logic. Conversely, given an intermediate logic it is possible to construct its Lindenbaum algebra which is a Heyting algebra.

An intuitionistic Kripke frame F is a partially ordered set, and a Kripke model M is a Kripke frame with valuation such that $\{x \mid M, x \Vdash p\}$ is an upper subset of F . The set of propositional formulas that are valid in F is an intermediate logic. Given an intermediate logic L it is possible to construct a Kripke model M such that the logic of M is L (this construction is called *canonical model*). A Kripke frame with this property may not exist, but a general frame always does.

Relation to modal logics

Let A be a propositional formula. The *Gödel–Tarski translation* of A is defined recursively as follows:

- $T(p_n) = \Box p_n$
- $T(\neg A) = \Box \neg T(A)$
- $T(A \wedge B) = T(A) \wedge T(B)$
- $T(A \vee B) = T(A) \vee T(B)$
- $T(A \rightarrow B) = \Box(T(A) \rightarrow T(B))$

If M is a modal logic extending **S4** then $\rho M = \{A \mid T(A) \in M\}$ is a superintuitionistic logic, and M is called a *modal companion* of ρM . In particular:

- **IPC** = $\rho \mathbf{S4}$
- **KC** = $\rho \mathbf{S4.2}$
- **LC** = $\rho \mathbf{S4.3}$
- **CPC** = $\rho \mathbf{S5}$

For every intermediate logic L there are many modal logics M such that $L = \rho M$.

References

- [1] Constructive Logic and the Medvedev Lattice, Sebastiaan A. Terwijn, Notre Dame J. Formal Logic, Volume 47, Number 1 (2006), 73-82.
- [2] http://en.wikipedia.org/w/index.php?title=Intermediate_logic&action=edit
- Toshio Umezawa. On logics intermediate between intuitionistic and classical predicate logic. Journal of Symbolic Logic, 24(2):141–153, June 1959.
- Alexander Chagrov, Michael Zakharyashev. Modal Logic. Oxford University Press, 1997.

List of logic systems

This article contains a list of sample Hilbert-style deductive systems for propositional logic.

Classical propositional calculus systems

Classical propositional calculus is the standard propositional logic. Its intended semantics is bivalent and its main property is that it is syntactically complete, otherwise said that no new axiom not already consequence of the existing axioms can be added without making the logic inconsistent. Many different equivalent complete axiom systems have been formulated. They differ in the choice of basic connectives used, which in all cases have to be functionally complete (i.e. able to express by composition all n-ary truth tables), and in the exact complete choice of axioms over the chosen basis of connectives.

Implication and negation

The formulations here use implication and negation $\{\rightarrow, \neg\}$ as functionally complete set of basic connectives. Every logic system requires at least one non-nullary rule of inference. Classical propositional calculus typically uses the rule of modus ponens:

$$A, A \rightarrow B \vdash B.$$

We assume this rule is included in all systems below unless stated otherwise.

Frege's axiom system:^[1]

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ & (A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A) \\ & \neg \neg A \rightarrow A \\ & A \rightarrow \neg \neg A \end{aligned}$$

Hilbert's axiom system:

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \\ & (B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ & A \rightarrow (\neg A \rightarrow B) \\ & (A \rightarrow B) \rightarrow ((\neg A \rightarrow B) \rightarrow B) \end{aligned}$$

Łukasiewicz's axiom systems:

- First:

$$\begin{aligned} & (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)) \\ & (\neg A \rightarrow A) \rightarrow A \end{aligned}$$

$$A \rightarrow (\neg A \rightarrow B)$$

- Second:

$$\begin{aligned} & ((A \rightarrow B) \rightarrow C) \rightarrow (\neg A \rightarrow C) \\ & ((A \rightarrow B) \rightarrow C) \rightarrow (B \rightarrow C) \\ & (\neg A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow C)) \end{aligned}$$

- Third:

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ & (\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A) \end{aligned}$$

- Fourth:^[citation needed]

$$\begin{aligned} & (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)) \\ & A \rightarrow (\neg A \rightarrow B) \\ & (\neg A \rightarrow B) \rightarrow ((B \rightarrow A) \rightarrow A) \end{aligned}$$

Łukasiewicz and Tarski's axiom system:^[2]

$$[(A \rightarrow (B \rightarrow A)) \rightarrow ([(\neg C \rightarrow (D \rightarrow \neg E)) \rightarrow [(C \rightarrow (D \rightarrow F)) \rightarrow ((E \rightarrow D) \rightarrow (E \rightarrow F))]] \rightarrow G)] \rightarrow (H \rightarrow G)$$

Meredith's axiom system:

$$(((A \rightarrow B) \rightarrow (\neg C \rightarrow \neg D)) \rightarrow C) \rightarrow E \rightarrow ((E \rightarrow A) \rightarrow (D \rightarrow A))$$

Mendelson's axiom system:^[3]

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ & (\neg A \rightarrow \neg B) \rightarrow ((\neg A \rightarrow B) \rightarrow A) \end{aligned}$$

Russell's axiom system:

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow (B \rightarrow (A \rightarrow C)) \\ & \neg\neg A \rightarrow A \\ & (A \rightarrow \neg A) \rightarrow \neg A \\ & (A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A) \end{aligned}$$

Sobociński's axiom systems:

- First:

$$\begin{aligned} & (A \rightarrow B) \rightarrow (\neg B \rightarrow (A \rightarrow C)) \\ & A \rightarrow (B \rightarrow (C \rightarrow A)) \\ & (\neg A \rightarrow B) \rightarrow ((A \rightarrow B) \rightarrow B) \end{aligned}$$

- Second:

$$\begin{aligned} & \neg A \rightarrow (A \rightarrow B) \\ & A \rightarrow (B \rightarrow (C \rightarrow A)) \\ & (\neg A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \rightarrow B) \rightarrow C)) \end{aligned}$$

Implication and falsum

Instead of negation, classical logic can also be formulated using the functionally complete set $\{\rightarrow, \perp\}$ of connectives.

Tarski-Bernays-Wajsberg axiom system:

$$\begin{aligned} (A \rightarrow B) &\rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)) \\ A &\rightarrow (B \rightarrow A) \\ ((A \rightarrow B) \rightarrow A) &\rightarrow A \\ \perp &\rightarrow A \end{aligned}$$

Church's axiom system:

$$\begin{aligned} A &\rightarrow (B \rightarrow A) \\ (A \rightarrow (B \rightarrow C)) &\rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ ((A \rightarrow \perp) \rightarrow \perp) &\rightarrow A \end{aligned}$$

Meredith's axiom systems:

- First:^{[4][5][6]}

$$(((A \rightarrow B) \rightarrow (C \rightarrow \perp)) \rightarrow D) \rightarrow E) \rightarrow ((E \rightarrow A) \rightarrow (C \rightarrow A))$$

- Second:

$$((A \rightarrow B) \rightarrow ((\perp \rightarrow C) \rightarrow D)) \rightarrow ((D \rightarrow A) \rightarrow (E \rightarrow (F \rightarrow A)))$$

Negation and disjunction

Instead of implication, classical logic can also be formulated using the functionally complete set $\{\neg, \vee\}$ of connectives. These formulations use the following rule of inference;

$$A, \neg A \vee B \vdash B.$$

Russell-Bernays axiom system:

$$\begin{aligned} \neg(\neg B \vee C) &\vee (\neg(A \vee B) \vee (A \vee C)) \\ \neg(A \vee B) &\vee (B \vee A) \\ \neg A &\vee (B \vee A) \\ \neg(A \vee A) &\vee A \end{aligned}$$

Meredith's axiom systems:^[7]

- First:

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

- Second:

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

- Third:

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

Dually, classical propositional logic can be defined using only conjunction and negation.

Sheffer's stroke

Because Sheffer's stroke (also known as NAND operator) is functionally complete, it can be used to create an entire formulation of propositional calculus. NAND formulations use a rule of inference called Nicod's modus ponens:

$$A, A \mid (B \mid C) \vdash C.$$

Nicod's axiom system:

$$(A \mid (B \mid C)) \mid [(E \mid (E \mid E)) \mid ((D \mid B) \mid [(A \mid D) \mid (A \mid D)])]$$

Łukasiewicz's axiom systems:

- First:

$$(A \mid (B \mid C)) \mid [(D \mid (D \mid D)) \mid ((D \mid B) \mid [(A \mid D) \mid (A \mid D)])]$$

- Second:

$$(A \mid (B \mid C)) \mid [(A \mid (C \mid A)) \mid ((D \mid B) \mid [(A \mid D) \mid (A \mid D)])]$$

Wajsberg's axiom system:

$$(A \mid (B \mid C)) \mid [((D \mid C) \mid [(A \mid D) \mid (A \mid D)]) \mid (A \mid (A \mid B))]$$

Argonne axiom systems:

- First:

$$(A \mid (B \mid C)) \mid [(A \mid (B \mid C)) \mid ((D \mid C) \mid [(C \mid D) \mid (A \mid D)])]$$

- Second:

$$(A \mid (B \mid C)) \mid [[[B \mid D) \mid (A \mid D)] \mid (D \mid B)] \mid ((C \mid B) \mid A)]^{[8]}$$

Computer analysis by Argonne has revealed > 60 additional single axiom systems that can be used to formulate NAND propositional calculus.

Implicational propositional calculus

The implicational propositional calculus is the fragment of the classical propositional calculus which only admits the implication connective. It is not functionally complete (because it lacks the ability to express falsity and negation) but it is however syntactically complete. The implicational calculi below use modus ponens as an inference rule.

Bernays–Tarski axiom system:

$$A \rightarrow (B \rightarrow A)$$

$$(A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$$

$$((A \rightarrow B) \rightarrow A) \rightarrow A$$

Łukasiewicz and Tarski's axiom systems:

- First:

$$[(A \rightarrow (B \rightarrow A)) \rightarrow [([(C \rightarrow D) \rightarrow E] \rightarrow F) \rightarrow [(D \rightarrow F) \rightarrow (C \rightarrow F)]] \rightarrow G]$$

- Second:

$$[(A \rightarrow B) \rightarrow ((C \rightarrow D) \rightarrow E)] \rightarrow ([F \rightarrow ((C \rightarrow D) \rightarrow E)] \rightarrow [(A \rightarrow F) \rightarrow (D \rightarrow E)])$$

- Third:

$$((A \rightarrow B) \rightarrow (C \rightarrow D)) \rightarrow (E \rightarrow ((D \rightarrow A) \rightarrow (C \rightarrow A)))$$

- Fourth:

$$((A \rightarrow B) \rightarrow (C \rightarrow D)) \rightarrow ((D \rightarrow A) \rightarrow (E \rightarrow (C \rightarrow A)))$$

Łukasiewicz's axiom system:

$$((A \rightarrow B) \rightarrow C) \rightarrow ((C \rightarrow A) \rightarrow (D \rightarrow A))$$

Intuitionistic and intermediate logics

Intuitionistic logic is a subsystem of classical logic. It is commonly formulated with $\{\rightarrow, \wedge, \vee, \perp\}$ as the set of (functionally complete) basic connectives. It is not syntactically complete since it lacks excluded middle $A \vee \neg A$ or Peirce's law $((A \rightarrow B) \rightarrow A) \rightarrow A$ which can be added without making the logic inconsistent. It has modus ponens as inference rule, and the following axioms:

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ & (A \wedge B) \rightarrow A \\ & (A \wedge B) \rightarrow B \\ & A \rightarrow (B \rightarrow (A \wedge B)) \\ & A \rightarrow (A \vee B) \\ & B \rightarrow (A \vee B) \\ & (A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C)) \\ & \perp \rightarrow A \end{aligned}$$

Alternatively, intuitionistic logic may be axiomatized using $\{\rightarrow, \wedge, \vee, \neg\}$ as the set of basic connectives, replacing the last axiom with

$$\begin{aligned} & (A \rightarrow \neg A) \rightarrow \neg A \\ & \neg A \rightarrow (A \rightarrow B) \end{aligned}$$

Intermediate logics are in between intuitionistic logic and classical logic. Here are a few intermediate logics:

- Jankov logic (KC) is an extension of intuitionistic logic, which can be axiomatized by the intuitionistic axiom system plus the axiom^[9]
- $$\neg A \vee \neg \neg A.$$
- Gödel–Dummett logic (LC) can be axiomatized over intuitionistic logic by adding the axiom
- $$(A \rightarrow B) \vee (B \rightarrow A).$$

Positive implicational calculus

The positive implicational calculus is the implicational fragment of intuitionistic logic. The calculi below use modus ponens as an inference rule.

Łukasiewicz's axiom system:

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \end{aligned}$$

Meredith's axiom systems:

- First:

$$E \rightarrow ((A \rightarrow B) \rightarrow (((D \rightarrow A) \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C)))$$

- Second:

$$\begin{aligned} & A \rightarrow (B \rightarrow A) \\ & (A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C)) \end{aligned}$$

- Third:

$$((A \rightarrow B) \rightarrow C) \rightarrow (D \rightarrow ((B \rightarrow (C \rightarrow E)) \rightarrow (B \rightarrow E)))^{[10]}$$

Hilbert's axiom systems:

- First:

$$\begin{aligned}(A \rightarrow (A \rightarrow B)) &\rightarrow (A \rightarrow B) \\ (B \rightarrow C) &\rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ (A \rightarrow (B \rightarrow C)) &\rightarrow (B \rightarrow (A \rightarrow C)) \\ A &\rightarrow (B \rightarrow A)\end{aligned}$$

- Second:

$$\begin{aligned}(A \rightarrow (A \rightarrow B)) &\rightarrow (A \rightarrow B) \\ (A \rightarrow B) &\rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)) \\ A &\rightarrow (B \rightarrow A)\end{aligned}$$

- Third:

$$\begin{aligned}A &\rightarrow A \\ (A \rightarrow B) &\rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C)) \\ (B \rightarrow C) &\rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C)) \\ (A \rightarrow (A \rightarrow B)) &\rightarrow (A \rightarrow B)\end{aligned}$$

Positive propositional calculus

Positive propositional calculus is the fragment of intuitionistic logic using only the (non functionally complete) connectives $\{\rightarrow, \wedge, \vee\}$. It can be axiomatized by any of the above mentioned calculi for positive implicational calculus together with the axioms

$$\begin{aligned}(A \wedge B) &\rightarrow A \\ (A \wedge B) &\rightarrow B \\ A &\rightarrow (B \rightarrow (A \wedge B)) \\ A &\rightarrow (A \vee B) \\ B &\rightarrow (A \vee B) \\ (A \rightarrow C) &\rightarrow ((B \rightarrow C) \rightarrow ((A \vee B) \rightarrow C))\end{aligned}$$

Optionally, we may also include the connective \leftrightarrow and the axioms

$$\begin{aligned}(A \leftrightarrow B) &\rightarrow (A \rightarrow B) \\ (A \leftrightarrow B) &\rightarrow (B \rightarrow A) \\ (A \rightarrow B) &\rightarrow ((B \rightarrow A) \rightarrow (A \leftrightarrow B))\end{aligned}$$

Johansson's minimal logic can be axiomatized by any of the axiom systems for positive propositional calculus and expanding its language with the nullary connective \perp , with no additional axiom schemas. Alternatively, it can also be axiomatized in the language $\{\rightarrow, \wedge, \vee, \neg\}$ by expanding the positive propositional calculus with the axiom

$$(A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A)$$

or the pair of axioms

$$\begin{aligned}(A \rightarrow B) &\rightarrow (\neg B \rightarrow \neg A) \\ A &\rightarrow \neg \neg A\end{aligned}$$

Intuitionistic logic in language with negation can be axiomatized over the positive calculus by the pair of axioms

$$\begin{aligned}(A \rightarrow \neg B) &\rightarrow (B \rightarrow \neg A) \\ \neg A &\rightarrow (A \rightarrow B)\end{aligned}$$

or the pair of axioms^[11]

$$(A \rightarrow \neg A) \rightarrow \neg A$$

$$\neg A \rightarrow (A \rightarrow B)$$

Classical logic in the language $\{\rightarrow, \wedge, \vee, \neg\}$ can be obtained from the positive propositional calculus by adding the axiom

$$(\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$$

or the pair of axioms

$$(A \rightarrow \neg B) \rightarrow (B \rightarrow \neg A)$$

$$\neg \neg A \rightarrow A$$

Fitch calculus takes any of the axiom systems for positive propositional calculus and adds the axioms

$$\neg A \rightarrow (A \rightarrow B)$$

$$A \leftrightarrow \neg \neg A$$

$$\neg(A \vee B) \leftrightarrow (\neg A \wedge \neg B)$$

$$\neg(A \wedge B) \leftrightarrow (\neg A \vee \neg B)$$

Note that the first and third axioms are also valid in intuitionistic logic.

Equivalential calculus

Equivalential calculus is the subsystem of classical propositional calculus that only allows the (functionally incomplete) equivalence connective, denoted here as \equiv . The rule of inference used in these systems is as follows:

$$A, A \equiv B \vdash B$$

Iséki's axiom system:^[12]

$$((A \equiv C) \equiv (B \equiv A)) \equiv (C \equiv B)$$

$$(A \equiv (B \equiv C)) \equiv ((A \equiv B) \equiv C)$$

Iséki-Arai axiom system:^[13]

$$A \equiv A$$

$$(A \equiv B) \equiv (B \equiv A)$$

$$(A \equiv B) \equiv ((B \equiv C) \equiv (A \equiv C))$$

Arai's axiom systems;

- First:

$$(A \equiv (B \equiv C)) \equiv ((A \equiv B) \equiv C)$$

$$((A \equiv C) \equiv (B \equiv A)) \equiv (C \equiv B)$$

- Second:

$$(A \equiv B) \equiv (B \equiv A)$$

$$((A \equiv C) \equiv (B \equiv A)) \equiv (C \equiv B)$$

Łukasiewicz's axiom systems:^[14]

- First:

$$(A \equiv B) \equiv ((C \equiv B) \equiv (A \equiv C))$$

- Second:

$$(A \equiv B) \equiv ((A \equiv C) \equiv (C \equiv B))$$

- Third:

$$(A \equiv B) \equiv ((C \equiv A) \equiv (B \equiv C))$$

Meredith's axiom systems:

- First:

$$((A \equiv B) \equiv C) \equiv (B \equiv (C \equiv A))$$

- Second:

$$A \equiv ((B \equiv (A \equiv C)) \equiv (C \equiv B))$$

- Third:

$$(A \equiv (B \equiv C)) \equiv (C \equiv (A \equiv B))$$

- Fourth:

$$(A \equiv B) \equiv (C \equiv ((B \equiv C) \equiv A))$$

- Fifth:

$$(A \equiv B) \equiv (C \equiv ((C \equiv B) \equiv A))$$

- Sixth:

$$((A \equiv (B \equiv C)) \equiv C) \equiv (B \equiv A)$$

- Seventh:

$$((A \equiv (B \equiv C)) \equiv B) \equiv (C \equiv A)$$

Kalman's axiom system:

$$A \equiv ((B \equiv (C \equiv A)) \equiv (C \equiv B))$$

Winker's axiom systems:

- First:

$$A \equiv ((B \equiv C) \equiv ((A \equiv C) \equiv B))$$

- Second:

$$A \equiv ((B \equiv C) \equiv ((C \equiv A) \equiv B))$$

XCB axiom system:

$$A \equiv (((A \equiv B) \equiv (C \equiv B)) \equiv C)$$

References

- [1] Yasuyuki Imai, Kiyoshi Iséki, On axiom systems of propositional calculi, I, Proceedings of the Japan Academy. Volume 41, Number 6 (1965), 436–439.
- [2] Part XIII: Shôtarô Tanaka. On axiom systems of propositional calculi, XIII. Proc. Japan Acad., Volume 41, Number 10 (1965), 904–907.
- [3] Elliott Mendelson, *Introduction to Mathematical Logic*, Van Nostrand, New York, 1979, p. 31.
- [4] [Fitelson, 2001] "New Elegant Axiomatizations of Some Sentential Logics" (<http://fitelson.org/ar.html>) by Branden Fitelson
- [5] (Computer analysis by Argonne has revealed this to be the shortest single axiom with least variables for propositional calculus).
- [6] "Some New Results in Logical Calculi Obtained Using Automated Reasoning", Zac Ernst, Ken Harris, & Branden Fitelson, <http://www.mcs.anl.gov/research/projects/AR/award-2001/fitelson.pdf>
- [7] C. Meredith, *Single axioms for the systems (C, N), (C, 0) and (A, N) of the two-valued propositional calculus*, Journal of Computing Systems, p. 155-164, 1954.
- [8] , p. 9, A Spectrum of Applications of Automated Reasoning (http://arxiv.org/PS_cache/cs/pdf/0205/0205078v1.pdf), Larry Wos; arXiv:cs/0205078v1
- [9] A. Chagrov, M. Zakharyashev, *Modal logic*, Oxford University Press, 1997.
- [10] C. Meredith, *A single axiom of positive logic*, Journal of Computing Systems, p. 169-170, 1954.
- [11] L. H. Hackstaff, *Systems of Formal Logic*, Springer, 1966.
- [12] Kiyoshi Iséki, On axiom systems of propositional calculi, XV, Proceedings of the Japan Academy. Volume 42, Number 3 (1966), 217–220.
- [13] Yoshinari Arai, On axiom systems of propositional calculi, XVII, Proceedings of the Japan Academy. Volume 42, Number 4 (1966), 351–354.
- [14] XCB, the Last of the Shortest Single Axioms for the Classical Equivalential Calculus (<http://www.mcs.anl.gov/uploads/cels/papers/P966.pdf>), LARRY WOS, DOLPH ULRICH, BRANDEN FITELSON; arXiv:cs/0211015v1

Literal (mathematical logic)

In mathematical logic, a **literal** is an atomic formula (atom) or its negation. The definition mostly appears in proof theory (of classical logic), e.g. in conjunctive normal form and the method of resolution.

Literals can be divided into two types:

- A **positive literal** is just an atom.
- A **negative literal** is the negation of an atom.

For a literal l , the **complementary literal** is a literal corresponding to the negation of l , we can write \bar{l} to denote the complementary literal of l . More precisely, if $l \equiv x$ then \bar{l} is $\neg x$ and if $l \equiv \neg x$ then \bar{l} is x .

In the context of a formula in the conjunctive normal form, a literal is **pure** if the literal's complement does not appear in the formula.

Examples

In propositional calculus a literal is simply a propositional variable or its negation.

In predicate calculus a literal is an atomic formula or its negation, where an atomic formula is a predicate symbol applied to some terms, $P(t_1, \dots, t_n)$ with the terms recursively defined starting from constant symbols, variable symbols, and function symbols. For example, $\neg Q(f(g(x), y, 2), x)$ is a negative literal with the constant symbol 2, the variable symbols x, y , the function symbols f, g , and the predicate symbol Q .

References

- Samuel R. Buss (1998). "An introduction to proof theory"^[1]. In Samuel R. Buss. *Handbook of proof theory*. Elsevier. pp. 1–78. ISBN 0-444-89840-9.

References

[1] <http://math.ucsd.edu/~sbuss/ResearchWeb/handbookI/>

Logical consequence

Logical consequence (also **entailment**) is one of the most fundamental concepts in logic. It is the relationship between statements that holds true when one logically "follows from" one or more others. Valid logical arguments are ones in which the conclusions follow from its premises, and its conclusions are consequences of its premises. The philosophical analysis of logical consequence involves asking, 'in what sense does a conclusion follow from its premises?' and 'what does it mean for a conclusion to be a consequence of premises?'^[1] All of philosophical logic can be thought of as providing accounts of the nature of logical consequence, as well as logical truth.^[2]

Logical consequence is taken to be both necessary and formal with examples explicated using models and proofs. A sentence is said to be a logical consequence of a set of sentences, for a given language, if and only if, using logic alone (i.e. without regard to any interpretations of the sentences) the sentence must be true if every sentence in the set were to be true.^[3]

Logicians make precise accounts of logical consequence with respect to a given language \mathcal{L} by constructing a deductive system for \mathcal{L} , or by formalizing the intended semantics for \mathcal{L} . Alfred Tarski highlighted three salient features for which any adequate characterization of logical consequence needs to account: 1) that the logical consequence relation relies on the logical form of the sentences involved, 2) that the relation is a priori, i.e. it can be determined whether or not it holds without regard to sense experience, and 3) that the relation has a modal component.

Formal accounts of logical consequence

The most widely prevailing view on how to best account for logical consequence is to appeal to formality. This is to say that whether statements follow from one another logically depends on the structure or logical form of the statements without regard to the contents of that form.

Syntactic accounts of logical consequence rely on schemes using inference rules. For instance, we can express the logical form of a valid argument as "All A are B . All C are A . Therefore, All C are B ." This argument is formally valid, because every instance of arguments constructed using this scheme are valid.

This is in contrast to an argument like "Fred is Mike's brother's son. Therefore Fred is Mike's nephew." Since this argument depends on the meanings of the words "brother", "son", and "nephew", the statement "Fred is Mike's nephew" is a so-called material consequence of "Fred is Mike's brother's son," not a formal consequence. A formal consequence must be true *in all cases*, however this is an incomplete definition of formal consequence, since even the argument " P is Q 's brother's son, therefore P is Q 's nephew" is valid in all cases, but is not a *formal* argument.

A priori property of logical consequence

If you know that Q follows logically from P no information about the possible interpretations of P or Q will affect that knowledge. Our knowledge that Q is a logical consequence of P cannot be influenced by empirical knowledge. Deductively valid arguments can be known to be so without recourse to experience, so they must be knowable a priori. However, formality alone does not guarantee that logical consequence is not influenced by empirical knowledge. So the a priori property of logical consequence is considered to be independent of formality.

Proofs and models

The two prevailing techniques for providing accounts of logical consequence involve expressing the concept in terms of *proofs* and via *models*. The study of the syntactic consequence (of a logic) is called (its) proof theory whereas the study of (its) semantic consequence is called (its) model theory.

Syntactic consequence

A formula A is a **syntactic consequence**^{[4][5][6][7]} within some formal system \mathcal{FS} of a set Γ of formulas if there is a formal proof in \mathcal{FS} of A from the set Γ .

$$\Gamma \vdash_{\mathcal{FS}} A$$

Syntactic consequence does not depend on any interpretation of the formal system.^[8]

Semantic consequence

A formula A is a **semantic consequence** within some formal system \mathcal{FS} of a set of statements Γ

$$\Gamma \models_{\mathcal{FS}} A,$$

if and only if there is no model \mathcal{I} in which all members of Γ are true and A is false.^[9] Or, in other words, the set of the interpretations that make all members of Γ true is a subset of the set of the interpretations that make A true.

Modal accounts

Modal accounts of logical consequence are variations on the following basic idea:

$\Gamma \vdash A$ is true if and only if it is *necessary* that if all of the elements of Γ are true, then A is true.

Alternatively (and, most would say, equivalently):

$\Gamma \vdash A$ is true if and only if it is *impossible* for all of the elements of Γ to be true and A false.

Such accounts are called "modal" because they appeal to the modal notions of logical necessity and logical possibility. 'It is necessary that' is often expressed as a universal quantifier over possible worlds, so that the accounts above translate as:

$\Gamma \vdash A$ is true if and only if there is no possible world at which all of the elements of Γ are true and A is false (untrue).

Consider the modal account in terms of the argument given as an example above:

All frogs are green.

Kermit is a frog.

Therefore, Kermit is green.

The conclusion is a logical consequence of the premises because we can't imagine a possible world where (a) all frogs are green; (b) Kermit is a frog; and (c) Kermit is not green.

Modal-formal accounts

Modal-formal accounts of logical consequence combine the modal and formal accounts above, yielding variations on the following basic idea:

$\Gamma \vdash A$ if and only if it is impossible for an argument with the same logical form as Γ / A to have true premises and a false conclusion.

Warrant-based accounts

The accounts considered above are all "truth-preservational," in that they all assume that the characteristic feature of a good inference is that it never allows one to move from true premises to an untrue conclusion. As an alternative, some have proposed "warrant-preservational" accounts, according to which the characteristic feature of a good inference is that it never allows one to move from justifiably assertible premises to a conclusion that is not justifiably assertible. This is (roughly) the account favored by intuitionists such as Michael Dummett.

Non-monotonic logical consequence

The accounts discussed above all yield monotonic consequence relations, i.e. ones such that if A is a consequence of Γ , then A is a consequence of any superset of Γ . It is also possible to specify non-monotonic consequence relations to capture the idea that, e.g., 'Tweety can fly' is a logical consequence of

{Birds can typically fly, Tweety is a bird}

but not of

{Birds can typically fly, Tweety is a bird, Tweety is a penguin}.

For more on this, see Belief revision#Non-monotonic inference relation.

References

- [1] Beall, JC and Restall, Greg, *Logical Consequence* (<http://plato.stanford.edu/archives/fall2009/entries/logical-consequence/>) The Stanford Encyclopedia of Philosophy (Fall 2009 Edition), Edward N. Zalta (ed.).
- [2] Quine, Willard Van Orman, *Philosophy of logic*
- [3] McKeon, Matthew, *Logical Consequence* (<http://www.iep.utm.edu/logcon/>) Internet Encyclopedia of Philosophy.
- [4] Dummett, Michael (1993) philosophy of language (http://books.google.com/books?id=EYP7uCZIRQYC&pg=PA82&lpg=PA82&dq=syntactic+consequence&source=bl&ots=Ms58438B6w&sig=FE38FCaZpRpAr18gtG7INX4wieM&hl=en&ei=qOy7SoLIEI7KsQPgnYG7BA&sa=X&oi=book_result&ct=result&resnum=6#v=onepage&q=syntactic%20consequence&f=false) Harvard University Press, p.82ff
- [5] Lear, Jonathan (1986) and Logical Theory (http://books.google.com/books?id=lXI7AAAAIAAJ&pg=PA1&lpg=PA1&dq=syntactic+consequence&source=bl&ots=8IYWyFYTN-&sig=wrOg75cFxQwn1Uq-8LShBNXf9w0&hl=en&ei=I-y7SpHtLZLotgOsnLHcBQ&sa=X&oi=book_result&ct=result&resnum=10#v=onepage&q=syntactic%20consequence&f=false) Cambridge University Press, 136p.
- [6] Creath, Richard, and Friedman, Michael (2007) Cambridge companion to Carnap (http://books.google.com/books?id=87BcFLgJmxMC&pg=PA189&lpg=PA189&dq=syntactic+consequence&source=bl&ots=Fn2zomcMZP&sig=8hnJWsJFysNhmWLskICo4IQDYAc&hl=en&ei=I-y7SpHtLZLotgOsnLHcBQ&sa=X&oi=book_result&ct=result&resnum=6#v=onepage&q=syntactic%20consequence&f=false) Cambridge University Press, 371p.
- [7] FOLDOC: "syntactic consequence" (<http://www.swif.uniba.it/lei/foldop/foldoc.cgi?syntactic+consequence>)
- [8] Hunter, Geoffrey, *Metalogic: An Introduction to the Metatheory of Standard First-Order Logic*, University of California Pres, 1971, p. 75.
- [9] Etchemendy, John, *Logical consequence*, The Cambridge Dictionary of Philosophy

Nicod's axiom

Nicod's axiom is an axiom in propositional calculus that can be used as a sole wff in a two-axiom formalization of zeroth-order logic.

The axiom states the following always has a true truth value.

$$((\varphi \sqcap (\chi \sqcap \psi)) \sqcap ((\tau \sqcap (\tau \sqcap \tau)) \sqcap ((\theta \sqcap \chi) \sqcap ((\varphi \sqcap \theta) \sqcap (\varphi \sqcap \theta))))^{[1]}$$

To utilize this axiom, Nicod made a rule of inference, called Nicod's Modus Ponens.

1. φ
2. $(\varphi \sqcap (\chi \sqcap \psi))$
- $\therefore \psi^{[2]}$

In 1931, Mordechaj Wajsberg found an adequate, and easier-to-work-with alternative.

$$((\varphi \sqcap (\psi \sqcap \chi)) \sqcap (((\tau \sqcap \chi) \sqcap ((\varphi \sqcap \tau) \sqcap (\varphi \sqcap \tau))) \sqcap (\varphi \sqcap (\varphi \sqcap \psi))))^{[3]}$$

[1] <http://us.metamath.org/mpegin/nic-ax.html>

[2] <http://us.metamath.org/mpegin/nic-mp.html>

[3] <http://www.wolframscience.com/nksonline/page-1151a-text>

Open sentence

In mathematics, an **open sentence** (usually an equation or equality) is described as "open" in the sense that its truth value is meaningless until its variables are replaced with specific numbers, at which point the truth value can usually be determined (and hence the sentences are no longer regarded as "open"). These possible replacement values are assumed to range over a subset of either the real or complex numbers, depending on the equation or inequality under consideration (in applications, real numbers are usually associated also with measurement units). The replacement values which produce a true equation or inequality are called solutions of the equation or inequality, and are said to "satisfy" it.

In mathematical logic, a non-closed formula is a formula which contains free variables. (Note that in logic, a "sentence" is a formula *without* free variables, and a formula is "open" if it contains no quantifiers, which disagrees with the terminology of this article.) Unlike closed formulas, which contain constants, non-closed formulas do not express propositions; they are neither true nor false. Hence, the formula

(1) x is a number

has no truth-value. A formula is said to be *satisfied* by any object(s) such that if it is written in place of the variable(s), it will form a sentence expressing a true proposition. Hence, "5" satisfies (1). Any sentence which results from a formula in such a way is said to be a *substitution instance* of that formula. Hence, "5 is a number" is a substitution instance of (1).

Mathematicians have not adopted that nomenclature, but refer instead to *equations*, *inequalities* with free variables, etc.

Such replacements are known as *solutions* to the sentence. An *identity* is an open sentence for which every number is a solution.

Examples of open sentences include:

1. $3x - 9 = 21$, whose only solution for x is 10;
2. $4x + 3 > 9$, whose solutions for x are all numbers greater than $3/2$;
3. $x + y = 0$, whose solutions for x and y are all pairs of numbers that are additive inverses;
4. $3x + 9 = 3(x + 3)$, whose solutions for x are all numbers.

5. $3x + 9 = 3(x + 4)$, which has no solution.

Example 4 is an identity. Examples 1, 3, and 4 are equations, while example 2 is an inequality. Example 5 is a contradiction.

Every open sentence must have (usually implicitly) a *universe of discourse* describing which numbers are under consideration as solutions. For instance, one might consider all real numbers or only integers. For example, in example 2 above, 1.6 is a solution if the universe of discourse is all real numbers, but not if the universe of discourse is only integers. In that case, only the integers greater than $3/2$ are solutions: 2, 3, 4, and so on. On the other hand, if the universe of discourse consists of all complex numbers, then example 2 doesn't even make sense (although the other examples do). An identity is only required to hold for the numbers in its universe of discourse.

This same universe of discourse can be used to describe the solutions to the open sentence in symbolic logic using universal quantification. For example, the solution to example 2 above can be specified as:

For all x , $4x + 3 > 9$ if and only if $x > 3/2$.

Here, the phrase "for all" implicitly requires a universe of discourse to specify *which* mathematical objects are "all" the possibilities for x .

The idea can even be generalised to situations where the variables don't refer to numbers at all, as in a functional equation. For example of this, consider

$$f * f = f,$$

which says that $f(x) * f(x) = f(x)$ for every value of x . If the universe of discourse consists of all functions from the real line \mathbf{R} to itself, then the solutions for f are all functions whose only values are one and zero. But if the universe of discourse consists of all continuous functions from \mathbf{R} to itself, then the solutions for f are only the constant functions with value one or zero.

References

- Definition at The Math Resource ^[1]
- The Math Forum @ Drexel ^[2]

References

[1] http://www.mathresources.com/products/mathresource/maa/open_sentence.html

[2] <http://mathforum.org/library/drmath/view/53280.html>

Predicate (mathematical logic)

In mathematics, a **predicate** is commonly understood to be a Boolean-valued function $P: X \rightarrow \{\text{true, false}\}$, called the predicate on X . However, predicates have many different uses and interpretations in mathematics and logic, and their precise definition, meaning and use will vary from theory to theory. So, for example, when a theory defines the concept of a relation, then a predicate is simply the characteristic function or the indicator function of a relation. However, not all theories have relations, or are founded on set theory, and so one must be careful with the proper definition and semantic interpretation of a predicate.

Simplified overview

Informally, a **predicate** is a statement that may be true or false depending on the values of its variables. It can be thought of as an operator or function that returns a value that is either true or false. For example, predicates are sometimes used to indicate set membership: when talking about sets, it is sometimes inconvenient or impossible to describe a set by listing all of its elements. Thus, a predicate $P(x)$ will be true or false, depending on whether x belongs to a set.

Predicates are also commonly used to talk about the properties of objects, by defining the set of all objects that have some property in common. So, for example, when P is a predicate on X , one might sometimes say P is a property of X . Similarly, the notation $P(x)$ is used to denote a sentence or statement P concerning the variable object x . The set defined by $P(x)$ is written as $\{x \mid P(x)\}$, and is just a collection of all the objects for which P is true.

For instance, $\{x \mid x \text{ is a positive integer less than } 4\}$ is the set $\{1, 2, 3\}$.

If t is an element of the set $\{x \mid P(x)\}$, then the statement $P(t)$ is *true*.

Here, $P(x)$ is referred to as the *predicate*, and x the *subject* of the *proposition*. Sometimes, $P(x)$ is also called a propositional function, as each choice of x produces a proposition.

Formal definition

The precise semantic interpretation of an atomic formula and an atomic sentence will vary from theory to theory.

- In propositional logic, atomic formulas are called propositional variables. In a sense, these are nullary (i.e. 0-arity) predicates.
- In first-order logic, an atomic formula consists of a predicate symbol applied to an appropriate number of terms.
- In set theory, predicates are understood to be characteristic functions or set indicator functions, *i.e.* functions from a set element to a truth value. Set-builder notation makes use of predicates to define sets.
- In autoepistemic logic, which rejects the law of excluded middle, predicates may be true, false, or simply *unknown*; *i.e.* a given collection of facts may be insufficient to determine the truth or falsehood of a predicate.
- In fuzzy logic, predicates are the characteristic functions of a probability distribution. That is, the strict true/false valuation of the predicate is replaced by a quantity interpreted as the degree of truth.

References

External links

- Introduction to predicates (http://cs.odu.edu/~toida/nerzic/content/logic/pred_logic/predicate/pred_intro.html)

Principle of distributivity

The **principle of distributivity** states that the algebraic distributive law is valid for classical logic, where both logical conjunction and logical disjunction are distributive over each other so that for any propositions A , B and C the equivalences

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

and

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

hold.

The principle of distributivity is valid in classical logic, but invalid in quantum logic. The article *Is logic empirical?* discusses the case that quantum logic is the correct, empirical logic, on the grounds that the principle of distributivity is inconsistent with a reasonable interpretation of quantum phenomena.

Proof by contrapositive

In logic, the **contrapositive** of a conditional statement is formed by negating both terms and reversing the direction of inference. Explicitly, the contrapositive of the statement "if A , then B " is "if not B , then not A ." A statement and its contrapositive are logically equivalent: if the statement is true, then its contrapositive is true, and vice versa.^[1]

In mathematics, **proof by contraposition** is a rule of inference used in proofs. This rule infers a conditional statement from its contrapositive. In other words, the conclusion "if A , then B " is drawn from the single premise "if not B , then not A ."

Example

Let x be an integer.

To prove: *If x^2 is even, then x is even.*

Although a direct proof can be given, we choose to prove this statement by contraposition. The contrapositive of the above statement is:

If x is not even, then x^2 is not even.

This latter statement can be proven as follows. Suppose x is not even. Then x is odd. The product of two odd numbers is odd, hence $x^2 = x \cdot x$ is odd. Thus x^2 is not even.

Having proved the contrapositive, we infer the original statement.^[2]

Relation to proof by contradiction

Any proof by contrapositive can also be trivially formulated in terms of a Proof by contradiction: To prove the proposition $P \Rightarrow Q$, we consider the opposite, $\neg(P \Rightarrow Q) \equiv \neg(\neg P \vee Q) \equiv P \wedge \neg Q$. Since we have a proof that $\neg Q \Rightarrow \neg P$, we have $P \wedge \neg Q \Rightarrow P \wedge \neg P \equiv \perp$ which arrives at the contradiction we want. So proof by contrapositive is in some sense "at least as hard to formulate" as proof by contradiction.

References

- [1] Regents Exam Prep, contrapositive (<http://www.regentsprep.org/Regents/math/geometry/GP2/Lcontrap.htm>) definition
- [2] (p. 50).

Proposition

The term '**proposition**' has a broad use in contemporary philosophy. It is used to refer to some or all of the following: the primary bearers of truth-value, the objects of belief and other "propositional attitudes" (i.e., what is believed, doubted, etc.[1]), the referents of that-clauses, and the meanings of sentences. Propositions are the sharable objects of the attitudes and the primary bearers of truth and falsity. This stipulation rules out certain candidates for propositions, including thought- and utterance-tokens, which presumably are not sharable, and concrete events or facts, which presumably cannot be false.^[1]

Historical usage

Usage by Aristotle

Aristotelian logic identifies a proposition as a sentence which affirms or denies a predicate of a subject. An Aristotelian proposition may take the form "All men are mortal" or "Socrates is a man." In the first example the subject is "All men" and the predicate "are mortal." In the second example the subject is "Socrates" and the predicate is "is a man."

Usage by the logical positivists

Often propositions are related to closed sentences to distinguish them from what is expressed by an open sentence. In this sense, propositions are "statements" that are truth bearers. This conception of a proposition was supported by the philosophical school of logical positivism.

Some philosophers argue that some (or all) kinds of speech or actions besides the declarative ones also have propositional content. For example, yes–no questions present propositions, being inquiries into the truth value of them. On the other hand, some signs can be declarative assertions of propositions without forming a sentence nor even being linguistic, e.g. traffic signs convey definite meaning which is either true or false.

Propositions are also spoken of as the content of beliefs and similar intentional attitudes such as desires, preferences, and hopes. For example, "I desire that I have a new car," or "I wonder whether it will snow" (or, whether it is the case that "it will snow"). Desire, belief, and so on, are thus called propositional attitudes when they take this sort of content.

Usage by Russell

Bertrand Russell held that propositions were structured entities with objects and properties as constituents. Wittgenstein held that a proposition is the set of possible worlds/states of affairs in which it is true. One important difference between these views is that on the Russellian account, two propositions that are true in all the same states of affairs can still be differentiated. For instance, the proposition that two plus two equals four is distinct on a Russellian account from three plus three equals six. If propositions are sets of possible worlds, however, then all mathematical truths (and all other necessary truths) are the same set (the set of all possible worlds).

Relation to the mind

In relation to the mind, propositions are discussed primarily as they fit into propositional attitudes. Propositional attitudes are simply attitudes characteristic of folk psychology (belief, desire, etc.) that one can take toward a proposition (e.g. 'it is raining,' 'snow is white,' etc.). In English, propositions usually follow folk psychological attitudes by a "that clause" (e.g. "Jane believes *that* it is raining"). In philosophy of mind and psychology, mental states are often taken to primarily consist in propositional attitudes. The propositions are usually said to be the "mental content" of the attitude. For example, if Jane has a mental state of believing that it is raining, her mental content is the proposition 'it is raining.' Furthermore, since such mental states are *about* something (namely propositions), they are said to be intentional mental states. Philosophical debates surrounding propositions as they relate to propositional attitudes have also recently centered on whether they are internal or external to the agent or whether they are mind-dependent or mind-independent entities (see the entry on internalism and externalism in philosophy of mind).

Treatment in logic

As noted above, in Aristotelian logic a proposition is a particular kind of sentence, one which affirms or denies a predicate of a subject. Aristotelian propositions take forms like "All men are mortal" and "Socrates is a man."

Propositions show up in formal logic as objects of a formal language. A formal language begins with different types of symbols. These types can include variables, operators, function symbols, predicate (or relation) symbols, quantifiers, and propositional constants. (Grouping symbols are often added for convenience in using the language but do not play a logical role.) Symbols are concatenated together according to recursive rules in order to construct strings to which truth-values will be assigned. The rules specify how the operators, function and predicate symbols, and quantifiers are to be concatenated with other strings. A proposition is then a string with a specific form. The form that a proposition takes depends on the type of logic.

The type of logic called propositional, sentential, or statement logic includes only operators and propositional constants as symbols in its language. The propositions in this language are propositional constants, which are considered atomic propositions, and composite propositions, which are composed by recursively applying operators to propositions. *Application* here is simply a short way of saying that the corresponding concatenation rule has been applied.

The types of logics called predicate, quantificational, or *n*-order logic include variables, operators, predicate and function symbols, and quantifiers as symbols in their languages. The propositions in these logics are more complex. First, terms must be defined. A term is (i) a variable or (ii) a function symbol applied to the number of terms required by the function symbol's arity. For example, if + is a binary function symbol and x , y , and z are variables, then $x+(y+z)$ is a term, which might be written with the symbols in various orders. A proposition is (i) a predicate symbol applied to the number of terms required by its arity, (ii) an operator applied to the number of propositions required by its arity, or (iii) a quantifier applied to a proposition. For example, if = is a binary predicate symbol and \exists is a quantifier, then $\exists x,y,z [(x = y) \rightarrow (x+z = y+z)]$ is a proposition. This more complex structure of propositions allows these logics to make finer distinctions between inferences, i.e., to have greater expressive power.

In this context, propositions are also called sentences, statements, statement forms, formulas, and well-formed formulas, though these terms are usually not synonymous within a single text. This definition treats propositions as syntactic objects, as opposed to semantic or mental objects. That is, propositions in this sense are meaningless, formal, abstract objects. They are assigned meaning and truth-values by mappings called interpretations and valuations, respectively.

Objections to propositions

Attempts to provide a workable definition of proposition include

Two meaningful declarative sentences express the same proposition if and only if they mean the same thing.

thus defining *proposition* in terms of synonymy. For example, "Snow is white" (in English) and "Schnee ist weiß" (in German) are different sentences, but they say the same thing, so they express the same proposition.

Two meaningful declarative sentence-tokens express the same proposition if and only if they mean the same thing.

Unfortunately, the above definition has the result that two sentences/sentence-tokens which have the same meaning and thus express the same proposition, could have different truth-values, e.g. "I am Spartacus" said by Spartacus and said by John Smith; and e.g. "It is Wednesday" said on a Wednesday and on a Thursday.

A number of philosophers and linguists claim that all definitions of a proposition are too vague to be useful. For them, it is just a misleading concept that should be removed from philosophy and semantics. W.V. Quine maintained that the indeterminacy of translation prevented any meaningful discussion of propositions, and that they should be discarded in favor of sentences.^[2] Strawson advocated the use of the term "statement".

Related concepts

Facts are verifiable information.^{[citation needed][3]} Simple facts are often stated as propositions: "Apples are a type of fruit." The opposite statement—"Apples are not a type of fruit"—is still a properly formulated proposition, even though it is false (not a fact). Most statements of fact are compound facts: e.g., that apples exist, that fruit exists, that there are multiple types of fruit, etc.

A premise is a proposition that is used as the foundation for drawing conclusions. For example:

- *Premise*: "Apples are a type of fruit."
- *Premise*: "All types of fruit are food."
- *Conclusion*: "Therefore, apples are food."

If the conclusion is false then either one or more of the premises is false or the process of combining the premises is logically invalid. If the premises are true and the process is logically valid, then the conclusion .

References

- [1] see e.g. <http://plato.stanford.edu/entries/propositions/>
- [2] Quine W.V. *Philosophy of Logic*, Prentice-Hall NJ USA: 1970, pp 1-14
- [3] <http://plato.stanford.edu/entries/propositions/#nature>

External links

- Stanford Encyclopedia of Philosophy articles on:
 - Propositions (<http://plato.stanford.edu/entries/propositions/>), by Matthew McGrath
 - Singular Propositions (<http://plato.stanford.edu/entries/propositions-singular/>), by Greg Fitch
 - Structured Propositions (<http://plato.stanford.edu/entries/propositions-structured/>), by Jeffrey C. King

Propositional proof system

In propositional calculus and proof complexity a **propositional proof system (pps)**, also called a **Cook–Reckhow propositional proof system**, is system for proving classical propositional tautologies.

Mathematical definition

Formally a pps is a polynomial-time function P whose range is the set of all propositional tautologies (denoted TAUT). If A is a formula, then any x such that $P(x) = A$ is called a P -proof of A . The condition defining pps can be broken up as follows:

- Completeness: every propositional tautology has a P -proof,
- Soundness: if a propositional formula has a P -proof then it is a tautology,
- Efficiency: P runs in polynomial time.

In general, a proof system for a language L is a polynomial-time function whose range is L . Thus, a propositional proof system is a proof system for TAUT.

Sometimes the following alternative definition is considered: a pps is given as a proof-verification algorithm $P(A,x)$ with two inputs. If P accepts the pair (A,x) we say that x is a P -proof of A . P is required to run in polynomial time, and moreover, it must hold that A has a P -proof if and only if it is a tautology.

If P_1 is a pps according to the first definition, then P_2 defined by $P_2(A,x)$ if and only if $P_1(x) = A$ is a pps according to the second definition. Conversely, if P_2 is a pps according to the second definition, then P_1 defined by

$$P_1(\langle x, A \rangle) = \begin{cases} A & \text{if } P_2(A, x) \\ \top & \text{otherwise} \end{cases}$$

(P_1 takes pairs as input) is a pps according to the first definition, where \top is a fixed tautology.

Algorithmic interpretation

One can view the second definition as a non-deterministic algorithm for solving membership in TAUT. This means that proving a superpolynomial proof size lower-bound for pps would rule out existence of a certain class of polynomial-time algorithms based on that pps.

As an example, exponential proof size lower-bounds in resolution for the pigeon hole principle imply that any algorithm based on resolution cannot decide TAUT or SAT efficiently and will fail on pigeon hole principle tautologies. This is significant because the class of algorithms based on resolution includes most of current propositional proof search algorithms and modern industrial SAT solvers.

History

Historically, Frege's propositional calculus was the first propositional proof system. The general definition of a propositional proof system is due to Stephen Cook and Robert A. Reckhow (1979).

Relation with computational complexity theory

Propositional proof system can be compared using the notion of p-simulation. A propositional proof system P *p-simulates* Q (written as $P \leq_p Q$) when there is a polynomial-time function F such that $P(F(x)) = Q(x)$ for every x . That is, given a Q -proof x , we can find in polynomial time a P -proof of the same tautology. If $P \leq_p Q$ and $Q \leq_p P$, the proof systems P and Q are *p-equivalent*. There is also a weaker notion of simulation: a pps P *simulates* or *weakly p-simulates* a pps Q if there is a polynomial p such that for every Q -proof x of a tautology A , there is a P -proof y of A such that the length of y , $|y|$ is at most $p(|x|)$. (Some authors use the words p-simulation and simulation interchangeably for either of these two concepts, usually the latter.)

A propositional proof system is called *p-optimal* if it *p-simulates* all other propositional proof systems, and it is *optimal* if it simulates all other pps. A propositional proof system P is *polynomially bounded* (also called super) if every tautology has a short (i.e., polynomial-size) P -proof.

If P is polynomially bounded and Q simulates P , then Q is also polynomially bounded.

The set of propositional tautologies, TAUT, is a coNP-complete set. A propositional proof system is a certificate-verifier for membership in TAUT. Existence of a polynomially bounded propositional proof system means that there is a verifier with polynomial-size certificates, i.e., TAUT is in NP. In fact these two statements are equivalent, i.e., there is a polynomially bounded propositional proof system if and only if the complexity classes NP and coNP are equal.

Some equivalence classes of proof systems under simulation or *p*-simulation are closely related to theories of bounded arithmetic; they are essentially "non-uniform" versions of the bounded arithmetic, in the same way that circuit classes are non-uniform versions of resource-based complexity classes. "Extended Frege" systems (allowing the introduction of new variables by definition) correspond in this way to polynomially-bounded systems, for example. Where the bounded arithmetic in turn corresponds to a circuit-based complexity class, there are often similarities between the theory of proof systems and the theory of the circuit families, such as matching lower bound results and separations. For example, just as counting cannot be done by an AC^0 circuit family of subexponential size, many tautologies relating to the pigeonhole principle cannot have subexponential proofs in a proof system based on bounded-depth formulas (and in particular, not by resolution-based systems, since they rely solely on depth 1 formulas).

Examples of propositional proof systems

Some examples of propositional proof systems studied are:

- Propositional Resolution and various restrictions and extensions of it like DPLL algorithm
- Natural deduction
- Sequent calculus
- Frege system
- Extended Frege
- Polynomial calculus
- Nullstellensatz system
- Cutting-plane method

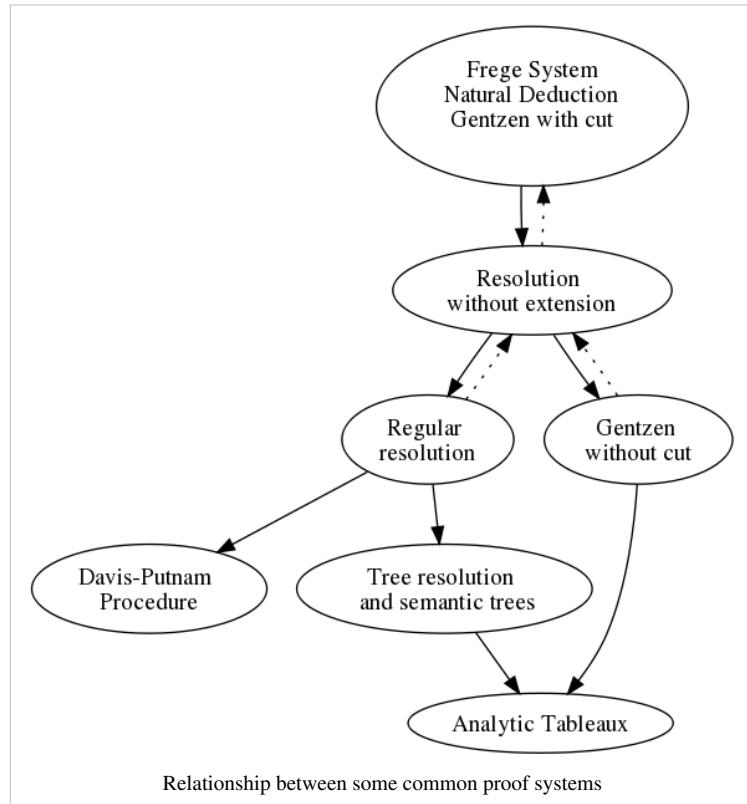
References

Further reading

- Samuel Buss (1998), "An introduction to proof theory", in: *Handbook of Proof Theory* (ed. S.R.Buss), Elsevier (1998).
- P. Pudlák (1998), "The lengths of proofs", in: *Handbook of Proof Theory* (ed. S.R.Buss), Elsevier, (1998).
- P. Beame and T. Pitassi (1998). Propositional proof complexity: past, present and future (<http://eccc.uni-trier.de/eccc-reports/1998/TR98-067/index.html>). Technical Report TR98-067, Electronic Colloquium on Computational Complexity.
- Nathan Segerlind (2007) "The Complexity of Propositional Proofs" (<http://www.math.ucla.edu/~asl/bsl/1304/1304-001.ps>), *Bulletin of Symbolic Logic* 13(4): 417–481
- J. Krajíček (1995), *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, Cambridge University Press.
- J. Krajíček, Proof complexity (<http://www.karlin.mff.cuni.cz/~krajicek/ecm.pdf>), in: Proc. 4th European congress of mathematics (ed. A. Laptev), EMS, Zurich, pp. 221–231, (2005).
- J. Krajíček, Propositional proof complexity I. (<http://www.karlin.mff.cuni.cz/~krajicek/ds1.ps>) and Proof complexity and arithmetic (<http://www.karlin.mff.cuni.cz/~krajicek/ds2.ps>).
- Stephen Cook and Phuong Nguyen, Logical Foundations of Proof Complexity (<http://www.cup.es/us/catalogue/catalogue.asp?isbn=9780521517294>), Cambridge University Press, 2010 (draft from 2008 (<http://www.cs.toronto.edu/~sacook/homepage/book>))

External links

- Proof Complexity (<http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume21/dixon04a-html/node9.html>)



Propositional variable

In mathematical logic, a **propositional variable** (also called a **sentential variable** or **sentential letter**) is a variable which can either be **true** or **false**. Propositional variables are the basic building-blocks of propositional formulas, used in propositional logic and higher logics.

Formulas in logic are typically built up recursively from some propositional variables, some number of logical connectives, and some logical quantifiers. Propositional variables are the atomic formulas of propositional logic. For example, in a given propositional logic, we might define a formula as follows:

- Every propositional variable is a formula.
- Given a formula X the negation $\neg X$ is a formula.
- Given two formulas X and Y , and a binary connective b (such as the logical conjunction \wedge), then $(X b Y)$ is a formula. (Note the parentheses.)

In this way, all of the formulas of propositional logic are built up from propositional variables as a basic unit. Propositional variables should not be confused with the metavariables which appear in the typical axioms of propositional calculus; the latter effectively range over well-formed formulae.

Propositional variables are represented as nullary predicates in first order logic.

References

- Smullyan, Raymond M. *First-Order Logic*. 1968. Dover edition, 1995. Chapter 1.1: Formulas of Propositional Logic.

Rule of inference

Transformation rules
Propositional calculus

Rules of inference
• <i>Modus ponens</i>
• <i>Modus tollens</i>
• Biconditional introduction
• Biconditional elimination
• Conjunction introduction
• Simplification
• Disjunction introduction
• Disjunction elimination
• Disjunctive syllogism
• Hypothetical syllogism
• Constructive dilemma
• Destructive dilemma
• Absorption
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

In logic, a **rule of inference**, **inference rule**, or **transformation rule** is the act of drawing a conclusion based on the form of premises interpreted as a function which takes premises, analyzes their syntax, and returns a conclusion (or conclusions). For example, the rule of inference modus ponens takes two premises, one in the form of "If p then q " and another in the form of " p " and returns the conclusion " q ". The rule is valid with respect to the semantics of classical logic (as well as the semantics of many other non-classical logics), in the sense that if the premises are true (under an interpretation) then so is the conclusion.

Typically, a rule of inference preserves truth, a semantic property. In many-valued logic, it preserves a general designation. But a rule of inference's action is purely syntactic, and does not need to preserve any semantic property: any function from sets of formulae to formulae counts as a rule of inference. Usually only rules that are recursive are important; i.e. rules such that there is an effective procedure for determining whether any given formula is the conclusion of a given set of formulae according to the rule. An example of a rule that is not effective in this sense is the infinitary ω -rule.

Popular rules of inference in propositional logic include modus ponens, modus tollens, and contraposition. First-order predicate logic uses rules of inference to deal with logical quantifiers. See List of rules of inference for examples.

The standard form of rules of inference

In formal logic (and many related areas), rules of inference are usually given in the following standard form:

Premise#1

Premise#2

...

Premise#n

Conclusion

This expression states that whenever in the course of some logical derivation the given premises have been obtained, the specified conclusion can be taken for granted as well. The exact formal language that is used to describe both premises and conclusions depends on the actual context of the derivations. In a simple case, one may use logical formulae, such as in:

$A \rightarrow B$

A

B

This is just the modus ponens rule of propositional logic. Rules of inference are often formulated as schemata employing of metavariables. In the rule (schema) above, the metavariables A and B can be instantiated to any element of the universe (or sometimes, by convention, some restricted subset such as propositions) to form an infinite set of inference rules. A proof system is formed from a set of rules chained together to form proofs, or *derivations*. Any derivation has only one final conclusion, which is the statement proved or derived. If premises are left unsatisfied in the derivation, then the derivation is a proof of a *hypothetical* statement: "if the premises hold, then the conclusion holds."

Axiom schemas and axioms

Inference rules may also be stated in this form: (1) some (perhaps zero) premises, (2) a turnstile symbol \vdash , which means "infers", "proves" or "concludes", (3) a conclusion. This usually embodies the relational (as opposed to functional) view of a rule of inference, where the turnstile stands for a deducibility relation holding between premises and conclusion.

An inference rule containing no premises is called an axiom schema or it if contains no metavariables simply an axiom.

Rules of inference must be distinguished from axioms of a theory. In terms of semantics, axioms are valid assertions. Axioms are usually regarded as starting points for applying rules of inference and generating a set of conclusions. Or, in less technical terms:

Rules are statements ABOUT the system, axioms are statements IN the system. For example:

- The RULE that from $\vdash p$ you can infer $\vdash \text{Provable}(p)$ is a statement that says if you've proven p, then it is provable that p is provable. This holds in Peano arithmetic, for example.
- The Axiom $p \rightarrow \text{Provable}(p)$ would mean that every true statement is provable. This does not hold in Peano arithmetic.

Rules of inference play a vital role in the specification of logical calculi as they are considered in proof theory, such as the sequent calculus and natural deduction.

Example: Hilbert systems for two propositional logics

In a Hilbert system the premises and conclusion of the inference rules are simply formulas of some language, usually employing metavariables. For graphical compactness of the presentation and to emphasize the distinction between axioms and rules of inference we use the sequent notation (\vdash) in this section instead of a vertical presentation of rules.

The formal language for classical propositional logic can be expressed using just negation (\neg), implication (\rightarrow) and propositional symbols. A well-known axiomatization, comprising three axiom schema and one inference rule (modus ponens) is:

$$(CA1) \quad \vdash A \rightarrow (B \rightarrow A)$$

$$(CA2) \quad \vdash (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$$

$$(CA3) \quad \vdash (\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$$

$$(MP) \quad A, \quad A \rightarrow B \vdash B$$

It may seem redundant to have two notions of inference in this case, \vdash and \rightarrow . In classical propositional logic they indeed coincide; the deduction theorem states that $A \vdash B$ if and only if $\vdash A \rightarrow B$. There is however a distinction worth emphasizing even in this case: the first notation describes a deduction, that is an activity of passing from sentences to sentences, whereas $A \rightarrow B$ is simply a formula made with a logical connective, implication in this case. Without an inference rule (like modus ponens in this case), there is no deduction or inference; this point is also illustrated in *What the Tortoise Said to Achilles*.^[1]

For some non-classical logics, the deduction theorem does not hold. For example, the three-valued logic Ł3 of Łukasiewicz can be axiomatized as:

$$(CA1) \quad \vdash A \rightarrow (B \rightarrow A)$$

$$(LA2) \quad \vdash (A \rightarrow B) \rightarrow ((B \rightarrow C) \rightarrow (A \rightarrow C))$$

$$(CA3) \quad \vdash (\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A)$$

$$(LA4) \quad \vdash ((A \rightarrow \neg A) \rightarrow A) \rightarrow A$$

$$(MP) \quad A, \quad A \rightarrow B \vdash B$$

This differs from classical logic by the change in axiom 2 and the addition of axiom 4. The classical deduction theorem does not hold for this logic, however a modified form does hold, namely $A \vdash B$ if and only if $\vdash A \rightarrow (A \rightarrow B)$.

Admissibility and derivability

In a set of rules, an inference rule could be redundant in the sense that it is *admissible* or *derivable*. A derivable rule is one whose conclusion can be derived from its premises using the other rules. An admissible rule is one whose conclusion holds whenever the premises hold. All derivable rules are admissible. To appreciate the difference, consider the following set of rules for defining the natural numbers (the judgment $n \text{ nat}$ asserts the fact that n is a natural number):

$$\frac{}{\mathbf{0} \text{ nat}} \quad \frac{n \text{ nat}}{s(n) \text{ nat}}$$

The first rule states that **0** is a natural number, and the second states that **s(n)** is a natural number if **n** is. In this proof system, the following rule demonstrating that the second successor of a natural number is also a natural number, is derivable:

$$\frac{n \text{ nat}}{\mathbf{s}(\mathbf{s}(n)) \text{ nat}}$$

Its derivation is just the composition of two uses of the successor rule above. The following rule for asserting the existence of a predecessor for any nonzero number is merely admissible:

$$\frac{\mathbf{s}(n) \text{ nat}}{n \text{ nat}}$$

This is a true fact of natural numbers, as can be proven by induction. (To prove that this rule is admissible, assume a derivation of the premise and induct on it to produce a derivation of **n nat**.) However, it is not derivable, because it depends on the structure of the derivation of the premise. Because of this, derivability is stable under additions to the proof system, whereas admissibility is not. To see the difference, suppose the following nonsense rule were added to the proof system:

$$\frac{}{\mathbf{s}(-\mathbf{3}) \text{ nat}}$$

In this new system, the double-successor rule is still derivable. However, the rule for finding the predecessor is no longer admissible, because there is no way to derive **-3 nat**. The brittleness of admissibility comes from the way it is proved: since the proof can induct on the structure of the derivations of the premises, extensions to the system add new cases to this proof, which may no longer hold.

Admissible rules can be thought of as theorems of a proof system. For instance, in a sequent calculus where cut elimination holds, the *cut* rule is admissible.

References

- [1] preprint (with different pagination) (<http://www.mi.sanu.ac.rs/~kosta/LOGCONS.pdf>)

Rule of replacement

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In logic, a **rule of replacement**^[1] is a transformation rule that may be applied to only a particular segment of an expression. A logical system may be constructed so that it uses either axioms, rules of inference, or both as transformation rules for logical expressions in the system. Whereas a rule of inference is always applied to a whole logical expression, a rule of replacement may be applied to only a particular segment. Within the context of a logical proof, logically equivalent expressions may replace each other. Rules of replacement are used in propositional logic to manipulate propositions.

Common rules of replacement include de Morgan's laws, commutativity, associativity, distribution, double negation,^[2] transposition, material implication, material equivalence, exportation, and tautology. Wikipedia:Please clarify.

References

- [1] Moore and Parker
- [2] not admitted in intuitionistic logic

Second-order propositional logic

A **second-order propositional logic** is a propositional logic extended with quantification over propositions. A special case are the logics that allow **second-order Boolean propositions**, where quantifiers may range either just over the Boolean truth values, or over the Boolean-valued truth functions.

The most widely known formalism is the intuitionistic logic with impredicative quantification, system F. Parigot (1997) showed how this calculus can be extended to admit classical logic.

References

- Parigot, Michel (1997). Proofs of strong normalisation for second order classical natural deduction. *Journal of Symbolic Logic* 62(4):1461–1479.

Substitution (logic)

Substitution is a fundamental concept in logic. A **substitution** is a syntactic transformation on formal expressions. To **apply** a substitution to an expression means to consistently replace its variable, or placeholder, symbols by other expressions. The result expression is called a **substitution instance** of the original expression.

Propositional logic

Definition

Where Ψ and Φ represent formulas of propositional logic, Ψ is a **substitution instance** of Φ if and only if Ψ may be obtained from Φ by substituting formulas for symbols in Φ , always replacing an occurrence of the same symbol by an occurrence of the same formula. For example:

$$(R \rightarrow S) \ \& \ (T \rightarrow S)$$

is a substitution instance of:

$$P \ \& \ Q$$

and

$$(A \leftrightarrow A) \leftrightarrow (A \leftrightarrow A)$$

is a substitution instance of:

$$(A \leftrightarrow A)$$

In some deduction systems for propositional logic, a new expression (a proposition) may be entered on a line of a derivation if it is a substitution instance of a previous line of the derivation (Hunter 1971, p. 118). This is how new lines are introduced in some axiomatic systems. In systems that use rules of transformation, a rule may include the use of a *substitution instance* for the purpose of introducing certain variables into a derivation.

In first-order logic, every closed propositional formula that can be derived from an open propositional formula a by substitution is said to be a substitution instance of a . If a is a closed propositional formula we count a itself as its only substitution instance.

Tautologies

A propositional formula is a tautology if it is true under every valuation (or interpretation) of its predicate symbols. If Φ is a tautology, and Θ is a substitution instance of Φ , then Θ is again a tautology. This fact implies the soundness of the deduction rule described in the previous section.

First-order logic

In first-order logic, a **substitution** is a total mapping $\sigma: V \rightarrow T$ from variables to terms; the notation $\{ x_1 \mapsto t_1, \dots, x_k \mapsto t_k \}$ ^[1] refers to a substitution mapping each variable x_i to the corresponding term t_i , for $i=1,\dots,k$, and every other variable to itself; the x_i must be pairwise distinct. **Applying** that substitution to a term t is written in postfix notation as $t \{ x_1 \mapsto t_1, \dots, x_k \mapsto t_k \}$; it means to (simultaneously) replace every occurrence of each x_i in t by t_i .^[2] The result $t\sigma$ of applying a substitution σ to a term t is called an **instance** of that term t . For example, applying the substitution $\{ x \mapsto z, z \mapsto h(a,y) \}$ to the term

$$\begin{aligned} & f(z \quad .a,g(\quad x \quad),y) \text{ yields} \\ & f(h(a,y) \quad .a,g(\quad z \quad),y) \quad . \end{aligned}$$

The **domain** $dom(\sigma)$ of a substitution σ is commonly defined as the set of variables actually replaced, i.e. $dom(\sigma) = \{ x \in V \mid x\sigma \neq x \}$. A substitution is called a **ground** substitution if it maps all variables of its domain to ground, i.e. variable-free, terms. The substitution instance $t\sigma$ of a ground substitution is a ground term if all of t 's variables are in σ 's domain, i.e. if $vars(t) \subseteq dom(\sigma)$. A substitution σ is called a **linear** substitution if $t\sigma$ is a linear term for some (and hence every) term t containing just the variables of σ 's domain, i.e. with $vars(t) = dom(\sigma)$. A substitution σ is called a **flat** substitution if $x\sigma$ is a variable for every variable x . A substitution σ is called a **renaming** substitution, if it is a permutation on the set of all variables. Like every permutation, a renaming substitution σ always has an **inverse** substitution σ^{-1} , such that $t\sigma\sigma^{-1} = t = \sigma^{-1}\sigma$ for every term t . However, it is not possible to define an inverse for an arbitrary substitution.

For example, $\{ x \mapsto 2, y \mapsto 3+4 \}$ is a ground substitution, $\{ x \mapsto x_1, y \mapsto y_2+4 \}$ is non-ground and non-flat, but linear, $\{ x \mapsto y_2, y \mapsto y_2+4 \}$ is non-linear and non-flat, $\{ x \mapsto y_2, y \mapsto y_2 \}$ is flat, but non-linear, $\{ x \mapsto x_1, y \mapsto y_2 \}$ is both linear and flat, but not a renaming, since it maps both y and y_2 to y_2 ; each of these substitutions has the set $\{x,y\}$ as its domain. An example for a renaming substitution is $\{ x \mapsto x_1, x_1 \mapsto y, y \mapsto y_2, y_2 \mapsto x \}$, it has the inverse $\{ x \mapsto y_2, y_2 \mapsto y, y \mapsto x_1, x_1 \mapsto x \}$. The flat substitution $\{ x \mapsto z, y \mapsto z \}$ cannot have an inverse, since e.g. $(x+y) \{ x \mapsto z, y \mapsto z \} = z+z$, and the latter term cannot be transformed back to $x+y$, as the information about the origin $a z$ stems from is lost. The linear substitution $\{ x \mapsto 2 \}$ cannot have an inverse due to a similar loss of origin information e.g. in $(x+2) \{ x \mapsto 2 \} = 2+2$, even if replacing constants by variables was allowed by some fictitious kind of "generalized substitutions".

Two substitutions are considered **equal** if they map each variable to structurally equal result terms, formally: $\sigma = \tau$ if $x\sigma = x\tau$ for each variable $x \in V$. The **composition** of two substitutions $\sigma = \{ x_1 \mapsto t_1, \dots, x_k \mapsto t_k \}$ and $\tau = \{ y_1 \mapsto u_1, \dots, y_l \mapsto u_l \}$ is obtained by removing from the substitution $\{ x_1 \mapsto t_1\tau, \dots, x_k \mapsto t_k\tau, y_1 \mapsto u_1, \dots, y_l \mapsto u_l \}$ those pairs $y_i \mapsto u_i$ for which $y_i \in \{ x_1, \dots, x_k \}$. The composition of σ and τ is denoted by $\sigma\tau$. Composition is an associative operation, and is compatible with substitution application, i.e. $(\rho\sigma)\tau = \rho(\sigma\tau)$, and $(t\sigma)\tau = t(\sigma\tau)$, respectively, for every substitutions ρ , σ , τ , and every term t . A substitution σ is called **idempotent** if $\sigma\sigma = \sigma$, and hence $t\sigma\sigma = t\sigma$ for every term t . The substitution $\{ x_1 \mapsto t_1, \dots, x_k \mapsto t_k \}$ is idempotent if and only if none of the variables x_i occurs in any t_i . Substitution composition is not commutative, that is, $\sigma\tau$ may be different from $\tau\sigma$, even if σ and τ are idempotent.^[3]
^[4]

For example, $\{ x \mapsto 2, y \mapsto 3+4 \}$ is equal to $\{ y \mapsto 3+4, x \mapsto 2 \}$, but different from $\{ x \mapsto 2, y \mapsto 7 \}$. The substitution $\{ x \mapsto y+y \}$ is idempotent, e.g. $((x+y) \{ x \mapsto y+y \}) \{ x \mapsto y+y \} = ((y+y)+y) \{ x \mapsto y+y \} = (y+y)+y$, while the substitution $\{ x \mapsto x+y \}$ is non-idempotent, e.g. $((x+y) \{ x \mapsto x+y \}) \{ x \mapsto x+y \} = ((x+y)+y) \{ x \mapsto x+y \} = (((x+y)+y)+y)+y$. An

example for non-commuting substitutions is $\{x \mapsto y\} \{y \mapsto z\} = \{x \mapsto z, y \mapsto z\}$, but $\{y \mapsto z\} \{x \mapsto y\} = \{x \mapsto y, y \mapsto z\}$.

References

- Hunter, G. (1971). *Metalogic: An Introduction to the Metatheory of Standard First Order Logic*. University of California Press. ISBN 0-520-01822-2
- Kleene, S. C. (1967). *Mathematical Logic*. Reprinted 2002, Dover. ISBN 0-486-42533-9

[1] some authors use $[t_1/x_1, \dots, t_k/x_k]$ to denote that substitution, e.g., here: p.682;

[2] From a term algebra point of view, the set T of terms is the free term algebra over the set V of variables, hence for each substitution mapping $\sigma: V \rightarrow T$ there is a unique homomorphism $\hat{\sigma}: T \rightarrow T$ that agrees with σ on $V \subseteq T$; the above-defined application of σ to a term t is then viewed as applying the function to the argument t .

[3] ; here: p.73-74

[4] ; here: p.445-446

Syncategorematic term

In scholastic logic, a **syncategorematic term** (*syncategorema*) is a word that cannot serve as the subject or the predicate of a proposition, and thus cannot stand for any of Aristotle's categories, but can be used with other terms to form a proposition. Words such as 'all', 'and', 'if' are examples of such terms.^[1]

The distinction between categorematic and syncategorematic terms was established in ancient Greek grammar. Words that designate self-sufficient entities (i.e., nouns or adjectives) were called categorematic, and those that do not stand by themselves were dubbed syncategorematic, (i.e., prepositions, logical connectives, etc.). Priscian in his *Institutiones grammaticae*^[2] translates the word as *consignificantia*. Scholastics retained the difference, which became a dissertable topic after the 13th century revival of logic. William of Sherwood, a representative of terminism, wrote a treatise called *Syncategoremata*. Later his pupil, Peter of Spain, produced a similar work entitled *Syncategoreumata*.^[3]

In propositional calculus, a **syncategorematic term** is a term that has no individual meaning (a term with an individual meaning is called categorematic). Whether a term is syncategorematic or not is determined by the way it is defined or introduced in the language.

In the common definition of propositional logic, examples of syncategorematic terms are the logical connectives. Let us take the connective \wedge for instance, its semantic rule is:

$$\|\phi \wedge \psi\| = 1 \text{ iff } \|\phi\| = \|\psi\| = 1$$

So its meaning is defined when it occurs in combination with two formulas ϕ and ψ . But it has no meaning when taken in isolation, i.e. $\|\wedge\|$ is not defined.

We could however define the \wedge in a different manner, e.g., using λ -abstraction: $(\lambda b.(\lambda v.b(v)(b)))$, which expects a pair of Boolean-valued arguments, i.e., arguments that are either *TRUE* or *FALSE*, defined as $(\lambda x.(\lambda y.x))$ and $(\lambda x.(\lambda y.y))$ respectively. This is an expression of type $\langle\langle t, t \rangle, t \rangle$. Its meaning is thus a binary function from pairs of entities of type truth-value to an entity of type truth-value. Under this definition it would be non-syncategorematic, or categorematic. Note that while this definition would formally define the \wedge function, it requires the use of λ -abstraction, in which case the λ itself is introduced syncategorematically, thus simply moving the issue up another level of abstraction.

Notes

- [1] Grant, p. 120.
- [2] Priscian, *Institutiones grammaticae*, II, 15
- [3] Peter of Spain (<http://plato.stanford.edu/entries/peter-spain/#4>), *Stanford Encyclopedia of Philosophy* online

References

- Grant, Edward, *God and Reason in the Middle Ages*, Cambridge University Press (July 30, 2001), ISBN 978-0-521-00337-7.

System L

System L is a natural deductive logic developed by E.J. Lemmon. Derived from Suppes' method, it represents natural deduction proofs as sequences of justified steps.

Description of the deductive system

The syntax of proof is governed by nine primitive rules:

1. The Rule of Assumption (A)
2. Modus Ponendo Ponens (MPP)
3. The Rule of Double Negation (DN)
4. The Rule of Conditional Proof (CP)
5. The Rule of \wedge -introduction ($\wedge I$)
6. The Rule of \wedge -elimination ($\wedge E$)
7. The Rule of \vee -introduction ($\vee I$)
8. The Rule of \vee -elimination ($\vee E$)
9. Reductio Ad Absurdum (RAA)

In system L, a proof has a definition with the following conditions:

1. has a finite sequence of well-formed formulas (or *wffs*)
2. each line of it is justified by a rule of the system L
3. the last line of the proof is what is intended, and this last line of the proof uses only the premises which were given, if any.

If no premise is given, the sequent is called theorem. Therefore, the definition of a theorem in system L is:

- a theorem is a sequent that can be proved in system L, using an empty set of assumptions.

Examples

An example of the proof of a sequent (Modus Tollendo Tollens in this case):

$p \rightarrow q, \neg q \vdash \neg p$ [Modus Tollendo Tollens (MTT)]			
Assumption number	Line number	Formula (wff)	Lines in-use and Justification
1	(1)	$(p \rightarrow q)$	A
2	(2)	$\neg q$	A
3	(3)	p	A (for RAA)
1,3	(4)	q	1,3,MPP
1,2,3	(5)	$q \wedge \neg q$	2,4, $\wedge I$
1,2	(6)	$\neg p$	3,5,RAA
Q.E.D			

An example of the proof of a sequent (a theorem in this case):

$\vdash p \vee \neg p$			
Assumption number	Line number	Formula (wff)	Lines in-use and Justification
1	(1)	$\neg(p \vee \neg p)$	A (for RAA)
2	(2)	p	A (for RAA)
2	(3)	$(p \vee \neg p)$	2, $\vee I$
1, 2	(4)	$(p \vee \neg p) \wedge \neg(p \vee \neg p)$	1, 2, $\wedge I$
1	(5)	$\neg p$	2, 4, RAA
1	(6)	$(p \vee \neg p)$	5, $\vee I$
1	(7)	$(p \vee \neg p) \wedge \neg(p \vee \neg p)$	1, 6, $\wedge I$
	(8)	$\neg\neg(p \vee \neg p)$	1, 7, RAA
	(9)	$(p \vee \neg p)$	8, DN
Q.E.D			

Each rule of system L has its own requirements for the type of input(s) or entry(es) that it can accept and has its own way of treating and calculating the assumptions used by its inputs.

External links

- Pelletier, Jeff, "A History of Natural Deduction and Elementary Logic Textbooks. ^[1]"

References

[1] <http://www.sfu.ca/~jeffpell/papers/pelletierNDtexts.pdf>

Unsatisfiable core

In mathematical logic, given an unsatisfiable boolean propositional formula in conjunctive normal form, a subset of clauses whose conjunction is still unsatisfiable is called an **unsatisfiable core** of the original formula.

Many SAT solvers can produce a *resolution graph* which proves the unsatisfiability of the original problem. This can be analyzed to produce a smaller unsatisfiable core.

An unsatisfiable core is called a *minimal unsatisfiable core*, if every proper subset (allowing removal of any arbitrary clause or clauses) of it is satisfiable. Thus, such a core is a local minimum, though not necessarily a global one. There are several practical methods of computing minimal unsatisfiable cores.^{[1][2]}

A *minimum unsatisfiable core* contains the smallest number of the original clauses required to still be unsatisfiable. No practical algorithms for computing the minimum core are known. Algorithms for Computing Minimal Unsatisfiable Subsets^[3]. Notice the terminology: whereas *minimal unsatisfiable core* was a local problem with an easy solution, the *minimum unsatisfiable core* is a global problem with no known easy solution.

References

- [1] N. Dershowitz, Z. Hanna, and A. Nadel, A Scalable Algorithm for Minimal Unsatisfiable Core Extraction (http://www.cs.tau.ac.il/~ale1/muc_sat06_short_8.pdf)
- [2] Stefan Szeider, Minimal unsatisfiable formulas with bounded clause-variable difference are fixed-parameter tractable (http://www.sciencedirect.com/science?_ob=ArticleURL&_udi=B6WJ0-4D1DD82-1&_user=121711&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000009978&_version=1&_urlVersion=0&_userid=121711&md5=13f75e9221d3d5e1a533420649a65093)
- [3] http://www.iwu.edu/~mliffito/publications/jar_liffiton_CAMUS.pdf

Zeroth-order logic

Zeroth-order logic is first-order logic without quantifiers. A finitely axiomatizable zeroth-order logic is isomorphic to a propositional logic. Zeroth-order logic with axiom schema is a more expressive system than propositional logic. An example is given by the system Primitive recursive arithmetic, or PRA.

Example

The well-known syllogism

- All men are mortal
- Socrates is a man
- Therefore, Socrates is mortal

cannot be formalized in propositional logic, because of the use of predicates like "is a man" and "is mortal". The obvious formalization in first-order logic uses universal quantification to model the use of "All".

The following weak version of the syllogism can be formalized in propositional logic:

- If Socrates is a man, then Socrates is mortal
- Socrates is a man
- Therefore, Socrates is mortal

This can be done by introducing propositional constants *SMN* (for "Socrates is a man") and *SML* (for "Socrates is mortal"), and the two axioms

- $SMN \rightarrow SML$, and
- SMN .

Together with the usual rule of modus ponens the conclusion, *SML*, follows.

In this weak version most of the essence of the original syllogism has been lost. In predicate logic one can instead introduce predicates *Man* (for "is a man"), *Mortal* (for "is mortal"), constants *A* (for "Aristotle"), *S* (for "Socrates"), *Z* (for "Zeus"), and so on, and use a multitude of axioms, one for each individual:

- $\text{Man}(A) \rightarrow \text{Mortal}(A)$
- $\text{Man}(S) \rightarrow \text{Mortal}(S)$
- $\text{Man}(Z) \rightarrow \text{Mortal}(Z)$
- ...
- $\text{Man}(S)$
- $\neg\text{Mortal}(Z)$

Again, modus ponens allows to conclude $\text{Mortal}(S)$. If the axioms for contraposition are added, also $\neg\text{Man}(Z)$ becomes a theorem.

By using an axiom schema, the above can be collapsed into:

- $\text{Man}(x) \rightarrow \text{Mortal}(x)$
- $\text{Man}(S)$
- $\neg\text{Mortal}(Z)$

The first line uses the variable *x*, which can be instantiated by any constant for an individual, such as *S*. The axioms are then the substitution instances of the schema.

An equivalent approach is to declare the schema to be a plain axiom and to make variable substitution a special inference rule of the logic.

Relation to general first-order logic

At first glance it might appear that by using axiom schemata as in the example any first-order logic can be made zeroth-order. However, in general only universal quantifiers at the outermost level can be eliminated this way.

Propositional Fallacies

Affirming a disjunct

The formal fallacy of **affirming a disjunct** also known as **the fallacy of the alternative disjunct** or a **false exclusionary disjunct** occurs when a deductive argument takes the following logical form:

A or B

A

Therefore, it is not the case that B

Or in logical operators:

$p \vee q$

p

$\vdash \neg q$

Where \vdash denotes a logical assertion.

Explanation

The fallacy lies in concluding that one disjunct must be false because the other disjunct is true; in fact they may both be true because "or" is defined inclusively rather than exclusively. It is a fallacy of equivocation between the operations OR and XOR.

Affirming the disjunct should not be confused with the valid argument known as the disjunctive syllogism.

Example

The following argument indicates the invalidity of affirming a disjunct:

Max is a cat or Max is a mammal.

Max is a cat.

Therefore, Max is not a mammal.

This inference is invalid. If Max is a cat then Max is also a mammal. (Remember "or" is defined in an inclusive sense not an exclusive sense.)

The car is red or the car is large.

The car is red.

Therefore, the car is not large.

The above example of an argument also demonstrates the fallacy.

External links

- Fallacy files: affirming a disjunct [1]

References

[1] <http://www.fallacyfiles.org/afonedis.html>

Affirming the consequent

Affirming the consequent, sometimes called **converse error** or **fallacy of the converse**, is a formal fallacy of inferring the converse from the original statement. The corresponding argument has the general form:

1. If P , then Q .
2. Q .
3. Therefore, P .

An argument of this form is invalid, i.e., the conclusion can be false even when statements 1 and 2 are true. Since P was never asserted as the *only* sufficient condition for Q , other factors could account for Q (while P was false).

To put it differently, if P implies Q , the **only** inference that can be made is $\text{non-}Q$ implies $\text{non-}P$. ($\text{Non-}P$ and $\text{non-}Q$ designate the opposite propositions to P and Q .) Symbolically:

$$(P \Rightarrow Q) \Leftrightarrow (\text{non-}Q \Rightarrow \text{non-}P)$$

The name *affirming the consequent* derives from the premise Q , which affirms the "then" clause of the conditional premise.

Examples

One way to demonstrate the invalidity of this argument form is with a counterexample with true premises but an obviously false conclusion. For example:

If Bill Gates owns Fort Knox, then he is rich.

Bill Gates is rich.

Therefore, Bill Gates owns Fort Knox.

Owning Fort Knox is not the *only* way to be rich. Any number of other ways exist to be rich.

However, one can affirm with certainty that "if Bill Gates is not rich" ($\text{non-}Q$) then "Bill Gates does not own Fort Knox" ($\text{non-}P$).

Arguments of the same form can sometimes seem superficially convincing, as in the following example:

If I have the flu, then I have a sore throat.

I have a sore throat.

Therefore, I have the flu.

But having the flu is not the *only* cause of a sore throat since many illnesses cause sore throat, such as the common cold or strep throat.

References

Denying the antecedent

Denying the antecedent, sometimes also called **inverse error** or **fallacy of the inverse**, is a formal fallacy of inferring the inverse from the original statement. It is committed by reasoning in the form:

If P , then Q .

Not P .

Therefore, not Q .

Arguments of this form are invalid. Informally, this means that arguments of this form do not give good reason to establish their conclusions, even if their premises are true.

The name *denying the antecedent* derives from the premise "not P ", which denies the "if" clause of the conditional premise.

One way to demonstrate the invalidity of this argument form is with a counterexample with true premises but an obviously false conclusion. For example:

If it rains, then the grass gets wet.

It is not raining.

Therefore, the grass is not wet.

The argument is invalid because there are other reasons for which the grass could be wet (being sprayed with water by a hose, for example). Another example:

If Queen Elizabeth is an American citizen, then she is a human being.

Queen Elizabeth is not an American citizen.

Therefore, Queen Elizabeth is not a human being.

That argument is obviously bad, but arguments of the same form can sometimes seem superficially convincing, as in the following example offered, with apologies for its lack of logical rigour, by Alan Turing in the article "Computing Machinery and Intelligence":

If each man had a definite set of rules of conduct by which he regulated his life he would be no better than a machine. But there are no such rules, so men cannot be machines.

However, men could still be machines that do not follow a definite set of rules. Thus this argument (as Turing intends) is invalid.

It is possible that an argument that denies the antecedent could be valid, if the argument instantiates some other valid form. For example, if the claims P and Q express the same proposition, then the argument would be trivially valid, as it would beg the question. In everyday discourse, however, such cases are rare, typically only occurring when the "if-then" premise is actually an "if and only if" claim (i.e., a biconditional/equality). For example:

If I am President of the United States, then I can veto Congress.

I am not President.

Therefore, I cannot veto Congress.

The above argument is not valid, but would be if the first premise ended thus: "...and if I can veto Congress, then I am the U.S. President" (as is in fact true). More to the point, the validity of the new argument stems not from denying the antecedent, but modus tollens (denying the consequent).

References

External links

- FallacyFiles.org: Denying the Antecedent (<http://www.fallacyfiles.org/denyante.html>)
- safalra.com: Denying The Antecedent (<http://www.safalra.com/philosophy/fallacies/antecedent/>)

Rules of Inference

List of rules of inference

This is a list of rules of inference, logical laws that relate to mathematical formulae.

Introduction

Rules of inference are syntactical **transform** rules which one can use to infer a conclusion from a premise to create an argument. A set of rules can be used to infer any valid conclusion if it is complete, while never inferring an invalid conclusion, if it is sound. A sound and complete set of rules need not include every rule in the following list, as many of the rules are redundant, and can be proven with the other rules.

Discharge rules permit inference from a subderivation based on a temporary assumption. Below, the notation

$$\varphi \vdash \psi$$

indicates such a subderivation from the temporary assumption φ to ψ .

Rules for classical sentential calculus

Sentential calculus is also known as propositional calculus.

Rules for negations

Reductio ad absurdum (or *Negation Introduction*)

$$\begin{array}{c} \varphi \vdash \psi \\ \varphi \vdash \neg\psi \\ \hline \neg\varphi \end{array}$$

Reductio ad absurdum (related to the law of excluded middle)

$$\begin{array}{c} \neg\varphi \vdash \psi \\ \neg\varphi \vdash \neg\psi \\ \hline \varphi \end{array}$$

Noncontradiction (or *Negation Elimination*)

$$\begin{array}{c} \varphi \\ \neg\varphi \\ \hline \psi \end{array}$$

Double negation elimination

$$\begin{array}{c} \neg\neg\varphi \\ \hline \varphi \end{array}$$

Double negation introduction

$$\begin{array}{c} \varphi \\ \hline \neg\neg\varphi \end{array}$$

Rules for conditionals

Deduction theorem (or *Conditional Introduction*)

$$\frac{\varphi \vdash \psi}{\varphi \rightarrow \psi}$$

Modus ponens (or *Conditional Elimination*)

$$\frac{\varphi \rightarrow \psi}{\frac{\varphi}{\psi}}$$

Modus tollens

$$\frac{\varphi \rightarrow \psi}{\frac{\neg \psi}{\neg \varphi}}$$

Rules for conjunctions

Adjunction (or *Conjunction Introduction*)

$$\frac{\varphi}{\frac{\psi}{\varphi \wedge \psi}}$$

Simplification (or *Conjunction Elimination*)

$$\frac{\varphi \wedge \psi}{\frac{\varphi}{\frac{\varphi \wedge \psi}{\psi}}}$$

Rules for disjunctions

Addition (or *Disjunction Introduction*)

$$\frac{\varphi}{\frac{\psi}{\varphi \vee \psi}}$$

Case analysis

$$\frac{\varphi \vee \psi}{\frac{\varphi \rightarrow \chi}{\frac{\psi \rightarrow \chi}{\chi}}}$$

Disjunctive syllogism

$$\frac{\varphi \vee \psi}{\frac{\neg \varphi}{\frac{\psi}{\varphi \vee \psi}}}$$

$$\frac{\neg\psi}{\varphi}$$

Rules for biconditionals

Biconditional introduction

$$\frac{\varphi \rightarrow \psi}{\frac{\psi \rightarrow \varphi}{\varphi \leftrightarrow \psi}}$$

Biconditional Elimination

$$\begin{aligned} & \varphi \leftrightarrow \psi \\ & \frac{\varphi}{\psi} \\ & \psi \\ & \varphi \leftrightarrow \psi \\ & \frac{\psi}{\varphi} \\ & \varphi \leftrightarrow \psi \\ & \frac{\neg\varphi}{\neg\psi} \\ & \neg\psi \\ & \varphi \leftrightarrow \psi \\ & \frac{\neg\psi}{\neg\varphi} \\ & \varphi \leftrightarrow \psi \\ & \frac{\psi \vee \varphi}{\psi \wedge \varphi} \\ & \psi \wedge \varphi \\ & \varphi \leftrightarrow \psi \\ & \frac{\neg\psi \vee \neg\varphi}{\neg\psi \wedge \neg\varphi} \end{aligned}$$

Rules of classical predicate calculus

In the following rules, $\varphi(\beta/\alpha)$ is exactly like φ except for having the term β everywhere φ has the free variable α .

Universal Introduction (or *Universal Generalization*)

$$\frac{\varphi(\beta/\alpha)}{\forall\alpha \varphi}$$

Restriction 1: β does not occur in φ .

Restriction 2: β is not mentioned in any hypothesis or undischarged assumptions.

Universal Elimination (or *Universal Instantiation*)

$$\frac{\forall\alpha \varphi}{\varphi(\beta/\alpha)}$$

Restriction: No free occurrence of α in φ falls within the scope of a quantifier quantifying a variable occurring in β .

Existential Introduction (or *Existential Generalization*)

$$\frac{\varphi(\beta/\alpha)}{\exists\alpha \varphi}$$

Restriction: No free occurrence of α in φ falls within the scope of a quantifier quantifying a variable occurring in β .

Existential Elimination (or *Existential Instantiation*)

$$\frac{\exists\alpha \varphi \quad \varphi(\beta/\alpha) \vdash \psi}{\psi}$$

Restriction 1: No free occurrence of α in φ falls within the scope of a quantifier quantifying a variable occurring in β .

Restriction 2: There is no occurrence, free or bound, of β in ψ .

Table: Rules of Inference - a short summary

The rules above can be summed up in the following table.^[1] The "Tautology" column shows how to interpret the notation of a given rule.

Rule of inference	Tautology	Name
$\frac{p}{\therefore p \vee q}$	$p \rightarrow (p \vee q)$	Addition
$\frac{p \wedge q}{\therefore p}$	$(p \wedge q) \rightarrow p$	Simplification
$\frac{p \\ q}{\therefore p \wedge q}$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
$\frac{p \\ p \rightarrow q}{\therefore q}$	$((p \wedge (p \rightarrow q)) \rightarrow q$	Modus ponens
$\frac{\neg q \\ p \rightarrow q}{\therefore \neg p}$	$((\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus tollens
$\frac{p \rightarrow q \\ q \rightarrow r}{\therefore p \rightarrow r}$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$\frac{p \vee q \\ \neg p}{\therefore q}$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive syllogism
$\frac{p \vee q \\ \neg p \vee r}{\therefore q \vee r}$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution

All rules use the basic logic operators. A complete table of "logic operators" is shown by a truth table, giving definitions of all the possible (16) truth functions of 2 boolean variables (p, q):

p	q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T
T	F	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T
F	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T
F	F	F	T	F	T	F	T	F	T	F	T	F	T	F	F	T	T

where T = true and F = false, and, the columns are the logical operators: **0**, false, Contradiction; **1**, NOR, Logical NOR; **2**, Converse nonimplication; **3**, $\neg p$, Negation; **4**, Material nonimplication; **5**, $\neg q$, Negation; **6**, XOR, Exclusive disjunction; **7**, NAND, Logical NAND; **8**, AND, Logical conjunction; **9**, XNOR, If and only if, Logical biconditional; **10**, q , Projection function; **11**, if/then, Logical implication; **12**, p , Projection function; **13**, then/if, Converse implication; **14**, OR, Logical disjunction; **15**, true, Tautology.

Each logic operator can be used in an assertion about variables and operations, showing a basic rule of inference. Examples:

- The column-14 operator (OR), shows *Addition rule*: when $p=T$ (the hypothesis selects the first two lines of the table), we see (at column-14) that $p \vee q=T$.

We can see also that, with the same premise, another conclusions are valid: columns 12, 14 and 15 are T.

- The column-8 operator (AND), shows *Simplification rule*: when $p \wedge q=T$ (first line of the table), we see that $p=T$.

With this premise, we also conclude that $q=T$, $p \vee q=T$, etc. as showed by columns 9-15.

- The column-11 operator (IF/THEN), shows *Modus ponens rule*: when $p \rightarrow q=T$ and $p=T$ only one line of the truth table (the first) satisfies these two conditions. On this line, q is also true. Therefore, whenever $p \rightarrow q$ is true and p is true, q must also be true.

Machines and well-trained people use this look at table approach to do basic inferences, and to check if other inferences (for the same premises) can be obtained.

Example 1

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

$$\begin{aligned} p &\rightarrow q \\ q &\rightarrow r \\ \therefore \overline{p} &\rightarrow r \end{aligned}$$

Example 2

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

Step	Reason
1. $\neg p \wedge q$	Hypothesis
2. $\neg p$	Simplification using Step 1
3. $r \rightarrow p$	Hypothesis
4. $\neg r$	Modus tollens using Step 2 and 3
5. $\neg r \rightarrow s$	Hypothesis
6. s	Modus ponens using Step 4 and 5
7. $s \rightarrow t$	Hypothesis
8. t	Modus ponens using Step 6 and 7

References

[1] Kenneth H. Rosen: *Discrete Mathematics and its Applications*, Fifth Edition, p. 58.

Absorption (logic)

Transformation rules
Propositional calculus
Rules of inference <ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement <ul style="list-style-type: none"> • 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

Absorption is a valid argument form and rule of inference of propositional logic.^[1] The rule states that if P implies Q , then P implies P and Q . The rule makes it possible to introduce conjunctions to proofs. It is called the law of absorption because the term Q is "absorbed" by the term P in the consequent.^[2] The rule can be stated:

$$\frac{P \rightarrow Q}{\therefore P \rightarrow (P \wedge Q)}$$

where the rule is that wherever an instance of " $P \rightarrow Q$ " appears on a line of a proof, " $P \rightarrow (P \wedge Q)$ " can be placed on a subsequent line.

Formal notation

The *absorption* rule may be expressed as a sequent:

$$P \rightarrow Q \vdash P \rightarrow (P \wedge Q)$$

where \vdash is a metalogical symbol meaning that $P \rightarrow (P \wedge Q)$ is a syntactic consequence of $(P \leftrightarrow Q)$ in some logical system;

and expressed as a truth-functional tautology or theorem of propositional logic. The principle was stated as a theorem of propositional logic by Russell and Whitehead in *Principia Mathematica* as:

$$(P \rightarrow Q) \leftrightarrow (P \rightarrow (P \wedge Q))$$

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

Examples

If it will rain, then I will wear my coat.

Therefore, if it will rain then it will rain and I will wear my coat.

Proof by truth table

P	Q	$P \rightarrow Q$	$P \rightarrow P \wedge Q$
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	T

Formal proof

<i>Proposition</i>	<i>Derivation</i>
$P \rightarrow Q$	Given
$\neg P \vee Q$	Material implication
$\neg P \vee P$	Law of Excluded Middle
$(\neg P \vee P) \wedge (\neg P \vee Q)$	Conjunction
$\neg P \vee (P \wedge Q)$	Reverse Distribution
$P \rightarrow (P \wedge Q)$	Material implication

References

- [1] <http://www.philosophypages.com/lg/e11a.htm>
[2] Russell and Whitehead, *Principia Mathematica*

Admissible rule

In logic, a rule of inference is **admissible** in a formal system if the set of theorems of the system does not change when that rule is added to the existing rules of the system. In other words, every formula that can be derived using that rule is already derivable without that rule, so, in a sense, it is redundant. The concept of an admissible rule was introduced by Paul Lorenzen (1955).

Definitions

Admissibility has been systematically studied only in the case of structural rules in propositional non-classical logics, which we will describe next.

Let a set of basic propositional connectives be fixed (for instance, $\{\rightarrow, \wedge, \vee, \perp\}$ in the case of superintuitionistic logics, or $\{\rightarrow, \perp, \Box\}$ in the case of monomodal logics). Well-formed formulas are built freely using these connectives from a countably infinite set of propositional variables p_n . A substitution σ is a function from formulas to formulas which commutes with the connectives, i.e.,

$$\sigma f(A_1, \dots, A_n) = f(\sigma A_1, \dots, \sigma A_n)$$

for every connective f , and formulas A_1, \dots, A_n . (We may also apply substitutions to sets Γ of formulas, making $\sigma\Gamma = \{\sigma A : A \in \Gamma\}$.) A Tarski-style consequence relation^[1] is a relation \vdash between sets of formulas, and formulas, such that

1. $A \vdash A$,
2. if $\Gamma \vdash A$ then $\Gamma, \Delta \vdash A$,
3. if $\Gamma \vdash A$ and $\Delta, A \vdash B$ then $\Gamma, \Delta \vdash B$,

for all formulas A, B , and sets of formulas Γ, Δ . A consequence relation such that

1. if $\Gamma \vdash A$ then $\sigma\Gamma \vdash \sigma A$

for all substitutions σ is called **structural**. (Note that the term "structural" as used here and below is unrelated to the notion of structural rules in sequent calculi.) A structural consequence relation is called a **propositional logic**. A formula A is a theorem of a logic \vdash if $\emptyset \vdash A$.

For example, we identify a superintuitionistic logic L with its standard consequence relation \vdash_L axiomatizable by modus ponens and axioms, and we identify a normal modal logic with its global consequence relation \vdash_L axiomatized by modus ponens, necessitation, and axioms.

A **structural inference rule**^[2] (or just **rule** for short) is given by a pair (Γ, B) , usually written as

$$\frac{A_1, \dots, A_n}{B} \quad \text{or} \quad A_1, \dots, A_n/B,$$

where $\Gamma = \{A_1, \dots, A_n\}$ is a finite set of formulas, and B is a formula. An **instance** of the rule is

$$\sigma A_1, \dots, \sigma A_n/\sigma B$$

for a substitution σ . The rule Γ/B is **derivable** in \vdash , if $\Gamma \vdash B$. It is **admissible** if for every instance of the rule, σB is a theorem whenever all formulas from $\sigma\Gamma$ are theorems.^[3] In other words, a rule is admissible if, when added to the logic, does not lead to new theorems.^[4] We also write $\Gamma \sim B$ if Γ/B is admissible. (Note that \sim is a structural consequence relation on its own.)

Every derivable rule is admissible, but not vice versa in general. A logic is **structurally complete** if every admissible rule is derivable, i.e., $\vdash = \sim$.^[5]

In logics with a well-behaved conjunction connective (such as superintuitionistic or modal logics), a rule $A_1, \dots, A_n/B$ is equivalent to $A_1 \wedge \dots \wedge A_n/B$ with respect to admissibility and derivability. It is therefore customary to only deal with unary rules A/B .

Examples

- Classical propositional calculus (*CPC*) is structurally complete.^[6] Indeed, assume that A/B is non-derivable rule, and fix an assignment v such that $v(A) = 1$, and $v(B) = 0$. Define a substitution σ such that for every variable p , $\sigma p = \top$ if $v(p) = 1$, and $\sigma p = \perp$ if $v(p) = 0$. Then σA is a theorem, but σB is not (in fact, $\neg\sigma B$ is a theorem). Thus the rule A/B is not admissible either. (The same argument applies to any multi-valued logic L complete with respect to a logical matrix whose all elements have a name in the language of L .)
- The Kreisel–Putnam rule (aka Harrop's rule, or independence of premise rule)

$$(KPR) \quad \frac{\neg p \rightarrow q \vee r}{(\neg p \rightarrow q) \vee (\neg p \rightarrow r)}$$

is admissible in the intuitionistic propositional calculus (*IPC*). In fact, it is admissible in every superintuitionistic logic.^[7] On the other hand, the formula

$$(\neg p \rightarrow q \vee r) \rightarrow (\neg p \rightarrow q) \vee (\neg p \rightarrow r)$$

is not an intuitionistic tautology, hence *KPR* is not derivable in *IPC*. In particular, *IPC* is not structurally complete.

- The rule

$$\frac{\square p}{p}$$

is admissible in many modal logics, such as *K*, *D*, *K4*, *S4*, *GL* (see this table for names of modal logics). It is derivable in *S4*, but it is not derivable in *K*, *D*, *K4*, or *GL*.

- The rule

$$\frac{\Diamond p \wedge \Diamond \neg p}{\perp}$$

is admissible in every normal modal logic.^[8] It is derivable in *GL* and *S4.1*, but it is not derivable in *K*, *D*, *K4*, *S4*, *S5*.

- Löb's rule

$$(LR) \quad \frac{\square p \rightarrow p}{p}$$

is admissible (but not derivable) in the basic modal logic *K*, and it is derivable in *GL*. However, *LR* is not admissible in *K4*. In particular, it is *not* true in general that a rule admissible in a logic *L* must be admissible in its extensions.

- The Gödel–Dummett logic (*LC*), and the modal logic *Grz*.3 are structurally complete.^[9] The product fuzzy logic is also structurally complete.^[10]

Decidability and reduced rules

The basic question about admissible rules of a given logic is whether the set of all admissible rules is decidable. Note that the problem is nontrivial even if the logic itself (i.e., its set of theorems) is decidable: the definition of admissibility of a rule A/B involves an unbounded universal quantifier over all propositional substitutions, hence *a priori* we only know that admissibility of rule in a decidable logic is Π_1^0 (i.e., its complement is recursively enumerable). For instance, it is known that admissibility in the bimodal logics K_u and $K4_u$ (the extensions of K or $K4$ with the universal modality) is undecidable.^[11] Remarkably, decidability of admissibility in the basic modal logic K is a major open problem.

Nevertheless, admissibility of rules is known to be decidable in many modal and superintuitionistic logics. The first decision procedures for admissible rules in basic transitive modal logics were constructed by Rybakov, using the **reduced form of rules**.^[12] A modal rule in variables p_0, \dots, p_k is called reduced if it has the form

$$\frac{\bigvee_{i=0}^n (\bigwedge_{j=0}^k \neg_{i,j}^0 p_j \wedge \bigwedge_{j=0}^k \neg_{i,j}^1 \Box p_j)}{p_0},$$

where each $\neg_{i,j}^u$ is either blank, or negation \neg . For each rule r , we can effectively construct a reduced rule s (called the reduced form of r) such that any logic admits (or derives) r if and only if it admits (or derives) s , by introducing extension variables for all subformulas in A , and expressing the result in the full disjunctive normal form. It is thus sufficient to construct a decision algorithm for admissibility of reduced rules.

Let $\bigvee_{i=0}^n \varphi_i/p_0$ be a reduced rule as above. We identify every conjunction φ_i with the set $\{\neg_{i,j}^0 p_j, \neg_{i,j}^1 \Box p_j \mid j \leq k\}$ of its conjuncts. For any subset W of the set $\{\varphi_i \mid i \leq n\}$ of all conjunctions, let us define a Kripke model $M = \langle W, R, \Vdash \rangle$ by

$$\varphi_i \Vdash p_j \iff p_j \in \varphi_i,$$

$$\varphi_i R \varphi_{i'} \iff \forall j \leq k (\Box p_j \in \varphi_i \Rightarrow \{p_j, \Box p_j\} \subseteq \varphi_{i'}).$$

Then the following provides an algorithmic criterion for admissibility in $K4$:^[13]

Theorem. The rule $\bigvee_{i=0}^n \varphi_i/p_0$ is *not* admissible in $K4$ if and only if there exists a set $W \subseteq \{\varphi_i \mid i \leq n\}$ such that

- $\varphi_i \not\Vdash p_0$ for some $i \leq n$,
- $\varphi_i \Vdash \varphi_i$ for every $i \leq n$,
- for every subset D of W there exist elements $\alpha, \beta \in W$ such that the equivalences

$$\alpha \Vdash \Box p_j \text{ if and only if } \varphi \Vdash p_j \wedge \Box p_j \text{ for every } \varphi \in D$$

$$\alpha \Vdash \Box p_j \text{ if and only if } \alpha \Vdash p_j \text{ and } \varphi \Vdash p_j \wedge \Box p_j \text{ for every } \varphi \in D$$

hold for all j .

Similar criteria can be found for the logics *S4*, *GL*, and *Grz*.^[14] Furthermore, admissibility in intuitionistic logic can be reduced to admissibility in *Grz* using the Gödel–McKinsey–Tarski translation:^[15]

$$A \sim_{IPC} B \text{ if and only if } T(A) \sim_{Grz} T(B).$$

Rybakov (1997) developed much more sophisticated techniques for showing decidability of admissibility, which apply to a robust (infinite) class of transitive (i.e., extending $K4$ or *IPC*) modal and superintuitionistic logics, including e.g. *S4.1*, *S4.2*, *S4.3*, *KC*, T_k (as well as the above mentioned logics *IPC*, *K4*, *S4*, *GL*, *Grz*).^[16]

Despite being decidable, the admissibility problem has relatively high computational complexity, even in simple logics: admissibility of rules in the basic transitive logics *IPC*, *K4*, *S4*, *GL*, *Grz* is coNEXP-complete.^[17] This should be contrasted with the derivability problem (for rules or formulas) in these logics, which is PSPACE-complete.^[18]

Projectivity and unification

Admissibility in propositional logics is closely related to unification in the equational theory of modal or Heyting algebras. The connection was developed by Ghilardi (1999, 2000). In the logical setup, a **unifier** of a formula A in a logic L (an L -unifier for short) is a substitution σ such that σA is a theorem of L . (Using this notion, we can rephrase admissibility of a rule A/B in L as "every L -unifier of A is an L -unifier of B ".) An L -unifier σ is **less general** than an L -unifier τ , written as $\sigma \leq \tau$, if there exists a substitution ν such that

$$\vdash_L \sigma p \leftrightarrow \nu \tau p$$

for every variable p . A **complete set of unifiers** of a formula A is a set S of L -unifiers of A such that every L -unifier of A is less general than some unifier from S . A most general unifier (mgu) of A is a unifier σ such that $\{\sigma\}$ is a complete set of unifiers of A . It follows that if S is a complete set of unifiers of A , then a rule A/B is L -admissible if and only if every σ in S is an L -unifier of B . Thus we can characterize admissible rules if we can find well-behaved complete sets of unifiers.

An important class of formulas which have a most general unifier are the **projective formulas**: these are formulas A such that there exists a unifier σ of A such that

$$A \vdash_L B \leftrightarrow \sigma B$$

for every formula B . Note that σ is a mgu of A . In transitive modal and superintuitionistic logics with the finite model property (fmp), one can characterize projective formulas semantically as those whose set of finite L -models has the **extension property**:^[19] if M is a finite Kripke L -model with a root r whose cluster is a singleton, and the formula A holds in all points of M except for r , then we can change the valuation of variables in r so as to make A true in r as well. Moreover, the proof provides an explicit construction of a mgu for a given projective formula A .

In the basic transitive logics IPC , $K4$, $S4$, GL , Grz (and more generally in any transitive logic with the fmp whose set of finite frame satisfies another kind of extension property), we can effectively construct for any formula A its **projective approximation** $\Pi(A)$:^[20] a finite set of projective formulas such that

1. $P \vdash_L A$ for every $P \in \Pi(A)$,
2. every unifier of A is a unifier of a formula from $\Pi(A)$.

It follows that the set of mgus of elements of $\Pi(A)$ is a complete set of unifiers of A . Furthermore, if P is a projective formula, then

$$P \succsim_L B \text{ if and only if } P \vdash_L B$$

for any formula B . Thus we obtain the following effective characterization of admissible rules:^[21]

$$A \succsim_L B \text{ if and only if } \forall P \in \Pi(A) (P \vdash_L B).$$

Bases of admissible rules

Let L be a logic. A set R of L -admissible rule is called a **basis**^[22] of admissible rules, if every admissible rule Γ/B can be derived from R and the derivable rules of L , using substitution, composition, and weakening. In other words, R is a basis if and only if \succsim_L is the smallest structural consequence relation which includes \vdash_L and R .

Notice that decidability of admissible rules of a decidable logic is equivalent to the existence of recursive (or recursively enumerable) bases: on the one hand, the set of *all* admissible rule is a recursive basis if admissibility is decidable. On the other hand, the set of admissible rules is always co-r.e., and if we further have an r.e. basis, it is also r.e., hence it is decidable. (In other words, we can decide admissibility of A/B by the following algorithm: we start in parallel two exhaustive searches, one for a substitution σ which unifies A but not B , and one for a derivation of A/B from R and \vdash_L . One of the searches has to eventually come up with an answer.) Apart from decidability, explicit bases of admissible rules are useful for some applications, e.g. in proof complexity.^[23]

For a given logic, we can ask whether it has a recursive or finite basis of admissible rules, and to provide an explicit basis. If a logic has no finite basis, it can nevertheless has an **independent basis**: a basis R such that no proper subset

of R is a basis.

In general, very little can be said about existence of bases with desirable properties. For example, while tabular logics are generally well-behaved, and always finitely axiomatizable, there exist tabular modal logics without a finite or independent basis of rules.^[24] Finite bases are relatively rare: even the basic transitive logics IPC , $K4$, $S4$, GL , Grz do not have a finite basis of admissible rules,^[25] though they have independent bases.^[26]

Examples of bases

- The empty set is a basis of L -admissible rules if and only if L is structurally complete.
- Every extension of the modal logic $S4.3$ (including, notably, $S5$) has a finite basis consisting of the single rule^[27]

$$\frac{\Diamond p \wedge \Diamond \neg p}{\perp}.$$

- Visser's rules

$$\frac{\left(\bigwedge_{i=1}^n (p_i \rightarrow q_i) \rightarrow p_{n+1} \vee p_{n+2} \right) \vee r}{\bigvee_{j=1}^{n+2} \left(\bigwedge_{i=1}^n (p_i \rightarrow q_i) \rightarrow p_j \right) \vee r}, \quad n \geq 1$$

are a basis of admissible rules in IPC or KC .^[28]

- The rules

$$\frac{\Box \left(\Box q \rightarrow \bigvee_{i=1}^n \Box p_i \right) \vee \Box r}{\bigvee_{i=1}^n \Box (q \wedge \Box q \rightarrow p_i) \vee r}, \quad n \geq 0$$

are a basis of admissible rules of GL .^[29] (Note that the empty disjunction is defined as \perp .)

- The rules

$$\frac{\Box \left(\Box (q \rightarrow \Box q) \rightarrow \bigvee_{i=1}^n \Box p_i \right) \vee \Box r}{\bigvee_{i=1}^n \Box (\Box q \rightarrow p_i) \vee r}, \quad n \geq 0$$

are a basis of admissible rules of $S4$ or Grz .^[30]

Semantics for admissible rules

A rule Γ/B is **valid** in a modal or intuitionistic Kripke frame $F = \langle W, R \rangle$, if the following is true for every valuation \Vdash in F :

if $\forall x \in W (x \Vdash A)$ for all $A \in \Gamma$, then $\forall x \in W (x \Vdash B)$.

(The definition readily generalizes to general frames, if needed.)

Let X be a subset of W , and t a point in W . We say that t is

- a **reflexive tight predecessor** of X , if for every y in W : $t R y$ if and only if $t = y$ or $x = y$ or $x R y$ for some x in X ,
- an **irreflexive tight predecessor** of X , if for every y in W : $t R y$ if and only if $x = y$ or $x R y$ for some x in X .

We say that a frame F has reflexive (irreflexive) tight predecessors, if for every *finite* subset X of W , there exists a reflexive (irreflexive) tight predecessor of X in W .

We have:^[31]

- a rule is admissible in *IPC* if and only if it is valid in all intuitionistic frames which have reflexive tight predecessors,
- a rule is admissible in *K4* if and only if it is valid in all transitive frames which have reflexive and irreflexive tight predecessors,
- a rule is admissible in *S4* if and only if it is valid in all transitive reflexive frames which have reflexive tight predecessors,
- a rule is admissible in *GL* if and only if it is valid in all transitive converse well-founded frames which have irreflexive tight predecessors.

Note that apart from a few trivial cases, frames with tight predecessors must be infinite, hence admissible rules in basic transitive logics do not enjoy the finite model property.

Structural completeness

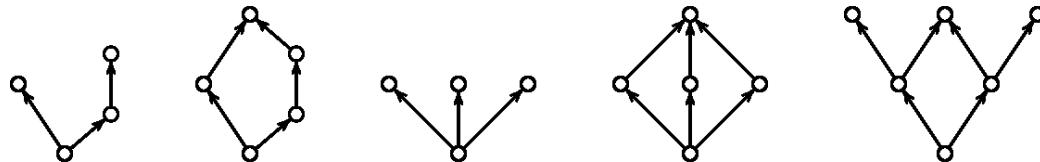
While a general classification of structurally complete logics is not an easy task, we have a good understanding of some special cases.

Intuitionistic logic itself is not structurally complete, but its *fragments* may behave differently. Namely, any disjunction-free rule or implication-free rule admissible in a superintuitionistic logic is derivable.^[32] On the other hand, the Mints rule

$$\frac{(p \rightarrow q) \rightarrow p \vee r}{((p \rightarrow q) \rightarrow p) \vee ((p \rightarrow q) \rightarrow r)}$$

is admissible in intuitionistic logic but not derivable, and contains only implications and disjunctions.

We know the *maximal* structurally incomplete transitive logics. A logic is called **hereditarily structurally complete**, if every its extension is structurally complete. For example, classical logic, as well as the logics *LC* and *Grz.3* mentioned above, are hereditarily structurally complete. A complete description of hereditarily structurally complete superintuitionistic and transitive modal logics was given by Citkin and Rybakov. Namely, a superintuitionistic logic is hereditarily structurally complete if and only if it is not valid in any of the five Kripke frames



Similarly, an extension of *K4* is hereditarily structurally complete if and only if it is not valid in any of certain twenty Kripke frames (including the five intuitionistic frames above).

There exist structurally complete logics that are not hereditarily structurally complete: for example, Medvedev's logic is structurally complete,^[33] but it is included in the structurally incomplete logic *KC*.

Variants

A **rule with parameters** is a rule of the form

$$\frac{A(p_1, \dots, p_n, s_1, \dots, s_k)}{B(p_1, \dots, p_n, s_1, \dots, s_k)},$$

whose variables are divided into the "regular" variables p_i , and the parameters s_i . The rule is L -admissible if every L -unifier σ of A such that $\sigma s_i = s_i$ for each i is also a unifier of B . The basic decidability results for admissible rules also carry to rules with parameters.^[34]

A **multiple-conclusion rule** is a pair (Γ, Δ) of two finite sets of formulas, written as

$$\frac{A_1, \dots, A_n}{B_1, \dots, B_m} \quad \text{or} \quad A_1, \dots, A_n / B_1, \dots, B_m.$$

Such a rule is admissible if every unifier of Γ is also a unifier of some formula from Δ .^[35] For example, a logic L is consistent iff it admits the rule

$$\frac{\perp}{},$$

and a superintuitionistic logic has the disjunction property iff it admits the rule

$$\frac{p \vee q}{p, q}.$$

Again, basic results on admissible rules generalize smoothly to multiple-conclusion rules.^[36] In logics with a variant of the disjunction property, the multiple-conclusion rules have the same expressive power as single-conclusion rules: for example, in $S4$ the rule above is equivalent to

$$\frac{A_1, \dots, A_n}{\Box B_1 \vee \dots \vee \Box B_m}.$$

Nevertheless, multiple-conclusion rules can often be employed to simplify arguments.

In proof theory, admissibility is often considered in the context of sequent calculi, where the basic objects are sequents rather than formulas. For example, one can rephrase the cut-elimination theorem as saying that the cut-free sequent calculus admits the cut rule

$$\frac{\Gamma \vdash A, \Delta \quad \Pi, A \vdash \Lambda}{\Gamma, \Pi \vdash \Delta, \Lambda}.$$

(By abuse of language, it is also sometimes said that the (full) sequent calculus admits cut, meaning its cut-free version does.) However, admissibility in sequent calculi is usually only a notational variant for admissibility in the corresponding logic: any complete calculus for (say) intuitionistic logic admits a sequent rule if and only if IPC admits the formula rule which we obtain by translating each sequent $\Gamma \vdash \Delta$ to its characteristic formula $\bigwedge \Gamma \rightarrow \bigvee \Delta$.

Notes

- [1] Blok & Pigozzi (1989), Kracht (2007)
- [2] Rybakov (1997), Def. 1.1.3
- [3] Rybakov (1997), Def. 1.7.2
- [4] From de Jongh's theorem to intuitionistic logic of proofs (<http://www.illc.uva.nl/D65/artemov.pdf>)
- [5] Rybakov (1997), Def. 1.7.7
- [6] Chagrov & Zakharyashev (1997), Thm. 1.25
- [7] Prucnal (1979), cf. Iemhoff (2006)
- [8] Rybakov (1997), p. 439
- [9] Rybakov (1997), Thms. 5.4.4, 5.4.8
- [10] Cintula & Metcalfe (2009)
- [11] Wolter & Zakharyashev (2008)
- [12] Rybakov (1997), §3.9
- [13] Rybakov (1997), Thm. 3.9.3
- [14] Rybakov (1997), Thms. 3.9.6, 3.9.9, 3.9.12; cf. Chagrov & Zakharyashev (1997), §16.7
- [15] Rybakov (1997), Thm. 3.2.2
- [16] Rybakov (1997), §3.5
- [17] Jeřábek (2007)
- [18] Chagrov & Zakharyashev (1997), §18.5
- [19] Ghilardi (2000), Thm. 2.2
- [20] Ghilardi (2000), p. 196
- [21] Ghilardi (2000), Thm. 3.6
- [22] Rybakov (1997), Def. 1.4.13
- [23] Mints & Kojevnikov (2004)
- [24] Rybakov (1997), Thm. 4.5.5
- [25] Rybakov (1997), §4.2
- [26] Jeřábek (2008)
- [27] Rybakov (1997), Cor. 4.3.20
- [28] Iemhoff (2001, 2005), Rozière (1992)
- [29] Jeřábek (2005)
- [30] Jeřábek (2005, 2008)
- [31] Iemhoff (2001), Jeřábek (2005)
- [32] Rybakov (1997), Thms. 5.5.6, 5.5.9
- [33] Prucnal (1976)
- [34] Rybakov (1997), §6.1
- [35] Jeřábek (2005); cf. Kracht (2007), §7
- [36] Jeřábek (2005, 2007, 2008)

References

- W. Blok, D. Pigozzi, *Algebraizable logics*, Memoirs of the American Mathematical Society 77 (1989), no. 396, 1989.
- A. Chagrov and M. Zakharyashev, *Modal Logic*, Oxford Logic Guides vol. 35, Oxford University Press, 1997. ISBN 0-19-853779-4
- P. Cintula and G. Metcalfe, *Structural completeness in fuzzy logics*, Notre Dame Journal of Formal Logic 50 (2009), no. 2, pp. 153–182. doi: 10.1215/00294527-2009-004 (<http://dx.doi.org/10.1215/00294527-2009-004>)
- A. I. Citkin, *On structurally complete superintuitionistic logics*, Soviet Mathematics Doklady, vol. 19 (1978), pp. 816–819.
- S. Ghilardi, *Unification in intuitionistic logic*, Journal of Symbolic Logic 64 (1999), no. 2, pp. 859–880. Project Euclid (<http://projecteuclid.org/euclid.jsl/1183745815>) JSTOR (<http://www.jstor.org/stable/view/2586506>)
- S. Ghilardi, *Best solving modal equations*, Annals of Pure and Applied Logic 102 (2000), no. 3, pp. 183–198. doi: 10.1016/S0168-0072(99)00032-9 ([http://dx.doi.org/10.1016/S0168-0072\(99\)00032-9](http://dx.doi.org/10.1016/S0168-0072(99)00032-9))

- R. Iemhoff, *On the admissible rules of intuitionistic propositional logic*, Journal of Symbolic Logic 66 (2001), no. 1, pp. 281–294. Project Euclid (<http://projecteuclid.org/euclid.jsl/1183746371>) JSTOR (<http://www.jstor.org/stable/view/2694922>)
- R. Iemhoff, *Intermediate logics and Visser's rules*, Notre Dame Journal of Formal Logic 46 (2005), no. 1, pp. 65–81. doi: 10.1305/ndjfl/1107220674 (<http://dx.doi.org/10.1305/ndjfl/1107220674>)
- R. Iemhoff, *On the rules of intermediate logics*, Archive for Mathematical Logic, 45 (2006), no. 5, pp. 581–599. doi: 10.1007/s00153-006-0320-8 (<http://dx.doi.org/10.1007/s00153-006-0320-8>)
- E. Jeřábek, *Admissible rules of modal logics*, Journal of Logic and Computation 15 (2005), no. 4, pp. 411–431. doi: 10.1093/logcom/exi029 (<http://dx.doi.org/10.1093/logcom/exi029>)
- E. Jeřábek, *Complexity of admissible rules*, Archive for Mathematical Logic 46 (2007), no. 2, pp. 73–92. doi: 10.1007/s00153-006-0028-9 (<http://dx.doi.org/10.1007/s00153-006-0028-9>)
- E. Jeřábek, *Independent bases of admissible rules*, Logic Journal of the IGPL 16 (2008), no. 3, pp. 249–267. doi: 10.1093/jigpal/jzn004 (<http://dx.doi.org/10.1093/jigpal/jzn004>)
- M. Kracht, *Modal Consequence Relations*, in: Handbook of Modal Logic (P. Blackburn, J. van Benthem, and F. Wolter, eds.), Studies of Logic and Practical Reasoning vol. 3, Elsevier, 2007, pp. 492–545. ISBN 978-0-444-51690-9
- P. Lorenzen, *Einführung in die operative Logik und Mathematik*, Grundlehren der mathematischen Wissenschaften vol. 78, Springer–Verlag, 1955.
- G. Mints and A. Kojevnikov, *Intuitionistic Frege systems are polynomially equivalent*, Zapiski Nauchnyh Seminarov POMI 316 (2004), pp. 129–146. gzipped PS (<http://www.emis.de/journals/ZPOMI/v316/p129.ps.gz>)
- T. Prucnal, *Structural completeness of Medvedev's propositional calculus*, Reports on Mathematical Logic 6 (1976), pp. 103–105.
- T. Prucnal, *On two problems of Harvey Friedman*, Studia Logica 38 (1979), no. 3, pp. 247–262. doi: 10.1007/BF00405383 (<http://dx.doi.org/10.1007/BF00405383>)
- P. Rozière, *Règles admissibles en calcul propositionnel intuitionniste*, Ph.D. thesis, Université de Paris VII, 1992. PDF (<http://www.pps.jussieu.fr/~roziere/admiss/these.pdf>)
- V. V. Rybakov, *Admissibility of Logical Inference Rules*, Studies in Logic and the Foundations of Mathematics vol. 136, Elsevier, 1997. ISBN 0-444-89505-1
- F. Wolter, M. Zakharyashev, *Undecidability of the unification and admissibility problems for modal and description logics*, ACM Transactions on Computational Logic 9 (2008), no. 4, article no. 25. doi: 10.1145/1380572.1380574 (<http://dx.doi.org/10.1145/1380572.1380574>) PDF (<http://tocl.acm.org/accepted/318wolter.pdf>)

Associative property

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In mathematics, the **associative property** is a property of some binary operations. In propositional logic, **associativity** is a valid rule of replacement for expressions in logical proofs.

Within an expression containing two or more occurrences in a row of the same associative operator, the order in which the operations are performed does not matter as long as the sequence of the operands is not changed. That is, rearranging the parentheses in such an expression will not change its value. Consider, for instance, the following equations:

$$(5 + 2) + 1 = 5 + (2 + 1) = 8$$

$$5 \times (5 \times 3) = (5 \times 5) \times 3 = 75$$

Consider the first equation. Even though the parentheses were rearranged (the left side requires adding 5 and 2 first, then adding 1 to the result, whereas the right side requires adding 2 and 1 first, then 5), the value of the expression was not altered. Since this holds true when performing addition on any real numbers, we say that "addition of real numbers is an associative operation."

Associativity is not to be confused with commutativity. Commutativity justifies changing the order or sequence of the operands within an expression while associativity does not. For example,

$$(5 + 2) + 1 = 5 + (2 + 1)$$

is an example of associativity because the parentheses were changed (and consequently the order of operations during evaluation) while the operands 5, 2, and 1 appeared in exactly the same order from left to right in the expression. In contrast,

$$(5 + 2) + 1 = (2 + 5) + 1$$

is an example of commutativity, not associativity, because the operand sequence changed when the 2 and 5 switched places.

Associative operations are abundant in mathematics; in fact, many algebraic structures (such as semigroups and categories) explicitly require their binary operations to be associative.

However, many important and interesting operations are non-associative; some examples include subtraction, exponentiation and the vector cross product.

Definition

Formally, a binary operation $*$ on a set S is called **associative** if it satisfies the **associative law**:

$$(x * y) * z = x * (y * z) \quad \text{for all } x, y, z \in S.$$

Here, $*$ is used to replace the symbol of the operation, which may be any symbol, and even the absence of symbol like for the multiplication.

$$(xy)z = x(yz) = xyz \quad \text{for all } x, y, z \in S.$$

The associative law can also be expressed in functional notation thus: $f(f(x, y), z) = f(x, f(y, z))$.

Generalized associative law

If a binary operation is associative, repeated application of the operation produces the same result regardless how valid pairs of parenthesis are inserted in the expression. This is called the **generalized associative law**. For instance, a product of four elements may be written in five possible ways:

1. $((ab)c)d$
2. $(ab)(cd)$
3. $(a(bc))d$
4. $a((bc)d)$
5. $a(b(cd))$

If the product operation is associative, the generalized associative law says that all these formulas will yield the same result, making the parenthesis unnecessary. Thus "the" product can be written unambiguously as

abcd.

As the number of elements increases, the number of possible ways to insert parentheses grows quickly, but they remain unnecessary for disambiguation.

Examples

Some examples of associative operations include the following.

- The concatenation of the three strings "hello", " ", "world" can be computed by concatenating the first two strings (giving "hello ") and appending the third string ("world"), or by joining the second and third string (giving " world") and concatenating the first string ("hello") with the result. The two methods produce the same result; string concatenation is associative (but not commutative).
- In arithmetic, addition and multiplication of real numbers are associative; i.e.,

$$\left. \begin{array}{l} (x+y)+z = x+(y+z) = x+y+z \\ (xy)z = x(yz) = xyz \end{array} \right\} \text{for all } x, y, z \in \mathbb{R}.$$

Because of associativity, the grouping parentheses can be omitted without ambiguity.

- Addition and multiplication of complex numbers and quaternions is associative. Addition of octonions is also associative, but multiplication of octonions is non-associative.
- The greatest common divisor and least common multiple functions act associatively.

$$\left. \begin{array}{l} \gcd(\gcd(x, y), z) = \gcd(x, \gcd(y, z)) = \gcd(x, y, z) \\ \operatorname{lcm}(\operatorname{lcm}(x, y), z) = \operatorname{lcm}(x, \operatorname{lcm}(y, z)) = \operatorname{lcm}(x, y, z) \end{array} \right\} \text{for all } x, y, z \in \mathbb{Z}.$$

- Taking the intersection or the union of sets:

$$\left. \begin{array}{l} (A \cap B) \cap C = A \cap (B \cap C) = A \cap B \cap C \\ (A \cup B) \cup C = A \cup (B \cup C) = A \cup B \cup C \end{array} \right\} \text{for all sets } A, B, C.$$

- If M is some set and S denotes the set of all functions from M to M , then the operation of functional composition on S is associative:

$$(f \circ g) \circ h = f \circ (g \circ h) = f \circ g \circ h \quad \text{for all } f, g, h \in S.$$

- Slightly more generally, given four sets M, N, P and Q , with $h: M$ to N , $g: N$ to P , and $f: P$ to Q , then

$$(f \circ g) \circ h = f \circ (g \circ h) = f \circ g \circ h$$

as before. In short, composition of maps is always associative.

- Consider a set with three elements, A, B, and C. The following operation:

x	A	B	C
A	A	A	A
B	A	B	C
C	A	A	A

is associative. Thus, for example, $A(BC)=(AB)C = A$. This mapping is not commutative.

- Because matrices represent linear transformation functions, with matrix multiplication representing functional composition, one can immediately conclude that matrix multiplication is associative.

Propositional logic

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

Rule of replacement

In standard truth-functional propositional logic, *association*,^{[1][2]} or *associativity*^[3] are two valid rules of replacement. The rules allow one to move parentheses in logical expressions in logical proofs. The rules are:

$$(P \vee (Q \vee R)) \Leftrightarrow ((P \vee Q) \vee R)$$

and

$$(P \wedge (Q \wedge R)) \Leftrightarrow ((P \wedge Q) \wedge R),$$

where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Truth functional connectives

Associativity is a property of some logical connectives of truth-functional propositional logic. The following logical equivalences demonstrate that associativity is a property of particular connectives. The following are truth-functional tautologies.

Associativity of disjunction:

$$\begin{aligned}(P \vee (Q \vee R)) &\leftrightarrow ((P \vee Q) \vee R) \\ ((P \vee Q) \vee R) &\leftrightarrow (P \vee (Q \vee R))\end{aligned}$$

Associativity of conjunction:

$$\begin{aligned}((P \wedge Q) \wedge R) &\leftrightarrow (P \wedge (Q \wedge R)) \\ (P \wedge (Q \wedge R)) &\leftrightarrow ((P \wedge Q) \wedge R)\end{aligned}$$

Associativity of equivalence:

$$\begin{aligned}((P \leftrightarrow Q) \leftrightarrow R) &\leftrightarrow (P \leftrightarrow (Q \leftrightarrow R)) \\ (P \leftrightarrow (Q \leftrightarrow R)) &\leftrightarrow ((P \leftrightarrow Q) \leftrightarrow R)\end{aligned}$$

Non-associativity

A binary operation $*$ on a set S that does not satisfy the associative law is called **non-associative**. Symbolically,

$$(x * y) * z \neq x * (y * z) \quad \text{for some } x, y, z \in S.$$

For such an operation the order of evaluation *does* matter. For example:

- Subtraction

$$(5 - 3) - 2 \neq 5 - (3 - 2)$$

- Division

$$(4/2)/2 \neq 4/(2/2)$$

- Exponentiation

$$2^{(1^2)} \neq (2^1)^2$$

Also note that infinite sums are not generally associative, for example:

$$(1 - 1) + (1 - 1) + (1 - 1) + (1 - 1) + (1 - 1) + (1 - 1) + \dots = 0$$

whereas

$$1 + (-1 + 1) + (-1 + 1) + (-1 + 1) + (-1 + 1) + (-1 + 1) + (-1 + \dots) = 1$$

The study of non-associative structures arises from reasons somewhat different from the mainstream of classical algebra. One area within non-associative algebra that has grown very large is that of Lie algebras. There the associative law is replaced by the Jacobi identity. Lie algebras abstract the essential nature of infinitesimal transformations, and have become ubiquitous in mathematics.

There are other specific types of non-associative structures that have been studied in depth. They tend to come from some specific applications. Some of these arise in combinatorial mathematics. Other examples: Quasigroup, Quasifield, Nonassociative ring.

Notation for non-associative operations

In general, parentheses must be used to indicate the order of evaluation if a non-associative operation appears more than once in an expression. However, mathematicians agree on a particular order of evaluation for several common non-associative operations. This is simply a notational convention to avoid parentheses.

A **left-associative** operation is a non-associative operation that is conventionally evaluated from left to right, i.e.,

$$\left. \begin{array}{l} x * y * z = (x * y) * z \\ w * x * y * z = ((w * x) * y) * z \\ \text{etc.} \end{array} \right\} \text{for all } w, x, y, z \in S$$

while a **right-associative** operation is conventionally evaluated from right to left:

$$\left. \begin{array}{l} x * y * z = x * (y * z) \\ w * x * y * z = w * (x * (y * z)) \\ \text{etc.} \end{array} \right\} \text{for all } w, x, y, z \in S$$

Both left-associative and right-associative operations occur. Left-associative operations include the following:

- Subtraction and division of real numbers:

$$\begin{aligned} x - y - z &= (x - y) - z && \text{for all } x, y, z \in \mathbb{R}; \\ x/y/z &= (x/y)/z && \text{for all } x, y, z \in \mathbb{R} \text{ with } y \neq 0, z \neq 0. \end{aligned}$$

- Function application:

$$(f x y) = ((f x) y)$$

This notation can be motivated by the currying isomorphism.

Right-associative operations include the following:

- Exponentiation of real numbers:

$$x^{yz} = x^{(yz)}.$$

The reason exponentiation is right-associative is that a repeated left-associative exponentiation operation would be less useful. Multiple appearances could (and would) be rewritten with multiplication:

$$(x^y)^z = x^{(yz)}.$$

- Function definition

$$\mathbb{Z} \rightarrow \mathbb{Z} \rightarrow \mathbb{Z} = \mathbb{Z} \rightarrow (\mathbb{Z} \rightarrow \mathbb{Z})$$

$$x \mapsto y \mapsto x - y = x \mapsto (y \mapsto x - y)$$

Using right-associative notation for these operations can be motivated by the Curry-Howard correspondence and by the currying isomorphism.

Non-associative operations for which no conventional evaluation order is defined include the following.

- Taking the Cross product of three vectors:

$$\vec{a} \times (\vec{b} \times \vec{c}) \neq (\vec{a} \times \vec{b}) \times \vec{c} \quad \text{for some } \vec{a}, \vec{b}, \vec{c} \in \mathbb{R}^3$$

- Taking the pairwise average of real numbers:

$$\frac{(x + y)/2 + z}{2} \neq \frac{x + (y + z)/2}{2} \quad \text{for all } x, y, z \in \mathbb{R} \text{ with } x \neq z.$$

- Taking the relative complement of sets $(A \setminus B) \setminus C$ is not the same as $A \setminus (B \setminus C)$. (Compare material nonimplication in logic.)

References

- [1] Moore and Parker
- [2] Copi and Cohen
- [3] Hurley

Biconditional elimination

Transformation rules	
Propositional calculus	
Rules of inference	
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption 	
Rules of replacement	
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology 	
Predicate logic	
Universal generalization	
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation 	

Biconditional elimination is the name of two valid rules of inference of propositional logic. It allows for one to infer a conditional from a biconditional. If $(P \leftrightarrow Q)$ is true, then one may infer that $(P \rightarrow Q)$ is true, and also that $(Q \rightarrow P)$ is true. For example, if it's true that I'm breathing if and only if I'm alive, then it's true that if I'm breathing, I'm alive; likewise, it's true that if I'm alive, I'm breathing. The rules can be stated formally as:

$$\frac{(P \leftrightarrow Q)}{\therefore (P \rightarrow Q)}$$

and

$$\frac{(P \leftrightarrow Q)}{\therefore (Q \rightarrow P)}$$

where the rule is that wherever an instance of " $(P \leftrightarrow Q)$ " appears on a line of a proof, either " $(P \rightarrow Q)$ " or " $(Q \rightarrow P)$ " can be placed on a subsequent line;

Formal notation

The *biconditional elimination* rule may be written in sequent notation:

$$(P \leftrightarrow Q) \vdash (P \rightarrow Q)$$

and

$$(P \leftrightarrow Q) \vdash (Q \rightarrow P)$$

where \vdash is a metalogical symbol meaning that $(P \rightarrow Q)$, in the first case, and $(Q \rightarrow P)$ in the other are syntactic consequences of $(P \leftrightarrow Q)$ in some logical system;

or as the statement of a truth-functional tautology or theorem of propositional logic:

$$(P \leftrightarrow Q) \rightarrow (P \rightarrow Q)$$

$$(P \leftrightarrow Q) \rightarrow (Q \rightarrow P)$$

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

References

Biconditional introduction

Transformation rules
Propositional calculus

Rules of inference
• <i>Modus ponens</i>
• <i>Modus tollens</i>
• Biconditional introduction
• Biconditional elimination
• Conjunction introduction
• Simplification
• Disjunction introduction
• Disjunction elimination
• Disjunctive syllogism
• Hypothetical syllogism
• Constructive dilemma
• Destructive dilemma
• Absorption
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

In propositional logic, **biconditional introduction** is a valid rule of inference. It allows for one to infer a biconditional from two conditional statements. The rule makes it possible to introduce a biconditional statement into a logical proof. If $P \rightarrow Q$ is true, and if $Q \rightarrow P$ is true, then one may infer that $P \leftrightarrow Q$ is true. For example, from the statements "if I'm breathing, then I'm alive" and "if I'm alive, then I'm breathing", it can be inferred that "I'm breathing if and only if I'm alive". Biconditional introduction is the converse of biconditional elimination. The rule can be stated formally as:

$$\frac{P \rightarrow Q, Q \rightarrow P}{\therefore P \leftrightarrow Q}$$

where the rule is that wherever instances of " $P \rightarrow Q$ " and " $Q \rightarrow P$ " appear on lines of a proof, " $P \leftrightarrow Q$ " can validly be placed on a subsequent line.

Formal notation

The *biconditional introduction* rule may be written in sequent notation:

$$(P \rightarrow Q), (Q \rightarrow P) \vdash (P \leftrightarrow Q)$$

where \vdash is a metalogical symbol meaning that $P \leftrightarrow Q$ is a syntactic consequence when $P \rightarrow Q$ and $Q \rightarrow P$ are both in a proof;

or as the statement of a truth-functional tautology or theorem of propositional logic:

$$((P \rightarrow Q) \wedge (Q \rightarrow P)) \rightarrow (P \leftrightarrow Q)$$

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

References

Commutative property

In mathematics, a binary operation is **commutative** if changing the order of the operands does not change the result. It is a fundamental property of many binary operations, and many mathematical proofs depend on it. The commutativity of simple operations, such as multiplication and addition of numbers, was for many years implicitly assumed and the property was not named until the 19th century when mathematics started to become formalized. By contrast, division and subtraction are *not* commutative.

Common uses

The *commutative property* (or *commutative law*) is a property associated with binary operations and functions. Similarly, if the commutative property holds for a pair of elements under a certain binary operation then it is said that the two elements *commute* under that operation.

Propositional logic

Transformation rules
Propositional calculus

Rules of inference
• <i>Modus ponens</i>
• <i>Modus tollens</i>
• Biconditional introduction
• Biconditional elimination
• Conjunction introduction
• Simplification
• Disjunction introduction
• Disjunction elimination
• Disjunctive syllogism
• Hypothetical syllogism
• Constructive dilemma
• Destructive dilemma
• Absorption
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

Rule of replacement

In standard truth-functional propositional logic, *commutation*,^{[1][2]} or *commutivity*^[3] are two valid rules of replacement. The rules allow one to transpose propositional variables within logical expressions in logical proofs. The rules are:

$$(P \vee Q) \Leftrightarrow (Q \vee P)$$

and

$$(P \wedge Q) \Leftrightarrow (Q \wedge P)$$

where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Truth functional connectives

Commutativity is a property of some logical connectives of truth functional propositional logic. The following logical equivalences demonstrate that commutativity is a property of particular connectives. The following are truth-functional tautologies.

Commutativity of conjunction

$$(P \wedge Q) \leftrightarrow (Q \wedge P)$$

Commutativity of disjunction

$$(P \vee Q) \leftrightarrow (Q \vee P)$$

Commutativity of implication (also called the Law of permutation)

$$(P \rightarrow (Q \rightarrow R)) \leftrightarrow (Q \rightarrow (P \rightarrow R))$$

Commutativity of equivalence (also called the Complete commutative law of equivalence)

$$(P \leftrightarrow Q) \leftrightarrow (Q \leftrightarrow P)$$

Set theory

In group and set theory, many algebraic structures are called commutative when certain operands satisfy the commutative property. In higher branches of mathematics, such as analysis and linear algebra the commutativity of well known operations (such as addition and multiplication on real and complex numbers) is often used (or implicitly assumed) in proofs.^{[4][5][6]}

Mathematical definitions

The term "commutative" is used in several related senses.^{[7][8]}

1. A binary operation $*$ on a set S is called *commutative* if:

$$x * y = y * x \quad \text{for all } x, y \in S$$

An operation that does not satisfy the above property is called **noncommutative**.

2. One says that x *commutes* with y under $*$ if:

$$x * y = y * x$$

3. A binary function $f: A \times A \rightarrow B$ is called *commutative* if:

$$f(x, y) = f(y, x) \quad \text{for all } x, y \in A$$

History and etymology

Records of the implicit use of the commutative property go back to ancient times. The Egyptians used the commutative property of multiplication to simplify computing products.^{[9][10]} Euclid is known to have assumed the commutative property of multiplication in his book *Elements*.^[11] Formal uses of the commutative property arose in the late 18th and early 19th centuries, when mathematicians began to work on a theory of functions. Today the commutative property is a well known and basic property used in most branches of mathematics.

(14)

f et f , sont telles qu'elles donnent
el que soit l'ordre dans lequel on les
appelées *commutatives entre elles*.

; aEz=Eaz ;

The first known use of the term was in a French Journal published in
1814

The first recorded use of the term *commutative* was in a memoir by François Servois in 1814,^{[12][13]} which used the word *commutatives* when describing functions that have what is now called the commutative property. The word is a combination of the French word *commuter* meaning "to substitute or switch" and the suffix *-ative* meaning "tending to" so the word literally means "tending to substitute or switch." The term then appeared in English in *Philosophical Transactions of the Royal Society* in 1844.

Related properties

Associativity

The associative property is closely related to the commutative property. The associative property of an expression containing two or more occurrences of the same operator states that the order operations are performed in does not affect the final result, as long as the order of terms doesn't change. In contrast, the commutative property states that the order of the terms does not affect the final result.

Most commutative operations encountered in practice are also associative. However, commutativity does not imply associativity. A counterexample is the function

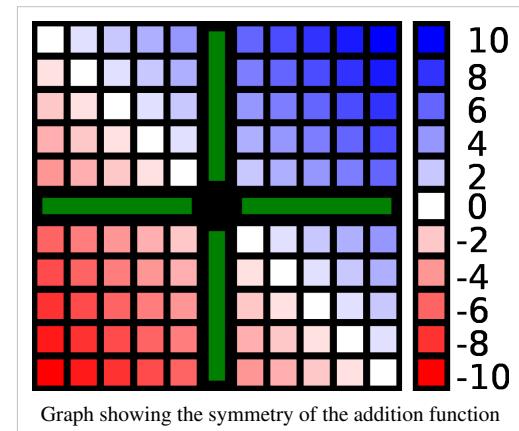
$$f(x, y) = \frac{x + y}{2},$$

which is clearly commutative (interchanging x and y does not affect the result), but it is not associative (since, for example, $f(1, f(2, 3)) = 1.75$ but $f(f(1, 2), 3) = 2.25$).

Symmetry

Some forms of symmetry can be directly linked to commutativity. When a commutative operator is written as a binary function then the resulting function is symmetric across the line $y = x$. As an example, if we let a function f represent addition (a commutative operation) so that $f(x,y) = x + y$ then f is a symmetric function, which can be seen in the image on the right.

For relations, a symmetric relation is analogous to a commutative operation, in that if a relation R is symmetric, then $aRb \Leftrightarrow bRa$



Examples

Commutative operations in everyday life

- Putting on socks resembles a commutative operation, since which sock is put on first is unimportant. Either way, the result (having both socks on), is the same.
- The commutativity of addition is observed when paying for an item with cash. Regardless of the order the bills are handed over in, they always give the same total.

Commutative operations in mathematics

Two well-known examples of commutative binary operations:

- The addition of real numbers is commutative, since

$$y + z = z + y \quad \text{for all } y, z \in \mathbb{R}$$

For example $4 + 5 = 5 + 4$, since both expressions equal 9.

- The multiplication of real numbers is commutative, since

$$yz = zy \quad \text{for all } y, z \in \mathbb{R}$$

For example, $3 \times 5 = 5 \times 3$, since both expressions equal 15.

- Some binary truth functions are also commutative, since the truth tables for the functions are the same when one changes the order of the operands.

For example, $Vpq = Vqp$; $Apq = Aqp$; $Dpq = Dqp$; $Eqp = Eqp$; $Jpq = Jqp$; $Kpq = Kqp$; $Xpq = Xqp$; $Opq = Oqp$.

- Further examples of commutative binary operations include addition and multiplication of complex numbers, addition and scalar multiplication of vectors, and intersection and union of sets.

Noncommutative operations in everyday life

- Concatenation, the act of joining character strings together, is a noncommutative operation. For example

$$EA + T = EAT \neq TEA = T + EA$$

- Washing and drying clothes resembles a noncommutative operation; washing and then drying produces a markedly different result to drying and then washing.
- Rotating a book 90° around a vertical axis then 90° around a horizontal axis produces a different orientation than when the rotations are performed in the opposite order.
- The twists of the Rubik's Cube are noncommutative. This can be studied using group theory.

Noncommutative operations in mathematics

Some noncommutative binary operations:^[14]

- Subtraction is noncommutative, since $0 - 1 \neq 1 - 0$
- Division is noncommutative, since $1/2 \neq 2/1$
- Some truth functions are noncommutative, since the truth tables for the functions are different when one changes the order of the operands.

For example, $Bpq = Cqp$; $Cpq = Bqp$; $Fpq = Gqp$; $Gpq = Fqp$; $Hpq = Iqp$; $Ipq = Hqp$; $Lpq = Mqp$; $Mpq = Lqp$.

- Matrix multiplication is noncommutative since

$$\begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \neq \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

- The vector product (or cross product) of two vectors in three dimensions is anti-commutative, i.e., $b \times a = -(a \times b)$.

Mathematical structures and commutativity

- A commutative semigroup is a set endowed with a total, associative and commutative operation.
- If the operation additionally has an identity element, we have a commutative monoid
- An abelian group, or *commutative group* is a group whose group operation is commutative.
- A commutative ring is a ring whose multiplication is commutative. (Addition in a ring is always commutative.)^[15]
- In a field both addition and multiplication are commutative.^[16]

Non-commuting operators in quantum mechanics

In quantum mechanics as formulated by Schrödinger, physical variables are represented by linear operators such as x (meaning multiply by x), and $\frac{d}{dx}$. These two operators do not commute as may be seen by considering the effect of their compositions $x\frac{d}{dx}$ and $\frac{d}{dx}x$ (also called products of operators) on a one-dimensional wave function $\psi(x)$:

$$x\frac{d}{dx}\psi = x\psi' \neq \frac{d}{dx}x\psi = \psi + x\psi'$$

According to the uncertainty principle of Heisenberg, if the two operators representing a pair of variables do not commute, then that pair of variables are mutually complementary, which means they cannot be simultaneously measured or known precisely. For example, the position and the linear momentum in the x -direction of a particle are represented respectively by the operators x and $-i\hbar\frac{\partial}{\partial x}$ (where \hbar is the reduced Planck constant). This is the same example except for the constant $-i\hbar$, so again the operators do not commute and the physical meaning is that the position and linear momentum in a given direction are complementary.

Notes

- [1] Moore and Parker
- [2] Copi and Cohen
- [3] Hurley
- [4] Axler, p.2
- [5] Gallian, p.34
- [6] p. 26,87
- [7] Krowne, p.1
- [8] Weisstein, *Commute*, p.1
- [9] Lumpkin, p.11
- [10] Gay and Shute, p.?
- [11] O'Conner and Robertson, *Real Numbers*
- [12] Cabillón and Miller, *Commutative and Distributive*
- [13] O'Conner and Robertson, *Servois*
- [14] Yark, p.1
- [15] Gallian p.236
- [16] Gallian p.250

References

Books

- Axler, Sheldon (1997). *Linear Algebra Done Right*, 2e. Springer. ISBN 0-387-98258-2.
Abstract algebra theory. Covers commutativity in that context. Uses property throughout book.
- Goodman, Frederick (2003). *Algebra: Abstract and Concrete, Stressing Symmetry*, 2e. Prentice Hall. ISBN 0-13-067342-0.
Abstract algebra theory. Uses commutativity property throughout book.
- Gallian, Joseph (2006). *Contemporary Abstract Algebra*, 6e. Boston, Mass.: Houghton Mifflin. ISBN 0-618-51471-6.
Linear algebra theory. Explains commutativity in chapter 1, uses it throughout.

Articles

- <http://www.ethnomath.org/resources/lumpkin1997.pdf> Lumpkin, B. (1997). The Mathematical Legacy Of Ancient Egypt - A Response To Robert Palter. Unpublished manuscript.
Article describing the mathematical ability of ancient civilizations.
- Robins, R. Gay, and Charles C. D. Shute. 1987. *The Rhind Mathematical Papyrus: An Ancient Egyptian Text*. London: British Museum Publications Limited. ISBN 0-7141-0944-4
Translation and interpretation of the Rhind Mathematical Papyrus.

Online resources

- Hazewinkel, Michiel, ed. (2001), "Commutativity" (<http://www.encyclopediaofmath.org/index.php?title=p/c023420>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Krowne, Aaron, Commutative (<http://planetmath.org/encyclopedia/Commutative.html>) at PlanetMath, Accessed 8 August 2007.
Definition of commutativity and examples of commutative operations
- Weisstein, Eric W., " Commute (<http://mathworld.wolfram.com/Commute.html>)", *MathWorld*., Accessed 8 August 2007.
Explanation of the term commute
- Yark (<http://planetmath.org/?op=getuser&id=2760>). Examples of non-commutative operations (<http://planetmath.org/encyclopedia/ExampleOfCommutative.html>) at PlanetMath, Accessed 8 August 2007
Examples proving some noncommutative operations
- O'Conner, J J and Robertson, E F. MacTutor history of real numbers (http://www-history.mcs.st-andrews.ac.uk/HistTopics/Real_numbers_1.html), Accessed 8 August 2007
Article giving the history of the real numbers
- Cabillón, Julio and Miller, Jeff. Earliest Known Uses Of Mathematical Terms (<http://jeff560.tripod.com/c.html>), Accessed 22 November 2008
Page covering the earliest uses of mathematical terms
- O'Conner, J J and Robertson, E F. MacTutor biography of François Servois (<http://www-groups.dcs.st-and.ac.uk/~history/Biographies/Servois.html>), Accessed 8 August 2007
Biography of Francois Servois, who first used the term

Commutativity of conjunction

In propositional logic, the **commutativity of conjunction** is a valid argument form and truth-functional tautology. It is considered to be a law of classical logic. It is the principle that the conjuncts of a logical conjunction may switch places with each other, while preserving the truth-value of the resulting proposition.

Formal notation

Commutativity of conjunction can be expressed in sequent notation as:

$$(P \wedge Q) \vdash (Q \wedge P)$$

and

$$(Q \wedge P) \vdash (P \wedge Q)$$

where \vdash is a metalogical symbol meaning that $(Q \wedge P)$ is a syntactic consequence of $(P \wedge Q)$, in the one case, and $(P \wedge Q)$ is a syntactic consequence of $(Q \wedge P)$ in the other, in some logical system; or in rule form:

$$\frac{P \wedge Q}{\therefore Q \wedge P}$$

and

$$\frac{Q \wedge P}{\therefore P \wedge Q}$$

where the rule is that wherever an instance of " $(P \wedge Q)$ " appears on a line of a proof, it can be replaced with " $(Q \wedge P)$ " and wherever an instance of " $(Q \wedge P)$ " appears on a line of a proof, it can be replaced with " $(P \wedge Q)$ ";

or as the statement of a truth-functional tautology or theorem of propositional logic:

$$(P \wedge Q) \rightarrow (Q \wedge P)$$

and

$$(Q \wedge P) \rightarrow (P \wedge Q)$$

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

Generalized principle

For any propositions H_1, H_2, \dots, H_n , and permutation $\sigma(n)$ of the numbers 1 through n , it is the case that:

$$H_1 \wedge H_2 \wedge \dots \wedge H_n$$

is equivalent to

$$H_{\sigma(1)} \wedge H_{\sigma(2)} \wedge \dots \wedge H_{\sigma(n)}.$$

For example, if H_1 is

It is raining

H_2 is

Socrates is mortal

and H_3 is

$2+2=4$

then

It is raining and Socrates is mortal and $2+2=4$

is equivalent to

Socrates is mortal and 2+2=4 and it is raining

and the other orderings of the predicates.

References

Conjunction introduction

Transformation rules	
Propositional calculus	
Rules of inference	
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption 	
Rules of replacement	
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology 	
Predicate logic	
Universal generalization	
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation 	

Conjunction introduction (often abbreviated simply as **conjunction**^{[1][2]}) is a valid rule of inference of propositional logic. The rule makes it possible to introduce a conjunction into a logical proof. It is the inference that if the proposition p is true, and proposition q is true, then the logical conjunction of the two propositions p and q is true. For example, if it's true that it's raining, and it's true that I'm inside, then it's true that "it's raining and I'm inside". The rule can be stated:

$$\frac{P, Q}{\therefore P \wedge Q}$$

where the rule is that wherever an instance of " P " and " Q " appear on lines of a proof, a " $P \wedge Q$ " can be placed on a subsequent line.

Formal notation

The *conjunction introduction* rule may be written in sequent notation:

$$P, Q \vdash P \wedge Q$$

where \vdash is a metalogical symbol meaning that $P \wedge Q$ is a syntactic consequence if P and Q are each on lines of a proof in some logical system;

where P and Q are propositions expressed in some logical system.

References

- [1] Copi and Cohen
- [2] Moore and Parker

Constructive dilemma

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • 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

Constructive dilemma^{[1][2][3]} is a name of a valid rule of inference of propositional logic. It is the inference that, if P implies Q and R implies S and either P or R is true, then Q or S has to be true. In sum, if two conditionals are true and at least one of their antecedents is, then at least one of their consequents must be too. *Constructive dilemma* is the disjunctive version of modus ponens, whereas, destructive dilemma is the disjunctive version of modus tollens. The rule can be stated:

$$\frac{P \rightarrow Q, R \rightarrow S, P \vee R}{\therefore Q \vee S}$$

where the rule is that whenever instances of " $P \rightarrow Q$ ", " $R \rightarrow S$ ", and " $P \vee R$ " appear on lines of a proof, " $Q \vee S$ " can be placed on a subsequent line.

Formal notation

The *constructive dilemma* rule may be written in sequent notation:

$$(P \rightarrow Q), (R \rightarrow S), (P \vee R) \vdash (Q \vee S)$$

where \vdash is a metalogical symbol meaning that $Q \vee S$ is a syntactic consequence of $P \rightarrow Q$, $R \rightarrow S$, and $Q \vee S$ in some logical system;

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

$$((P \rightarrow Q) \wedge (R \rightarrow S)) \wedge (P \vee R) \rightarrow (Q \vee S)$$

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

Variable English

If P then Q . If R then S . P or R . Therefore, Q or S .

Natural language example

If I win a million dollars, I will donate it to an orphanage.

If my friend wins a million dollars, he will donate it to a wildlife fund.

Either I win a million dollars, or my friend wins a million dollars.

Therefore, either an orphanage will get a million dollars, or a wildlife fund will get a million dollars.

The dilemma derives its name because of the transfer of disjunctive operants.

References

- [1] Hurley, Patrick. A Concise Introduction to Logic With Ilrn Printed Access Card. Wadsworth Pub Co, 2008. Page 361
- [2] Moore and Parker
- [3] Copi and Cohen

Contraposition (traditional logic)

In traditional logic, **contraposition** is a form of immediate inference in which from a given proposition another is inferred having for its subject the contradictory of the original predicate, and in some cases involving a change of quality (affirmation or negation).^[1] For its symbolic expression in modern logic see the rule of transposition. Contraposition also has distinctive applications in its philosophical application distinct from the other traditional inference processes of conversion and obversion where equivocation varies with different proposition types.

Traditional logic

In traditional logic the process of contraposition is a schema composed of several steps of inference involving categorical propositions and classes.^[2] A categorical proposition contains a subject and predicate where the existential import of the copula implies the proposition as referring to a class with at least one member, in contrast to the conditional form of hypothetical or materially implicative propositions, which are compounds of other propositions, e.g. *If P, then Q*, where P and Q are both propositions, and their existential import is dependent upon further propositions where in quantification existence is instantiated (existential instantiation).

Conversion by contraposition is the simultaneous interchange and negation of the subject and predicate, and is valid only for the type "A" and type "O" propositions of Aristotelian logic, with considerations for the validity an "E" type proposition with limitations and changes in quantity. This is considered full contraposition. Since in the process of contraposition the obverse can be obtained in all four types of traditional propositions, yielding propositions with the contradictory of the original predicate, contraposition is first obtained by converting the obvert of the original proposition. Thus, partial contraposition can be obtained conditionally in an "E" type proposition with a change in quantity. Because nothing is said in the definition of contraposition with regard to the predicate of the inferred proposition, it can be either the original subject, or its contradictory, resulting in two contrapositives which are the obverts of one another in the "A", "O", and "E" type propositions.^[3]

By example: from an original, 'A' type categorical proposition,

All residents are voters,

which presupposes that all classes have members and the existential import presumed in the form of categorical propositions, one can derive first by obversion the 'E' type proposition,

No residents are non-voters.

The contrapositive of the original proposition is then derived by conversion to another 'E' type proposition,

No non-voters are residents.

The process is completed by further obversion resulting in the 'A' type proposition that is the obverted contrapositive of the original proposition,

All non-voters are non-residents.

The schema of contraposition:^[4]

Original Proposition	Obversion		Contraposition	Obverted Contraposition
(A) All S is P	(E) No S is non-P	\leftrightarrow	(E) No non-P is S	(A) All non-P is non-S
(E) No S is P	(A) All S is non-P		None	None
(I) Some S is P	(O) Some S is not non-P		None	None
(O) Some S is not P	(I) Some S is non-P	\leftrightarrow	(I) Some non-P is S	(O) Some non-P is not non-S

Notice that contraposition is a valid form of immediate inference only when applied to "A" and "O" propositions. It is not valid for "I" propositions, where the obverse is an "O" proposition which has no converse. The contraposition of the "E" proposition is valid only with limitations (*per accidens*). This is because the obverse of the "E" proposition is an "A" proposition which cannot be validly converted except by limitation, that is, contraposition plus a change in the quantity of the proposition from universal to particular.

Also, notice that contraposition is a method of inference which may require the use of other rules of inference. The contrapositive is the product of the method of contraposition, with different outcomes depending upon whether the contraposition is full, or partial. The successive applications of conversion and obversion within the process of contraposition may be given by a variety of names.

The process of the logical equivalence of a statement and its contrapositive as defined in traditional class logic is *not* one of the axioms of propositional logic. In traditional logic there is more than one contrapositive inferred from each original statement. In regard to the "A" proposition this is circumvented in the symbolism of modern logic by the rule of transposition, or the law of contraposition. In its technical usage within the field of philosophic logic, the term "contraposition" may be limited by logicians (e.g. Irving Copi, Susan Stebbing) to traditional logic and categorical propositions. In this sense the use the term "contraposition" is usually referred to by "transposition" when applied to hypothetical propositions or material implications.

Notes

- [1] Brody, Bobuch A. "Glossary of Logical Terms". *Encyclopedia of Philosophy*. Vol. 5-6, p. 61. Macmillan, 1973. Also, Stebbing, L. Susan. *A Modern Introduction to Logic*. Seventh edition, p.65-66. Harper, 1961, and Irving Copi's *Introduction to Logic*, p. 141, Macmillan, 1953. All sources give virtually identical definitions.
- [2] Irving Copi's *Introduction to Logic*, pp. 123-157, Macmillan, 1953.
- [3] Brody, p. 61. Macmillan, 1973. Also, Stebbing, p.65-66, Harper, 1961, and Copi, p. 141-143, Macmillan, 1953.
- [4] Stebbing, L. Susan. *A Modern Introduction to Logic*. Seventh edition, p. 66. Harper, 1961.

References

- Blumberg, Albert E. "Logic, Modern". *Encyclopedia of Philosophy*, Vol.5, Macmillan, 1973.
- Brody, Bobuch A. "Glossary of Logical Terms". *Encyclopedia of Philosophy*. Vol. 5-6, p. 61. Macmillan, 1973.
- Copi, Irving. *Introduction to Logic*. MacMillan, 1953.
- Copi, Irving. *Symbolic Logic*. MacMillan, 1979, fifth edition.
- Prior, A.N. "Logic, Traditional". *Encyclopedia of Philosophy*, Vol.5, Macmillan, 1973.
- Stebbing, Susan. *A Modern Introduction to Logic*. Cromwell Company, 1931.

Destructive dilemma

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

Destructive dilemma^{[1][2]} is the name of a valid rule of inference of propositional logic. It is the inference that, if P implies Q and R implies S and either Q is false or S is false, then either P or R must be false. In sum, if two conditionals are true, but one of their consequents is false, then one of their antecedents has to be false. *Destructive dilemma* is the disjunctive version of *modus tollens*. The disjunctive version of *modus ponens* is the constructive dilemma. The rule can be stated:

$$\frac{P \rightarrow Q, R \rightarrow S, \neg Q \vee \neg S}{\therefore \neg P \vee \neg R}$$

where the rule is that wherever instances of " $P \rightarrow Q$ ", " $R \rightarrow S$ ", and " $\neg Q \vee \neg S$ " appear on lines of a proof, " $\neg P \vee \neg R$ " can be placed on a subsequent line.

Formal notation

The *destructive dilemma* rule may be written in sequent notation:

$$(P \rightarrow Q), (R \rightarrow S), (\neg Q \vee \neg S) \vdash (\neg P \vee \neg R)$$

where \vdash is a metalogical symbol meaning that $\neg P \vee \neg R$ is a syntactic consequence of $P \rightarrow Q, R \rightarrow S$, and $\neg Q \vee \neg S$ in some logical system;

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

$$((P \rightarrow Q) \wedge (R \rightarrow S)) \wedge (\neg Q \vee \neg S) \rightarrow (\neg P \vee \neg R)$$

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

Natural language example

If it rains, we will stay inside.

If it is sunny, we will go for a walk.

Either we will not stay inside, or we will not go for a walk.

Therefore, either it will not rain, or it will not be sunny.

Proof

<i>Proposition</i>	<i>Derivation</i>
$(A \rightarrow B) \wedge (C \rightarrow D)$	Given
$\neg B \vee \neg D$	Given
$B \rightarrow \neg D$	Material implication
$\neg D \rightarrow \neg C$	Transposition
$B \rightarrow \neg C$	Hypothetical syllogism
$A \rightarrow B$	Simplification
$A \rightarrow \neg C$	Hypothetical syllogism
$\neg A \vee \neg C$	Material implication

{}
}

Example proof

The validity of this argument structure can be shown by using both conditional proof (CP) and reductio ad absurdum (RAA) in the following way:

1. $((P \rightarrow Q) \ \& \ (R \rightarrow S)) \ \& \ (\neg Q \vee \neg S)$ (CP assumption)
2. $(P \rightarrow Q) \ \& \ (R \rightarrow S)$ (1: Simplification)
3. $(P \rightarrow Q)$ (2: simplification)
4. $(R \rightarrow S)$ (2: simplification)
5. $(\neg Q \vee \neg S)$ (1: simplification)
6. $\neg(\neg P \vee \neg R)$ (RAA assumption)
7. $\neg\neg P \ \& \ \neg\neg R$ (6: DeMorgan's Law)
8. $\neg\neg P$ (7: simplification)
9. $\neg\neg R$ (7: simplification)
10. P (8: double negation)
11. R (9: double negation)
12. Q (3,10: modus ponens)
13. S (4,11: modus ponens)
14. $\neg\neg Q$ (12: double negation)
15. $\neg S$ (5, 14: disjunctive syllogism)
16. $S \ \& \ \neg S$ (13,15: conjunction)
17. $\neg P \vee \neg R$ (6-16: RAA)
18. $((((P \rightarrow Q) \ \& \ (R \rightarrow S)) \ \& \ (\neg Q \vee \neg S))) \rightarrow \neg P \vee \neg R$ (1-17: CP)

References

- [1] Hurley, Patrick. A Concise Introduction to Logic With Ilrn Printed Access Card. Wadsworth Pub Co, 2008. Page 361
[2] Moore and Parker

- Howard-Snyder, Frances; Howard-Snyder, Daniel; Wasserman, Ryan. The Power of Logic (4th ed.). McGraw-Hill, 2009, ISBN 978-0-07-340737-1, p. 414.

External links

- <http://mathworld.wolfram.com/DestructiveDilemma.html>

Disjunction elimination

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

For the theorem of propositional logic which expresses Disjunction elimination, see Case analysis.

In propositional logic, **disjunction elimination**^{[1][2][3]} (sometimes named **proof by cases** or **case analysis**), is the valid argument form and rule of inference that allows one to eliminate a disjunctive statement from a logical proof. It is the inference that if a statement P implies a statement Q and a statement R also implies Q , then if either P or R is true, then Q has to be true. The reasoning is simple: since at least one of the statements P and R is true, and since either of them would be sufficient to entail Q , Q is certainly true.

If I'm inside, I have my wallet on me.

If I'm outside, I have my wallet on me.

It is true that either I'm inside or I'm outside.

Therefore, I have my wallet on me.

It is the rule can be stated as:

$$\frac{P \rightarrow Q, R \rightarrow Q, P \vee R}{\therefore Q}$$

where the rule is that whenever instances of " $P \rightarrow Q$ ", and " $R \rightarrow Q$ " and " $P \vee R$ " appear on lines of a proof, " Q " can be placed on a subsequent line.

Formal notation

The *disjunction elimination* rule may be written in sequent notation:

$$(P \rightarrow Q), (R \rightarrow Q), (P \vee R) \vdash Q$$

where \vdash is a metalogical symbol meaning that Q is a syntactic consequence of $P \rightarrow Q$, and $R \rightarrow Q$ and $P \vee R$ in some logical system;

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

$$((P \rightarrow Q) \wedge (R \rightarrow Q)) \wedge (P \vee R) \rightarrow Q$$

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

References

- [1] http://www.wordiq.com/definition/Disjunction_elimination
- [2] <http://www.lawrence.edu/dept/philosophy/research/ryckmant/Disjunction%20Elimination.htm>
- [3] <http://www.cs.gsu.edu/~cscskp/Automata/proofs/node6.html>

Disjunction introduction

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • 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

Disjunction introduction or addition^{[1][2][3]} is a simple valid argument form, an immediate inference and a rule of inference of propositional logic. The rule makes it possible to introduce disjunctions to logical proofs. It is the inference that if P is true, then P or Q must be true.

Socrates is a man.

Therefore, either Socrates is a man or pigs are flying in formation over the English Channel.

The rule can be expressed as:

$$\frac{P}{\therefore P \vee Q}$$

where the rule is that whenever instances of " P " appear on lines of a proof, " $P \vee Q$ " can be placed on a subsequent line.

Disjunction introduction is controversial in paraconsistent logic because in combination with other rules of logic, it leads to explosion (i.e. everything becomes provable). See Tradeoffs in Paraconsistent logic.

Formal notation

The *disjunction introduction* rule may be written in sequent notation:

$$P \vdash (P \vee Q)$$

where \vdash is a metalogical symbol meaning that $P \vee Q$ is a syntactic consequence of P in some logical system; and expressed as a truth-functional tautology or theorem of propositional logic:

$$P \rightarrow (P \vee Q)$$

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

References

- [1] Hurley
- [2] Moore and Parker
- [3] Copi and Cohen

Disjunctive syllogism

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In classical logic **disjunctive syllogism** (historically known as **modus tollendo ponens**) is a valid argument form which is a syllogism having a disjunctive statement for one of its premises.^{[1][2]}

Either the breach is a safety violation, or it is not subject to fines.

The breach is not a safety violation.

Therefore, it is not subject to fines.

In propositional logic, **disjunctive syllogism** (also known as **disjunction elimination** and **or elimination**, or abbreviated $\perp E$),^{[3][4][5][6]} is a valid rule of inference. If we are told that at least one of two statements is true; and also told that it is not the former that is true; we can infer that it has to be the latter that is true. If either P or Q is true and P is false, then Q is true. The reason this is called "disjunctive syllogism" is that, first, it is a syllogism, a three-step argument, and second, it contains a logical disjunction, which simply means an "or" statement. "Either P or Q " is a disjunction; P and Q are called the statement's *disjuncts*. The rule makes it possible to eliminate a disjunction from a logical proof. It is the rule that:

$$\frac{P \vee Q, \neg P}{\therefore Q}$$

where the rule is that whenever instances of " $P \vee Q$ ", and " $\neg P$ " appear on lines of a proof, " Q " can be placed on a subsequent line.

Disjunctive syllogism is closely related and similar to hypothetical syllogism, in that it is also type of syllogism, and also the name of a rule of inference.

Formal notation

The *disjunctive syllogism* rule may be written in sequent notation:

$$P \vee Q, \neg P \vdash Q$$

where \vdash is a metalogical symbol meaning that Q is a syntactic consequence of $P \vee Q$, and $\neg P$ in some logical system;

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

$$((P \vee Q) \wedge \neg P) \rightarrow Q$$

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

Natural language examples

Here is an example:

Either I will choose soup or I will choose salad.

I will not choose soup.

Therefore, I will choose salad.

Here is another example:

It is either red or blue.

It is not blue.

Therefore, it is red.

Inclusive and exclusive disjunction

Please observe that the disjunctive syllogism works whether 'or' is considered 'exclusive' or 'inclusive' disjunction. See below for the definitions of these terms.

There are two kinds of logical disjunction:

- *inclusive* means "and/or" - at least one of them is true, or maybe both.
- *exclusive* ("xor") means exactly one must be true, but they cannot both be.

The widely used English language concept of *or* is often ambiguous between these two meanings, but the difference is pivotal in evaluating disjunctive arguments.

This argument:

Either P or Q.

Not P.

Therefore, Q.

is valid and indifferent between both meanings. However, only in the *exclusive* meaning is the following form valid:

Either P or Q (exclusive).

P.

Therefore, not Q.

With the *inclusive* meaning you could draw no conclusion from the first two premises of that argument. See affirming a disjunct.

Related argument forms

Unlike modus ponendo ponens and modus ponendo tollens, with which it should not be confused, disjunctive syllogism is often not made an explicit rule or axiom of logical systems, as the above arguments can be proven with a (slightly devious) combination of reductio ad absurdum and disjunction elimination.

Other forms of syllogism:

- hypothetical syllogism
- categorical syllogism

Disjunctive syllogism holds in classical propositional logic and intuitionistic logic, but not in some paraconsistent logics.^[7]

References

- [1] Hurley
- [2] Copi and Cohen
- [3] Sanford, David Hawley. 2003. *If P, Then Q: Conditionals and the Foundations of Reasoning*. London, UK: Routledge: 39
- [4] Hurley
- [5] Copi and Cohen
- [6] Moore and Parker
- [7] Chris Mortensen, Inconsistent Mathematics (<http://plato.stanford.edu/entries/mathematics-inconsistent/>), *Stanford encyclopedia of philosophy*, First published Tue Jul 2, 1996; substantive revision Thu Jul 31, 2008

Distributive property

In abstract algebra and logic, the **distributive property** of binary operations generalizes the **distributive law** from elementary algebra. In propositional logic, **distribution** refers to two valid rules of replacement. The rules allow one to reformulate conjunctions and disjunctions within logical proofs.

For example, in arithmetic:

$$2 \times (1 + 3) = (2 \times 1) + (2 \times 3) \text{ but } 2 / (1 + 3) \neq (2 / 1) + (2 / 3).$$

In the left-hand side of the first equation, the 2 multiplies the sum of 1 and 3; on the right-hand side, it multiplies the 1 and the 3 individually, with the products added afterwards. Because these give the same final answer (8), we say that multiplication by 2 *distributes* over addition of 1 and 3. Since we could have put any real numbers in place of 2, 1, and 3 above, and still have obtained a true equation, we say that multiplication of real numbers *distributes* over addition of real numbers.

Definition

Given a set S and two binary operators \cdot and $+$ on S , we say that the operation \cdot

- is *left-distributive* over $+$ if, given any elements x, y , and z of S ,

$$x \cdot (y + z) = (x \cdot y) + (x \cdot z);$$

- is *right-distributive* over $+$ if, given any elements x, y , and z of S :

$$(y + z) \cdot x = (y \cdot x) + (z \cdot x);$$

- is *distributive* over $+$ if it is left- and right-distributive.^[1]

Notice that when \cdot is commutative, then the three above conditions are logically equivalent.

Propositional logic

Transformation rules
Propositional calculus
Rules of inference <ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement <ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization <ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

Rule of replacement

In standard truth-functional propositional logic, *distribution*^{[2][3][4]} are two valid rule of replacement. The rules allow one to distribute certain logical connectives within logical expressions in logical proofs. The rules are:

$$(P \wedge (Q \vee R)) \Leftrightarrow ((P \wedge Q) \vee (P \wedge R))$$

and

$$(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$$

where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Truth functional connectives

Distributivity is a property of some logical connectives of truth-functional propositional logic. The following logical equivalences demonstrate that distributivity is a property of particular connectives. The following are truth-functional tautologies.

Distribution of conjunction over conjunction

$$(P \wedge (Q \wedge R)) \Leftrightarrow ((P \wedge Q) \wedge (P \wedge R))$$

Distribution of conjunction over disjunction^[5]

$$(P \wedge (Q \vee R)) \Leftrightarrow ((P \wedge Q) \vee (P \wedge R))$$

Distribution of disjunction over conjunction^[6]

$$(P \vee (Q \wedge R)) \Leftrightarrow ((P \vee Q) \wedge (P \vee R))$$

Distribution of disjunction over disjunction

$$(P \vee (Q \vee R)) \Leftrightarrow ((P \vee Q) \vee (P \vee R))$$

Distribution of implication

$$(P \rightarrow (Q \rightarrow R)) \Leftrightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$$

Distribution of implication over equivalence

$$P \rightarrow (Q \leftrightarrow R) \Leftrightarrow ((P \rightarrow Q) \leftrightarrow (P \rightarrow R))$$

Distribution of disjunction over equivalence

$$(P \vee (Q \leftrightarrow R)) \Leftrightarrow ((P \vee Q) \leftrightarrow (P \vee R))$$

Double distribution

$$((P \wedge Q) \vee (R \wedge S)) \Leftrightarrow (((P \vee R) \wedge (P \vee S)) \wedge ((Q \vee R) \wedge (Q \vee S)))$$

$$((P \vee Q) \wedge (R \vee S)) \Leftrightarrow (((P \wedge R) \vee (P \wedge S)) \vee ((Q \wedge R) \vee (Q \wedge S)))$$

Examples

1. Multiplication of numbers is distributive over addition of numbers, for a broad class of different kinds of numbers ranging from natural numbers to complex numbers and cardinal numbers.
2. Multiplication of ordinal numbers, in contrast, is only left-distributive, not right-distributive.
3. The cross product is left- and right-distributive over vector addition, though not commutative.
4. Matrix multiplication is distributive over matrix addition, though also not commutative.
5. The union of sets is distributive over intersection, and intersection is distributive over union.
6. Logical disjunction ("or") is distributive over logical conjunction ("and"), and conjunction is distributive over disjunction.
7. For real numbers (and for any totally ordered set), the maximum operation is distributive over the minimum operation, and vice-versa: $\max(a,\min(b,c)) = \min(\max(a,b),\max(a,c))$ and $\min(a,\max(b,c)) = \max(\min(a,b),\min(a,c))$.

8. For integers, the greatest common divisor is distributive over the least common multiple, and vice-versa:
 $\gcd(a, \text{lcm}(b, c)) = \text{lcm}(\gcd(a, b), \gcd(a, c))$ and $\text{lcm}(a, \gcd(b, c)) = \gcd(\text{lcm}(a, b), \text{lcm}(a, c))$.
9. For real numbers, addition distributes over the maximum operation, and also over the minimum operation: $a + \max(b, c) = \max(a+b, a+c)$ and $a + \min(b, c) = \min(a+b, a+c)$.

Distributivity and rounding

In practice, the distributive property of multiplication (and division) over addition may appear to be compromised or lost because of the limitations of arithmetic precision. For example, the identity $\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = (1+1+1)/3$ appears to fail if the addition is conducted in decimal arithmetic; however, if many significant digits are used, the calculation will result in a closer approximation to the correct results. For example, if the arithmetical calculation takes the form: $0.33333+0.33333+0.33333 = 0.99999 \neq 1$, this result is a closer approximation than if fewer significant digits had been used. Even when fractional numbers can be represented exactly in arithmetical form, errors will be introduced if those arithmetical values are rounded or truncated. For example, buying two books, each priced at £14.99 before a tax of 17.5%, in two separate transactions will actually save £0.01, over buying them together: $\text{£}14.99 \times 1.175 = \text{£}17.61$ to the nearest £0.01, giving a total expenditure of £35.22, but $\text{£}29.98 \times 1.175 = \text{£}35.23$. Methods such as banker's rounding may help in some cases, as may increasing the precision used, but ultimately some calculation errors are inevitable.

Distributivity in rings

Distributivity is most commonly found in rings and distributive lattices.

A ring has two binary operations (commonly called "+" and "*"), and one of the requirements of a ring is that * must distribute over +. Most kinds of numbers (example 1) and matrices (example 4) form rings. A lattice is another kind of algebraic structure with two binary operations, \wedge and \vee . If either of these operations (say \wedge) distributes over the other (\vee), then \vee must also distribute over \wedge , and the lattice is called distributive. See also the article on distributivity (order theory).

Examples 4 and 5 are Boolean algebras, which can be interpreted either as a special kind of ring (a Boolean ring) or a special kind of distributive lattice (a Boolean lattice). Each interpretation is responsible for different distributive laws in the Boolean algebra. Examples 6 and 7 are distributive lattices which are not Boolean algebras.

Failure of one of the two distributive laws brings about near-rings and near-fields instead of rings and division rings respectively. The operations are usually configured to have the near-ring or near-field distributive on the right but not on the left.

Rings and distributive lattices are both special kinds of rigs, certain generalizations of rings. Those numbers in example 1 that don't form rings at least form rigs. Near-rigs are a further generalization of rigs that are left-distributive but not right-distributive; example 2 is a near-rig.

Generalizations of distributivity

In several mathematical areas, generalized distributivity laws are considered. This may involve the weakening of the above conditions or the extension to infinitary operations. Especially in order theory one finds numerous important variants of distributivity, some of which include infinitary operations, such as the infinite distributive law; others being defined in the presence of only *one* binary operation, such as the according definitions and their relations are given in the article distributivity (order theory). This also includes the notion of a **completely distributive lattice**.

In the presence of an ordering relation, one can also weaken the above equalities by replacing = by either \leq or \geq . Naturally, this will lead to meaningful concepts only in some situations. An application of this principle is the notion of **sub-distributivity** as explained in the article on interval arithmetic.

In category theory, if (S, μ, η) and (S', μ', η') are monads on a category C , a **distributive law** $S.S' \rightarrow S'.S$ is a natural transformation $\lambda : S.S' \rightarrow S'.S$ such that (S', λ) is a lax map of monads $S \rightarrow S'$ and (S, λ) is a colax map of monads $S' \rightarrow S$. This is exactly the data needed to define a monad structure on $S'.S$: the multiplication map is $S'\mu.\mu'S^2.S'\lambda.S$ and the unit map is $\eta'S.\eta$. See: distributive law between monads.

A generalized distributive law has also been proposed in the area of information theory.

Notes

- [1] Ayres, p. 20 (<http://books.google.com/books?id=7r3bVWx2GHkC&pg=PA20&dq=%22binary+operation%22+Left+distributive%22#PPA20,M1>).
- [2] Moore and Parker
- [3] Copi and Cohen
- [4] Hurley
- [5] Russell and Whitehead, *Principia Mathematica*
- [6] Russell and Whitehead, *Principia Mathematica*

References

- Ayres, Frank, *Schaum's Outline of Modern Abstract Algebra*, McGraw-Hill; 1st edition (June 1, 1965). ISBN 0-07-002655-6.

External links

- A demonstration of the Distributive Law (<http://www.cut-the-knot.org/Curriculum/Arithmetic/DistributiveLaw.shtml>) for integer arithmetic (from cut-the-knot)

Double negative elimination

Transformation rules
Propositional calculus

Rules of inference
• <i>Modus ponens</i>
• <i>Modus tollens</i>
• Biconditional introduction
• Biconditional elimination
• Conjunction introduction
• Simplification
• Disjunction introduction
• Disjunction elimination
• Disjunctive syllogism
• Hypothetical syllogism
• Constructive dilemma
• Destructive dilemma
• Absorption
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

For the theorem of propositional logic based on the same concept, see double negation.

In propositional logic, **double negative elimination** (also called **double negation elimination**, **double negative introduction**, **double negation introduction**, or simply **double negation**^{[1][2][3]}) are two valid rules of replacement. They are the inferences that if A is true, then *not not- A* is true and its converse, that, if *not not- A* is true, then A is true. The rule allows one to introduce or eliminate a negation from a logical proof. The rule is based on the equivalence of, for example, *It is false that it is not raining*. and *It is raining*.

The *double negation introduction* rule is:

$$P \Leftrightarrow \neg\neg P$$

and the *double negation elimination* rule is:

$$\neg\neg P \Leftrightarrow P$$

Where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Formal notation

The *double negation introduction* rule may be written in sequent notation:

$$P \vdash \neg\neg P$$

The *double negation elimination* rule may be written as:

$$\neg\neg P \vdash P$$

In rule form:

$$\frac{P}{\neg\neg P}$$

and

$$\frac{\neg\neg P}{P}$$

or as a tautology (plain propositional calculus sentence):

$$P \rightarrow \neg\neg P$$

and

$$\neg\neg P \rightarrow P$$

These can be combined together into a single biconditional formula:

$$\neg\neg P \leftrightarrow P.$$

Since biconditionality is an equivalence relation, any instance of $\neg\neg A$ in a well-formed formula can be replaced by A , leaving unchanged the truth-value of the well-formed formula.

Double negation elimination is a theorem of classical logic, but not of weaker logics such as intuitionistic logic and minimal logic. Because of their constructive flavor, a statement such as *It's not the case that it's not raining* is weaker than *It's raining*. The latter requires a proof of rain, whereas the former merely requires a proof that rain would not be contradictory. (This distinction also arises in natural language in the form of litotes.) Double negation introduction is a theorem of both intuitionistic logic and minimal logic, as is $\neg\neg\neg A \vdash \neg A$.

In set theory also we have the negation operation of the complement which obeys this property: a set A and a set $(A^C)^C$ (where A^C represents the complement of A) are the same.

References

- [1] Copi and Cohen
- [2] Moore and Parker
- [3] Hurley

Existential generalization

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In predicate logic, **existential generalization**^{[1][2]} (also known as **existential introduction**, $\exists I$) is a valid rule of inference that allows one to move from a specific statement to a quantified generalized statement. In first-order logic, it is often used as a rule for the existential quantifier (\exists) in formal proofs.

Example: "Rover loves to wag his tail. Therefore, something loves to wag its tail."

In the Fitch-style calculus:

$$Q(a) \rightarrow \exists x Q(x)$$

Where a replaces all free instances of x within $Q(x)$.^[3]

Quine

Universal instantiation and **Existential Generalization** are two aspects of a single principle, for instead of saying that ' $(x)(x=x)$ ' implies 'Socrates is Socrates', we could as well say that the denial 'Socrates \neq Socrates' implies ' $(\exists x)(x \neq x)$ '. The principle embodied in these two operations is the link between quantifications and the singular statements that are related to them as instances. Yet it is a principle only by courtesy. It holds only in the case where a term names and, furthermore, occurs referentially.^[4]

References

- [1] Hurley
- [2] Copi and Cohen
- [3] pg. 347. Jon Barwise and John Etchemendy, *Language proof and logic* Second Ed., CSLI Publications, 2008.
- [4] Quine,W.V.O., Quintessence, Extensionalism, Reference and Modality, P366

Existential instantiation

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In predicate logic, **existential instantiation**^{[1][2][3]} is a valid rule of inference which says that, given a formula of the form $(\exists x)\phi(x)$, one may infer $\phi(c)$ for a new constant symbol c . The rule has the restriction that the constant c introduced by the rule must be a new name that has not occurred earlier in the proof.

In one formal notation, the rule may be denoted

$$\exists (x) \mathcal{F} x :: \mathcal{F} a,$$

where a is an arbitrary name that has not been a part of our proof thus far.

References

- [1] Hurley, Patrick. A Concise Introduction to Logic. Wadsworth Pub Co, 2008.
- [2] Copi and Cohen
- [3] Moore and Parker

Exportation (logic)

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

Exportation^{[1][2]} is a valid rule of replacement in propositional logic. The rule allows conditional statements having conjunctive antecedents to be replaced by statements having conditional consequents and vice-versa in logical proofs. It is the rule that:

$$((P \wedge Q) \rightarrow R) \Leftrightarrow (P \rightarrow (Q \rightarrow R))$$

Where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Formal notation

The *exportation* rule may be written in sequent notation:

$$((P \wedge Q) \rightarrow R) \vdash (P \rightarrow (Q \rightarrow R))$$

where \vdash is a metalogical symbol meaning that $(P \rightarrow (Q \rightarrow R))$ is a syntactic consequence of $((P \wedge Q) \rightarrow R)$ in some logical system; or in rule form:

$$\frac{(P \wedge Q) \rightarrow R}{P \rightarrow (Q \rightarrow R)}.$$

where the rule is that wherever an instance of " $(P \wedge Q) \rightarrow R$ " appears on a line of a proof, it can be replaced with " $P \rightarrow (Q \rightarrow R)$ ";

or as the statement of a truth-functional tautology or theorem of propositional logic:

$$((P \wedge Q) \rightarrow R) \rightarrow (P \rightarrow (Q \rightarrow R))$$

where P , Q , and R are propositions expressed in some logical system.

Natural language

Truth values

At any time, if $P \rightarrow Q$ is true, it can be replaced by $P \rightarrow (P \wedge Q)$.

One possible case for $P \rightarrow Q$ is for P to be true and Q to be true; thus $P \wedge Q$ is also true, and $P \rightarrow (P \wedge Q)$ is true.

Another possible case sets P as false and Q as true. Thus, $P \wedge Q$ is false and $P \rightarrow (P \wedge Q)$ is false; false \rightarrow false is true.

The last case occurs when both P and Q are false. Thus, $P \wedge Q$ is false and $P \rightarrow (P \wedge Q)$ is true.

Example

It rains and the sun shines implies that there is a rainbow.

Thus, if it rains, then the sun shines implies that there is a rainbow.

Relation to functions

Exportation is associated with Currying via the Curry–Howard correspondence.

References

[1] Moore and Parker

[2] <http://www.philosophypages.com/lg/e11b.htm>

Hypothetical syllogism

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In classical logic, **hypothetical syllogism** is a valid argument form which is a syllogism having a conditional statement for one or both of its premises.^{[1][2]}

If I do not wake up, then I cannot go to work.

If I cannot go to work, then I will not get paid.

Therefore, if I do not wake up, then I will not get paid.

In propositional logic, **hypothetical syllogism** is the name of a valid rule of inference^{[3][4]} (often abbreviated **HS** and sometimes also called the **chain argument**, **chain rule**, or the principle of **transitivity of implication**). Hypothetical syllogism is one of the rules in classical logic that is not always accepted in certain systems of non-classical logic. The rule may be stated:

$$\frac{P \rightarrow Q, Q \rightarrow R}{\therefore P \rightarrow R}$$

where the rule is that whenever instances of " $P \rightarrow Q$ ", and " $Q \rightarrow R$ " appear on lines of a proof, " $P \rightarrow R$ " can be placed on a subsequent line.

Hypothetical syllogism is closely related and similar to disjunctive syllogism, in that it is also type of syllogism, and also the name of a rule of inference.

Formal notation

The *hypothetical syllogism* rule may be written in sequent notation:

$$(P \rightarrow Q), (Q \rightarrow R) \vdash (P \rightarrow R)$$

where \vdash is a metalogical symbol meaning that $P \rightarrow R$ is a syntactic consequence of $P \rightarrow Q$, and $Q \rightarrow R$ in some logical system;

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

$$((P \rightarrow Q) \wedge (Q \rightarrow R)) \rightarrow (P \rightarrow R)$$

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

External links

- Philosophy Index: Hypothetical Syllogism [5]

References

- [1] Hurley
- [2] Copi and Cohen
- [3] Hurley
- [4] Copi and Cohen
- [5] <http://www.philosophy-index.com/logic/forms/hypothetical-syllogism.php>

List of valid argument forms

Of the many and varied argument forms that can possibly be constructed, only very few are **valid argument forms**. In order to evaluate these forms, statements are put into logical form. Logical form replaces any sentences or ideas with letters to remove any bias from content and allow one to evaluate the argument without any bias due to its subject matter.

Being a valid argument does not necessarily mean the conclusion will be true. It is valid because if the premises are true, then the conclusion has to be true. This can be proven for any valid argument form using a truth table which shows that there is no situation in which there are all true premises and a false conclusion.

Valid syllogistic forms

In syllogistic logic, there are 256 possible ways to construct categorical syllogisms using the **A**, **E**, **I**, and **O** statement forms in the square of opposition. Of the 256, only 24 are valid forms. Of the 24 valid forms, 15 are unconditionally valid, and 9 are conditionally valid.

Unconditionally valid

Figure 1	Figure 2	Figure 3	Figure 4
AAA	EAE	IAI	AEE
EAE	AEE	AII	IAI
AII	EIO	OAO	EIO
EIO	AOO	EIO	

Conditionally valid

Figure 1	Figure 2	Figure 3	Figure 4	Required condition
AAI EAO	AEO EAO		AEO	<i>S</i> exists
		AAI EAO	EAO	<i>M</i> exists
			AAI	<i>P</i> exists

Valid propositional forms

Modus ponens

One valid argument form is known as modus ponens, not to be mistaken with modus tollens which is another valid argument form that has a like-sounding name and structure. Modus ponens (sometimes abbreviated as MP) says that if one thing is true, then another will be. It then states that the first is true. The conclusion is that the second thing is true. It is shown below in logical form.

If A, then B

A

Therefore, B

Before being put into logical form the above statement could have been something like below.

If Joe does not finish his homework, he will not go to class

Joe did not finish his homework

Therefore, Joe will not go to class

The first two statements are the premises while the third is the conclusion derived from

Modus tollens

Another form of argument is known as modus tollens (commonly abbreviated MT). In this form, you start with the same first premise as with modus ponens. However, the second part of the premise is denied, leading to the conclusion that the first part of the premise should be denied as well. It is shown below in logical form.

If A, then B

Not B

Therefore, not A.

When modus tollens is used with actual content, it looks like below.

If the Saints win the Super Bowl, there will be a party in New Orleans that night

There was no party in New Orleans that night

Therefore, the Saints did not win the Super Bowl

Hypothetical syllogism

Much like modus ponens and modus tollens, hypothetical syllogism (sometimes abbreviated as HS) contains two premises and a conclusion. It is however, slightly more complicated than the first two. In short, it states that if one thing happens, another will as well. If that second thing happens, a third will follow it. Therefore, if the first thing happens, it is inevitable that the third will too. It is shown below in logical form.

If A, then B

If B, then C

Therefore, if A, then C

When put into words it looks like below.

If it rains today, I will wear my rain jacket

If I wear my rain jacket, it will be easy for my friends to find me

Therefore, if it rains today, it will be easy for my friends to find me

This is a shortened example of what is known as a slippery slope. A slippery slope is the idea that if one single event happens, it will inevitably cause a whole list of other things to happen with no way to stop them.

Disjunctive syllogism

Disjunctive syllogism (sometimes abbreviated DS) has one of the same characteristics as modus tollens in that it contains a premise, then in a second premise it denies a statement, leading to the conclusion. In Disjunctive Syllogism, the first premise establishes two options. The second takes one away, so the conclusion states that the remaining one must be true. It is shown below in logical form.

A or B

Not A

Therefore, B

When used A and B are replaced with real life examples it looks like below.

Either you will see Joe in class today or he will oversleep

You did not see Joe in class today

Therefore, Joe overslept

Disjunctive syllogism takes two options and narrows it down to one.

Constructive dilemma

Another valid form of argument is known as constructive dilemma or sometimes just "dilemma". It does not leave the user with one statement alone at the end of the argument, instead it gives an option of two different statements. The first premise gives an option of two different statements. Then it states that if the first one happens, there will be a particular outcome and if the second happens, there will be a separate outcome. The conclusion is that either the first outcome or the second outcome will happen. The criticism with this form is that it does not give a definitive conclusion; just a statement of possibilities. When it is written in argument for it looks like below.

A or B

If A then C

If B then D

Therefore C or D

When content is inserted in place of the letters, it looks like below.

Bill will either take the stairs or the elevator to his room

If he takes the stairs, he will be tired when he gets to his room

If he takes the elevator, he will miss the start of the football game on TV

Therefore, Bill will either be tired when he gets to his room or he will miss the start of the football game

There is a slightly different version of dilemma that uses negation rather than affirming something known as destructive dilemma. When put in argument form it looks like below.

If A then C

If B then D

Not C or not D

Therefore not A or not B

References

Material implication (rule of inference)

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In propositional logic, **material implication** is a valid rule of replacement that allows for a conditional statement to be replaced by a disjunction if and only if the antecedent is negated. The rule states that P implies Q is logically equivalent to *not-P or Q* and can replace each other in logical proofs.

$$P \rightarrow Q \Leftrightarrow \neg P \vee Q$$

Where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Formal notation

The *material implication* rule may be written in sequent notation:

$$(P \rightarrow Q) \vdash (\neg P \vee Q)$$

where \vdash is a metalogical symbol meaning that $(\neg P \vee Q)$ is a syntactic consequence of $(P \rightarrow Q)$ in some logical system;
or in rule form:

$$\frac{P \rightarrow Q}{\neg P \vee Q}$$

where the rule is that wherever an instance of " $P \rightarrow Q$ " appears on a line of a proof, it can be replaced with " $\neg P \vee Q$ ";

or as the statement of a truth-functional tautology or theorem of propositional logic:

$$(P \rightarrow Q) \rightarrow (\neg P \vee Q)$$

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

Example

If it is a bear, then it can swim.

Thus, it is not a bear or it can swim.

where P is the statement "it is a bear" and Q is the statement "it can swim".

If it was found that the bear could not swim, written symbolically as $P \wedge \neg Q$, then both sentences are false but otherwise they are both true.

References

Modus non excipiens

In logic, **modus non excipiens**^{[1][2]} is a valid rule of inference that is closely related to modus ponens. This argument form was created by Bart Verheij to address certain arguments which are types of *modus ponens* arguments, but must be considered to be invalid. An instance of a particular *modus ponens* type argument is

A large majority accept A as true. Therefore, there exists a presumption in favor of A.

However, this is an *argumentum ad populum*, and is not deductively valid. The problem can be addressed by drawing a distinction between two types of inference identified by Verheij:

Modus ponens:

Premises:

- As a rule, if P then Q
- P

Conclusion:

- Q

and

Modus non excipiens

Premises:

- As a rule, if P then Q
- P
- It is not the case that there is an exception to the rule that if P then Q

Conclusion:

- Q

[1] Bart Verheij, 'Logic, Context and Valid Inference Or: Can There be a Logic of Law', 2000. Available on bart.verheij@metajur.unimaas.nl, (<http://www.metajur.unimaas.nl/~bart/>)

[2] Walton, Douglas; *Are Some Modus Ponens Arguments Deductively Invalid?*, University of Winnipeg

Modus ponendo tollens

Modus ponendo tollens (Latin: "mode that by affirming, denies")^[1] is a valid rule of inference for propositional logic, sometimes abbreviated **MPT**.^[2] It is closely related to *modus ponens* and *modus tollens*. It is usually described as having the form:

1. Not both A and B
2. A
3. Therefore, not B

For example:

1. Ann and Bill cannot both win the race.
2. Ann won the race.
3. Therefore, Bill cannot have won the race.

As E.J. Lemmon describes it: "*Modus ponendo tollens* is the principle that, if the negation of a conjunction holds and also one of its conjuncts, then the negation of its other conjunct holds."^[3]

In logic notation this can be represented as:

1. $\neg(A \wedge B)$
2. A
3. $\therefore \neg B$

References

- [1] Stone, Jon R. 1996. *Latin for the Illiterati: Exorcizing the Ghosts of a Dead Language*. London, UK: Routledge:60.
- [2] Politzer, Guy & Carles, Laure. 2001. 'Belief Revision and Uncertain Reasoning'. *Thinking and Reasoning*. 7:217-234.
- [3] Lemmon, Edward John. 2001. *Beginning Logic*. Taylor and Francis/CRC Press: 61.

Modus ponens

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In propositional logic, ***modus ponendo ponens*** (Latin for "the way that affirms by affirming"; often abbreviated to **MP** or ***modus ponens***^{[1][2][3]}) or **implication elimination** is a valid, simple argument form and rule of inference.^[4] It can be summarized as "*P* implies *Q*; *P* is asserted to be true, so therefore *Q* must be true." The history of ***modus ponens*** goes back to antiquity.^[5]

While ***modus ponens*** is one of the most commonly used concepts in logic it must not be mistaken for a logical law; rather, it is one of the accepted mechanisms for the construction of deductive proofs that includes the "rule of definition" and the "rule of substitution".^[6] ***Modus ponens*** allows one to eliminate a conditional statement from a logical proof or argument (the antecedents) and thereby not carry these antecedents forward in an ever-lengthening string of symbols; for this reason ***modus ponens*** is sometimes called the **rule of detachment**.^[7] Enderton, for example, observes that "***modus ponens*** can produce shorter formulas from longer ones",^[8] and Russell observes that "the process of the inference cannot be reduced to symbols. Its sole record is the occurrence of $\vdash q$ [the consequent]. . . an inference is the dropping of a true premise; it is the dissolution of an implication".^[9]

A justification for the "trust in inference is the belief that if the two former assertions [the antecedents] are not in error, the final assertion [the consequent] is not in error".^[10] In other words: if one statement or proposition implies a

second one, and the first statement or proposition is true, then the second one is also true. If P implies Q and P is true, then Q is true. An example is:

If it's raining, I'll meet you at the movie theater.

It's raining.

Therefore, I'll meet you at the movie theater.

Modus ponens can be stated formally as:

$$\frac{P \rightarrow Q, P}{\therefore Q}$$

where the rule is that whenever an instance of " $P \rightarrow Q$ " and " P " appear by themselves on lines of a logical proof, Q can validly be placed on a subsequent line; furthermore, the premise P and the implication "dissolves", their only trace being the symbol Q that is retained for use later e.g. in a more complex deduction.

It is closely related to another valid form of argument, *modus tollens*. Both have apparently similar but invalid forms such as affirming the consequent, denying the antecedent, and evidence of absence. Constructive dilemma is the disjunctive version of modus ponens. Hypothetical syllogism is closely related to modus ponens and sometimes thought of as "double modus ponens."

Formal notation

The *modus ponens* rule may be written in sequent notation:

$$P \rightarrow Q, P \vdash Q$$

where \vdash is a metalogical symbol meaning that Q is a syntactic consequence of $P \rightarrow Q$ and P in some logical system; or as the statement of a truth-functional tautology or theorem of propositional logic:

$$((P \rightarrow Q) \wedge P) \rightarrow Q$$

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

Explanation

The argument form has two premises (hypothesis). The first premise is the "if–then" or conditional claim, namely that P implies Q . The second premise is that P , the antecedent of the conditional claim, is true. From these two premises it can be logically concluded that Q , the consequent of the conditional claim, must be true as well. In artificial intelligence, *modus ponens* is often called forward chaining.

An example of an argument that fits the form *modus ponens*:

If today is Tuesday, then John will go to work.

Today is Tuesday.

Therefore, John will go to work.

This argument is valid, but this has no bearing on whether any of the statements in the argument are true; for *modus ponens* to be a sound argument, the premises must be true for any true instances of the conclusion. An argument can be valid but nonetheless unsound if one or more premises are false; if an argument is valid *and* all the premises are true, then the argument is sound. For example, John might be going to work on Wednesday. In this case, the reasoning for John's going to work (because it is Wednesday) is unsound. The argument is not only sound on Tuesdays (when John goes to work), but valid on every day of the week. A propositional argument using *modus ponens* is said to be deductive.

In single-conclusion sequent calculi, *modus ponens* is the Cut rule. The cut-elimination theorem for a calculus says that every proof involving Cut can be transformed (generally, by a constructive method) into a proof without Cut, and hence that Cut is admissible.

The Curry–Howard correspondence between proofs and programs relates *modus ponens* to function application: if f is a function of type $P \rightarrow Q$ and x is of type P , then $f x$ is of type Q .

Relation to Modus Tollens

Any Modus Ponens rule can be proved using a Modus Tollens rule and transposition.

The proof is as follows.

1. $P \rightarrow Q$
2. $P /:\! Q$
3. $\sim Q \rightarrow \sim P$ 1 Transposition
4. $\sim\sim P$ 2 Double Negation
5. $\sim\sim Q$ 3,4 Modus Tollens
6. 5 Double Negation

Justification via truth table

The validity of *modus ponens* in classical two-valued logic can be clearly demonstrated by use of a truth table.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

In instances of *modus ponens* we assume as premises that $p \rightarrow q$ is true and p is true. Only one line of the truth table—the first—satisfies these two conditions (p and $p \rightarrow q$). On this line, q is also true. Therefore, whenever $p \rightarrow q$ is true and p is true, q must also be true.

References

- [1] Copi and Cohen
- [2] Hurley
- [3] Moore and Parker
- [4] Enderton 2001:110
- [5] Susanne Bobzien (2002). The Development of Modus Ponens in Antiquity, *Phronesis* 47.
- [6] Alfred Tarski 1946:47. Also Enderton 2001:110ff.
- [7] Tarski 1946:47
- [8] Enderton 2001:111
- [9] Whitehead and Russell 1927:9
- [10] Whitehead and Russell 1927:9

Sources

- Alfred Tarski 1946 *Introduction to Logic and to the Methodology of the Deductive Sciences* 2nd Edition, reprinted by Dover Publications, Mineola NY. ISBN 0-486-28462-X (pbk).
- Alfred North Whitehead and Bertrand Russell 1927 *Principia Mathematica to *56 (Second Edition)* paperback edition 1962, Cambridge at the University Press, London UK. No ISBN, no LCCCN.
- Herbert B. Enderton, 2001, *A Mathematical Introduction to Logic Second Edition*, Harcourt Academic Press, Burlington MA, ISBN 978-0-12-238452-3.

External links

- Hazewinkel, Michiel, ed. (2001), "Modus ponens" (<http://www.encyclopediaofmath.org/index.php?title=p/m064570>), *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- Modus ponens (http://philpapers.org/s/modus_ponens) at PhilPapers
- Modus ponens* (<http://mathworld.wolfram.com/ModusPonens.html>) at Wolfram MathWorld

Modus tollens

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • 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

In propositional logic, ***modus tollens***^{[1][2][3][4]} (or ***modus tollendo tollens*** and also **denying the consequent**)^[5] (Latin for "the way that denies by denying")^[6] is a valid argument form and a rule of inference.

The first to explicitly state the argument form *modus tollens* were the Stoics.^[7]

The inference rule *modus tollens*, also known as the **law of contrapositive**, validates the inference from P implies Q and the contradictory of Q , to the contradictory of P .

The *modus tollens* rule can be stated formally as:

$$\frac{P \rightarrow Q, \neg Q}{\therefore \neg P}$$

where $P \rightarrow Q$ stands for " P implies Q ", $\neg Q$ stands for "it is not the case that Q " (or in brief "not Q "). Then, whenever " $P \rightarrow Q$ " and " $\neg Q$ " each appear by themselves as a line of a proof, " $\neg P$ " can validly be placed on a subsequent line. The history of the inference rule *modus tollens* goes back to antiquity.^[8]

Modus tollens is closely related to *modus ponens*. There are two similar, but invalid, argument forms known as affirming the consequent and denying the antecedent.

Formal notation

The *modus tollens* rule may be written in sequent notation:

$$P \rightarrow Q, \neg Q \vdash \neg P$$

where \vdash is a metalogical symbol meaning that $\neg P$ is a syntactic consequence of $P \rightarrow Q$ and $\neg Q$ in some logical system;

or as the statement of a functional tautology or theorem of propositional logic:

$$((P \rightarrow Q) \wedge \neg Q) \rightarrow \neg P$$

where P , and Q are propositions expressed in some logical system;

or including assumptions:

$$\frac{\Gamma \vdash P \rightarrow Q \quad \Gamma \vdash \neg Q}{\Gamma \vdash \neg P}$$

though since the rule does not change the set of assumptions, this is not strictly necessary.

More complex rewritings involving *modus tollens* are often seen, for instance in set theory:

$$\begin{aligned} & P \subseteq Q \\ & x \notin Q \\ & \therefore x \notin P \end{aligned}$$

(" P is a subset of Q . x is not in Q . Therefore, x is not in P ."

Also in first-order predicate logic:

$$\begin{aligned} & \forall x : P(x) \rightarrow Q(x) \\ & \exists x : \neg Q(x) \\ & \therefore \exists x : \neg P(x) \end{aligned}$$

("For all x if x is P then x is Q . There exists some x that is not Q . Therefore, there exists some x that is not P ."

Strictly speaking these are not instances of *modus tollens*, but they may be derived using *modus tollens* using a few extra steps.

Explanation

The argument has two premises. The first premise is a conditional or "if-then" statement, for example that if P then Q. The second premise is that it is not the case that Q . From these two premises, it can be logically concluded that it is not the case that P.

Consider an example:

If the watch-dog detects an intruder, the watch-dog will bark.

The watch-dog did not bark

Therefore, no intruder was detected by the watch-dog.

Supposing that the premises are both true (the dog will bark if it detects an intruder, and does indeed not bark), it follows that no intruder has been detected. This is a valid argument since it is not possible for the conclusion to be false if the premises are true. (It is conceivable that there may have been an intruder that the dog did not detect, but that does not invalidate the argument; the first premise goes "if the watch-dog **detects** an intruder." The thing of importance is that the dog detects or doesn't detect an intruder, not if there is one.)

Another example:

If I am the axe murderer, then I can use an axe.

I cannot use an axe.

Therefore, I am not the axe murderer.

Relation to *modus ponens*

Every use of *modus tollens* can be converted to a use of *modus ponens* and one use of transposition to the premise which is a material implication. For example:

If P, then Q. (premise -- material implication)

If not Q , then not P. (derived by transposition)

Not Q . (premise)

Therefore, not P. (derived by *modus ponens*)

Likewise, every use of *modus ponens* can be converted to a use of *modus tollens* and transposition.

Justification via truth table

The validity of *modus tollens* can be clearly demonstrated through a truth table.

p	q	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

In instances of *modus tollens* we assume as premises that $p \rightarrow q$ is true and q is false. There is only one line of the truth table—the fourth line—which satisfies these two conditions. In this line, p is false. Therefore, in every instance in which $p \rightarrow q$ is true and q is false, p must also be false.

Formal Proof

Via Disjunctive syllogism

Step	Proposition	Derivation
1	$P \rightarrow Q$	Given
2	$\neg Q$	Given
3	$\neg P \vee Q$	Material implication (1)
4	$\neg P$	Disjunctive syllogism (2,3)

Via Reductio ad absurdum

Step	Proposition	Derivation
1	$P \rightarrow Q$	Given
2	$\neg Q$	Given
3	P	Assumption
4	Q	Modus ponens (1,3)
5	$Q \wedge \neg Q$	Conjunction introduction (2,4)
6	$\neg P$	Reductio ad absurdum (3,5)

Notes

- [1] University of North Carolina, Philosophy Department, Logic Glossary (<http://www.philosophy.uncc.edu/mleldrid/Logic/logiglos.html#modustollens>). Accessed date on 31 October 2007.
- [2] Copi and Cohen
- [3] Hurley
- [4] Moore and Parker
- [5] Sanford, David Hawley. 2003. *If P, Then Q: Conditionals and the Foundations of Reasoning*. London, UK: Routledge: 39 "[Modus] tollens is always an abbreviation for modus tollendo tollens, the mood that by denying denies."
- [6] Stone, Jon R. 1996. *Latin for the Illiterati: Exorcizing the Ghosts of a Dead Language*. London, UK: Routledge: 60.
- [7] "Stanford Encyclopedia of Philosophy: Ancient Logic: The Stoics" (<http://plato.stanford.edu/entries/logic-ancient/#Sto>)
- [8] Susanne Bobzien (2002). "The Development of Modus Ponens in Antiquity" (<http://dx.doi.org/10.1163/156852802321016541>), *Phronesis* 47.

External links

- *Modus Tollens* (<http://mathworld.wolfram.com/ModusTollens.html>) at Wolfram MathWorld

Negation as failure

Negation as failure (NAF), for short) is a non-monotonic inference rule in logic programming, used to derive $\text{not } p$ (i.e. that p is assumed not to hold) from failure to derive p . Note that $\text{not } p$ can be different from the statement $\neg p$ of the logical negation of p , depending on the completeness of the inference algorithm and thus also on the formal logic system.

Negation as failure has been an important feature of logic programming since the earliest days of both Planner and Prolog. In Prolog, it is usually implemented using Prolog's extralogical constructs.

Planner semantics

In Planner, negation as failure could be implemented as follows:

```
if (not (goal p)), then (assert \neg p)
```

which says that if the goal to prove p fails, then assert $\neg p$. Note that the above example uses true mathematical negation, which cannot be expressed in Prolog.

Prolog semantics

In pure Prolog, NAF literals of the form $\text{not } p$ can occur in the body of clauses and can be used to derive other NAF literals. For example, given only the four clauses

$$p \leftarrow q \wedge \text{not } r$$

$$q \leftarrow s$$

$$q \leftarrow t$$

$$t \leftarrow$$

NAF derives $\text{not } s$, $\text{not } r$ and p .

Completion semantics

The semantics of NAF remained an open issue until Keith Clark [1978] showed that it is correct with respect to the completion of the logic program, where, loosely speaking, "only" and \leftarrow are interpreted as "if and only if", written as "iff" or " \equiv ".

For example, the completion of the four clauses above is

$$p \equiv q \wedge \text{not } r$$

$$q \equiv s \vee t$$

$$t \equiv \text{true}$$

$$r \equiv \text{false}$$

$$s \equiv \text{false}$$

The NAF inference rule simulates reasoning explicitly with the completion, where both sides of the equivalence are negated and negation on the right-hand side is distributed down to atomic formulae. For example, to show $\text{not } p$, NAF simulates reasoning with the equivalences

$$\text{not } p \equiv \text{not } q \vee r$$

$$\text{not } q \equiv \text{not } s \wedge \text{not } t$$

$$\text{not } t \equiv \text{false}$$

$$\text{not } r \equiv \text{true}$$

$\text{not } s \equiv \text{true}$

In the non-propositional case, the completion needs to be augmented with equality axioms, to formalise the assumption that individuals with distinct names are distinct. NAF simulates this by failure of unification. For example, given only the two clauses

$$\begin{aligned} p(a) &\leftarrow \\ p(b) &\leftarrow t \end{aligned}$$

NAF derives $\text{not } p(c)$.

The completion of the program is

$$p(X) \equiv X = a \vee X = b$$

augmented with unique names axioms and domain closure axioms.

The completion semantics is closely related both to circumscription and to the closed world assumption.

Autoepistemic semantics

The completion semantics justifies interpreting the result $\text{not } p$ of a NAF inference as the classical negation $\neg p$ of p . However, Michael Gelfond [1987] showed that it is also possible to interpret $\text{not } p$ literally as " p can not be shown", " p is not known" or " p is not believed", as in autoepistemic logic. The autoepistemic interpretation was developed further by Gelfond and Lifschitz [1988] and is the basis of answer set programming.

The autoepistemic semantics of a pure Prolog program P with NAF literals is obtained by "expanding" P with a set of ground (variable-free) NAF literals Δ that is stable in the sense that

$$\Delta = \{ \text{not } p \mid p \text{ is not implied by } P \cup \Delta \}$$

In other words, a set of assumptions Δ about what can not be shown is stable if and only if Δ is the set of all sentences that truly can not be shown from the program P expanded by Δ . Here, because of the simple syntax of pure Prolog programs, "implied by" can be understood very simply as derivability using modus ponens and universal instantiation alone.

A program can have zero, one or more stable expansions. For example

$$p \leftarrow \text{not } p$$

has no stable expansions.

$$p \leftarrow \text{not } q$$

has exactly one stable expansion $\Delta = \{ \text{not } q \}$

$$p \leftarrow \text{not } q$$

$$q \leftarrow \text{not } p$$

has exactly two stable expansions $\Delta_1 = \{ \text{not } p \}$ and $\Delta_2 = \{ \text{not } q \}$.

The autoepistemic interpretation of NAF can be combined with classical negation, as in extended logic programming and answer set programming. Combining the two negations, it is possible to express, for example

$\neg p \leftarrow \text{not } p$ (the closed world assumption) and

$p \leftarrow \text{not } \neg p$ (p holds by default).

References

- K. Clark [1978, 1987]. Negation as failure^[1]. *Readings in nonmonotonic reasoning*, Morgan Kaufmann Publishers, pages 311-325.
- M. Gelfond [1987] On Stratified Autoepistemic Theories^[2] Proc. AAAI, pages 207-211.
- M. Gelfond and V. Lifschitz [1988] The Stable Model Semantics for Logic Programming^[3] Proc. 5th International Conference and Symposium on Logic Programming (R. Kowalski and K. Bowen, eds), MIT Press, pages 1070-1080.
- J.C. Shepherdson [1984] *Negation as failure: a comparison of Clark's completed data base and Reiter's closed world assumption*, Journal of Logic Programming, vol 1, 1984, pages 51-81.
- J.C. Shepherdson [1985] *Negation as failure II*, Journal of Logic Programming, vol 3, 1985, pages 185-202.

External links

- Report^[4] from the W3C Workshop on Rule Languages for Interoperability. Includes notes on NAF and SNAF (scoped negation as failure).

References

- [1] <http://www.doc.ic.ac.uk/~klc/neg.html>
[2] <http://www.cs.ttu.edu/~mgelfond/papers/autoepistemic.pdf>
[3] <http://www.cs.ttu.edu/~mgelfond/papers/stable.pdf>
[4] <http://www.w3.org/2004/12/rules-ws/report/>

Resolution (logic)

In mathematical logic and automated theorem proving, **resolution** is a rule of inference leading to a refutation theorem-proving technique for sentences in propositional logic and first-order logic. In other words, iteratively applying the resolution rule in a suitable way allows for telling whether a propositional formula is satisfiable and for proving that a first-order formula is unsatisfiable; this method may prove the satisfiability of a first-order satisfiable formula, but not always, as it is the case for all methods for first-order logic (see Gödel's incompleteness theorems and Halting problem). Resolution was introduced by John Alan Robinson in 1965.

The clause produced by a resolution rule is sometimes called a **resolvent**.

Resolution in propositional logic

Resolution rule

The **resolution rule** in propositional logic is a single valid inference rule that produces a new clause implied by two clauses containing complementary literals. A literal is a propositional variable or the negation of a propositional variable. Two literals are said to be complements if one is the negation of the other (in the following, a_i is taken to be the complement to b_j). The resulting clause contains all the literals that do not have complements. Formally:

$$\frac{a_1 \vee \dots \vee a_i \vee \dots \vee a_n, \quad b_1 \vee \dots \vee b_j \vee \dots \vee b_m}{a_1 \vee \dots \vee a_{i-1} \vee a_{i+1} \vee \dots \vee a_n \vee b_1 \vee \dots \vee b_{j-1} \vee b_{j+1} \vee \dots \vee b_m}$$

where

all a s and b s are literals,

the dividing line stands for entails

The clause produced by the resolution rule is called the *resolvent* of the two input clauses. It is the principle of *consensus* applied to clauses rather than terms.^[1]

When the two clauses contain more than one pair of complementary literals, the resolution rule can be applied (independently) for each such pair; however, the result is always a tautology.

Modus ponens can be seen as a special case of resolution of a one-literal clause and a two-literal clause.

A resolution technique

When coupled with a complete search algorithm, the resolution rule yields a sound and complete algorithm for deciding the *satisfiability* of a propositional formula, and, by extension, the validity of a sentence under a set of axioms.

This resolution technique uses proof by contradiction and is based on the fact that any sentence in propositional logic can be transformed into an equivalent sentence in conjunctive normal form.^[citation needed] The steps are as follows.

- All sentences in the knowledge base and the *negation* of the sentence to be proved (the *conjecture*) are conjunctively connected.
- The resulting sentence is transformed into a conjunctive normal form with the conjuncts viewed as elements in a set, S , of clauses.
 - For example $(A_1 \vee A_2) \wedge (B_1 \vee B_2 \vee B_3) \wedge (C_1)$ gives rise to the set $S = \{A_1 \vee A_2, B_1 \vee B_2 \vee B_3, C_1\}$.
- The resolution rule is applied to all possible pairs of clauses that contain complementary literals. After each application of the resolution rule, the resulting sentence is simplified by removing repeated literals. If the sentence contains complementary literals, it is discarded (as a tautology). If not, and if it is not yet present in the clause set S , it is added to S , and is considered for further resolution inferences.
- If after applying a resolution rule the *empty clause* is derived, the original formula is unsatisfiable (or *contradictory*), and hence it can be concluded that the initial conjecture follows from the axioms.
- If, on the other hand, the empty clause cannot be derived, and the resolution rule cannot be applied to derive any more new clauses, the conjecture is not a theorem of the original knowledge base.

One instance of this algorithm is the original Davis–Putnam algorithm that was later refined into the DPLL algorithm that removed the need for explicit representation of the resolvents.

This description of the resolution technique uses a set S as the underlying data-structure to represent resolution derivations. Lists, Trees and Directed Acyclic Graphs are other possible and common alternatives. Tree representations are more faithful to the fact that the resolution rule is binary. Together with a sequent notation for clauses, a tree representation also makes it clear to see how the resolution rule is related to a special case of the cut-rule, restricted to atomic cut-formulas. However, tree representations are not as compact as set or list representations, because they explicitly show redundant subderivations of clauses that are used more than once in the derivation of the empty clause. Graph representations can be as compact in the number of clauses as list representations and they also store structural information regarding which clauses were resolved to derive each resolvent.

A simple example

$$\frac{a \vee b, \quad \neg a \vee c}{b \vee c}$$

In plain language: Suppose a is false. In order for the premise $a \vee b$ to be true, b must be true. Alternatively, suppose a is true. In order for the premise $\neg a \vee c$ to be true, c must be true. Therefore regardless of falsehood or veracity of a , if both premises hold, then the conclusion $b \vee c$ is true.

Resolution in first order logic

In first order logic, resolution condenses the traditional syllogisms of logical inference down to a single rule.

To understand how resolution works, consider the following example syllogism of term logic:

All Greeks are Europeans.

Homer is a Greek.

Therefore, Homer is a European.

Or, more generally:

$$\forall x.P(x) \Rightarrow Q(x)$$

$$P(a)$$

Therefore, $Q(a)$

To recast the reasoning using the resolution technique, first the clauses must be converted to conjunctive normal form (CNF). In this form, all quantification becomes implicit: universal quantifiers on variables (X, Y, \dots) are simply omitted as understood, while existentially-quantified variables are replaced by Skolem functions.

$$\neg P(x) \vee Q(x)$$

$$P(a)$$

Therefore, $Q(a)$

So the question is, how does the resolution technique derive the last clause from the first two? The rule is simple:

- Find two clauses containing the same predicate, where it is negated in one clause but not in the other.
- Perform a unification on the two predicates. (If the unification fails, you made a bad choice of predicates. Go back to the previous step and try again.)
- If any unbound variables which were bound in the unified predicates also occur in other predicates in the two clauses, replace them with their bound values (terms) there as well.
- Discard the unified predicates, and combine the remaining ones from the two clauses into a new clause, also joined by the " \vee " operator.

To apply this rule to the above example, we find the predicate P occurs in negated form

$$\neg P(X)$$

in the first clause, and in non-negated form

$$P(a)$$

in the second clause. X is an unbound variable, while a is a bound value (term). Unifying the two produces the substitution

$$X \mapsto a$$

Discarding the unified predicates, and applying this substitution to the remaining predicates (just $Q(X)$, in this case), produces the conclusion:

$$Q(a)$$

For another example, consider the syllogistic form

All Cretans are islanders.

All islanders are liars.

Therefore all Cretans are liars.

Or more generally,

$$\forall X P(X) \rightarrow Q(X)$$

$$\forall X Q(X) \rightarrow R(X)$$

$$\text{Therefore, } \forall X P(X) \rightarrow R(X)$$

In CNF, the antecedents become:

$$\neg P(X) \vee Q(X)$$

$$\neg Q(Y) \vee R(Y)$$

(Note that the variable in the second clause was renamed to make it clear that variables in different clauses are distinct.)

Now, unifying $Q(X)$ in the first clause with $\neg Q(Y)$ in the second clause means that X and Y become the same variable anyway. Substituting this into the remaining clauses and combining them gives the conclusion:

$$\neg P(X) \vee R(X)$$

The resolution rule, as defined by Robinson, also incorporated factoring, which unifies two literals in the same clause, before or during the application of resolution as defined above. The resulting inference rule is refutation-complete,^[2] in that a set of clauses is unsatisfiable if and only if there exists a derivation of the empty clause using resolution alone.

Implementations

- CARINE
- Gandalf
- Otter
- Prover9
- SNARK
- SPASS
- Vampire

Notes

[1] D.E. Knuth, *The Art of Computer Programming 4A: Combinatorial Algorithms*, part 1, p. 539

[2] Russell, Norvig (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Prentice Hall. p. 350.

References

- Robinson, J. Alan (1965). "A Machine-Oriented Logic Based on the Resolution Principle". *Journal of the ACM (JACM)* **12** (1): 23–41.
- Leitsch, Alexander (1997). *The Resolution Calculus*. Springer.
- Gallier, Jean H. (1986). *Logic for Computer Science: Foundations of Automatic Theorem Proving* (<http://www.cis.upenn.edu/~jean/gbooks/logic.html>). Harper & Row Publishers.
- Lee, Chin-Liang Chang, Richard Char-Tung (1987). *Symbolic logic and mechanical theorem proving* ([reprint] ed.). San Diego: Academic Press. ISBN 0-12-170350-9.

Approaches to **non-clausal resolution**, i.e. resolution of first-order formulas that need not be in clausal normal form, are presented in:

- D. Wilkins (1973). *QUEST — A Non-Clausal Theorem Proving System* (Master's Thesis). Univ. of Essex, England.
- Woodrow Wilson Bledsoe (1977). "Non-Resolution Theorem Proving". *Artificial Intelligence Journal* **9**: 1–35.
- N. Murray (1978). *A Proof Procedure for Non-Clausal First-Order Logic*. Syracuse Univ., Syracuse, N.Y..
- Zohar Manna, Richard Waldinger (Jan 1980). "A Deductive Approach to Program Synthesis". *ACM Transactions on Programming Languages and Systems* **2**: 90–121.
- N. V. Murray (1982). "Completely Non-Clausal Theorem Proving". *Artificial Intelligence* **18**: 67–85.
- Schmerl, U.R. (1988). "Resolution on Formula-Trees". *Acta Informatica* **25**: 425–438.

External links

- Alex Sakharov, " Resolution Principle (<http://mathworld.wolfram.com/ResolutionPrinciple.html>)", *MathWorld*.
- Alex Sakharov, " Resolution (<http://mathworld.wolfram.com/Resolution.html>)", *MathWorld*.

SLD resolution

SLD resolution (*Selective Linear Definite* clause resolution) is the basic inference rule used in logic programming. It is a refinement of resolution, which is both sound and refutation complete for Horn clauses.

The SLD inference rule

Given a goal clause:

$$\neg L_1 \vee \cdots \vee \neg L_i \vee \cdots \vee \neg L_n$$

with selected literal $\neg L_i$, and an input definite clause:

$$L \vee \neg K_1 \vee \cdots \vee \neg K_m$$

whose positive literal (atom) L unifies with the atom L_i of the selected literal $\neg L_i$, SLD resolution derives another goal clause, in which the selected literal is replaced by the negative literals of the input clause and the unifying substitution θ is applied:

$$(\neg L_1 \vee \cdots \vee \neg K_1 \vee \cdots \vee \neg K_m \vee \cdots \vee \neg L_n)\theta$$

In the simplest case, in propositional logic, the atoms L_i and L are identical, and the unifying substitution θ is vacuous. However, in the more general case, the unifying substitution is necessary to make the two literals identical.

The origin of the name "SLD"

The name "SLD resolution" was given by Maarten van Emden for the unnamed inference rule introduced by Robert Kowalski.^[1] Its name is derived from SL resolution,^[2] which is both sound and refutation complete for the unrestricted clausal form of logic. "SLD" stands for "SL resolution with Definite clauses".

In both, SL and SLD, "L" stands for the fact that a resolution proof can be restricted to a linear sequence of clauses:

$$C_1, C_2, \dots, C_l$$

where the "top clause" C_1 is an input clause, and every other clause C_{i+1} is a resolvent one of whose parents is the previous clause C_i . The proof is a refutation if the last clause C_l is the empty clause.

In SLD, all of the clauses in the sequence are goal clauses, and the other parent is an input clause. In SL resolution, the other parent is either an input clause or an ancestor clause earlier in the sequence.

In both SL and SLD, "S" stands for the fact that the only literal resolved upon in any clause C_i is one that is uniquely selected by a selection rule or selection function. In SL resolution, the selected literal is restricted to one

which has been most recently introduced into the clause. In the simplest case, such a last-in-first-out selection function can be specified by the order in which literals are written, as in Prolog. However, the selection function in SLD resolution is more general than in SL resolution and in Prolog. There is no restriction on the literal that can be selected.

The computational interpretation of SLD resolution

In clausal logic, an SLD refutation demonstrates that the input set of clauses is unsatisfiable. In logic programming, however, an SLD refutation also has a computational interpretation. The top clause $\neg L_1 \vee \dots \vee \neg L_i \vee \dots \vee \neg L_n$ can be interpreted as the denial of a conjunction of subgoals $L_1 \wedge \dots \wedge L_i \wedge \dots \wedge L_n$. The derivation of clause C_{i+1} from C_i is the derivation, by means of backward reasoning, of a new set of sub-goals using an input clause as a goal-reduction procedure. The unifying substitution θ both passes input from the selected subgoal to the body of the procedure and simultaneously passes output from the head of the procedure to the remaining unselected subgoals. The empty clause is simply an empty set of subgoals, which signals that the initial conjunction of subgoals in the top clause has been solved.

SLD resolution strategies

SLD resolution implicitly defines a search tree of alternative computations, in which the initial goal clause is associated with the root of the tree. For every node in the tree and for every definite clause in the program whose positive literal unifies with the selected literal in the goal clause associated with the node, there is a child node associated with the goal clause obtained by SLD resolution.

A leaf node, which has no children, is a success node if its associated goal clause is the empty clause. It is a failure node if its associated goal clause is non-empty but its selected literal unifies with the positive literal of no input clause.

SLD resolution is non-deterministic in the sense that it does not determine the search strategy for exploring the search tree. Prolog searches the tree depth-first, one branch at a time, using backtracking when it encounters a failure node. Depth-first search is very efficient in its use of computing resources, but is incomplete if the search space contains infinite branches and the search strategy searches these in preference to finite branches: the computation does not terminate. Other search strategies, including breadth-first, best-first, and branch-and-bound search are also possible. Moreover, the search can be carried out sequentially, one node at a time, or in parallel, many nodes simultaneously.

SLD resolution is also non-deterministic in the sense, mentioned earlier, that the selection rule is not determined by the inference rule, but is determined by a separate decision procedure, which can be sensitive to the dynamics of the program execution process.

The SLD resolution search space is an or-tree, in which different branches represent alternative computations. In the case of propositional logic programs, SLD can be generalised so that the search space is an and-or tree, whose nodes are labelled by single literals, representing subgoals, and nodes are joined either by conjunction or by disjunction. In the general case, where conjoint subgoals share variables, the and-or tree representation is more complicated.

Example

Given the logic program:

$q :- p$

p

and the top-level goal:

q

the search space consists of a single branch, in which q is reduced to p which is reduced to the empty set of subgoals, signalling a successful computation. In this case, the program is so simple that there is no role for the selection function and no need for any search.

In clausal logic, the program is represented by the set of clauses:

$q \vee \neg p$

p

and top-level goal is represented by the goal clause with a single negative literal:

$\neg q$

The search space consists of the single refutation:

$\neg q, \neg p, \text{false}$

where *false* represents the empty clause.

If the following clause were added to the program:

$q :- r$

then there would be an additional branch in the search space, whose leaf node r is a failure node. In Prolog, if this clause were added to the front of the original program, then Prolog would use the order in which the clauses are written to determine the order in which the branches of the search space are investigated. Prolog would try this new branch first, fail, and then backtrack to investigate the single branch of the original program and succeed.

If the clause

$p :- p$

were now added to the program, then the search tree would contain an infinite branch. If this clause were tried first, then Prolog would go into an infinite loop and not find the successful branch.

SLDNF

SLDNF^[3] is an extension of SLD resolution to deal with negation as failure. In SLDNF, goal clauses can contain negation as failure literals, say of the form $\text{not}(p)$, which can be selected only if they contain no variables. When such a variable-free literal is selected, a subproof (or subcomputation) is attempted to determine whether there is an SLDNF refutation starting from the corresponding unnegated literal p as top clause. The selected subgoal $\text{not}(p)$ succeeds if the subproof fails, and it fails if the subproof succeeds.

References

- [1] Robert Kowalski **Predicate Logic as a Programming Language** (<http://www.doc.ic.ac.uk/~rak/papers/IFIP%2074.pdf>) Memo 70, Department of Artificial Intelligence, Edinburgh University. 1973. Also in Proceedings IFIP Congress, Stockholm, North Holland Publishing Co., 1974, pp. 569-574.
- [2] Robert Kowalski and Donald Kuehner, **Linear Resolution with Selection Function** (<http://www.doc.ic.ac.uk/~rak/papers/sl.pdf>) Artificial Intelligence, Vol. 2, 1971, pp. 227-60.
- [3] Krzysztof Apt and Maarten van Emden, Contributions to the Theory of Logic Programming, Journal of the Association for Computing Machinery. Vol. 1982 - portal.acm.org
- Jean Gallier, *SLD-Resolution and Logic Programming* (<http://www.cis.upenn.edu/~cis510/tcl/chap9.pdf>) chapter 9 of *Logic for Computer Science: Foundations of Automatic Theorem Proving* (<http://www.cis.upenn.edu/~jean/gbooks/logic.html>), 2003 online revision (free to download), originally published by Wiley, 1986

External links

- (<http://foldoc.org/?SLD+resolution>) Definition from the Free On-Line Dictionary of Computing

Simplification

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In propositional logic, **simplification**^{[1][2][3]} (equivalent to **conjunction elimination**) is a valid immediate inference, argument form and rule of inference which makes the inference that, if the conjunction A and B is true, then A is true, and B is true. The rule makes it possible to shorten longer proofs by deriving one of the conjuncts of a conjunction on a line by itself.

An example in English:

It's raining and it's pouring.

Therefore it's raining.

The rule can be expressed in formal language as:

$$\frac{P \wedge Q}{\therefore P}$$

or as

$$\frac{P \wedge Q}{\therefore Q}$$

where the rule is that whenever instances of " $P \wedge Q$ " appear on lines of a proof, either " P " or " Q " can be placed on a subsequent line by itself.

Formal notation

The *simplification* rule may be written in sequent notation:

$$(P \wedge Q) \vdash P$$

or as

$$(P \wedge Q) \vdash Q$$

where \vdash is a metalogical symbol meaning that P is a syntactic consequence of $P \wedge Q$ and Q is also a syntactic consequence of $P \wedge Q$ in logical system;

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

$$(P \wedge Q) \rightarrow P$$

and

$$(P \wedge Q) \rightarrow Q$$

where P and Q are propositions expressed in some logical system.

References

- [1] Copi and Cohen
- [2] Moore and Parker
- [3] Hurley

Structural rule

In proof theory, a **structural rule** is an inference rule that does not refer to any logical connective, but instead operates on the judgements or sequents directly. Structural rules often mimic intended meta-theoretic properties of the logic. Logics that deny one or more of the structural rules are classified as substructural logics.

Common structural rules

- **Weakening**, where the hypotheses or conclusion of a sequent may be extended with additional members. In symbolic form weakening rules can be written as $\frac{\Gamma \vdash \Sigma}{\Gamma, A \vdash \Sigma}$ on the left of the turnstile, and $\frac{\Gamma \vdash \Sigma}{\Gamma \vdash A, \Sigma}$ on the right.
- **Contraction**, where two equal (or unifiable) members on the same side of a sequent may be replaced by a single member (or common instance). Symbolically: $\frac{\Gamma, A, A \vdash \Sigma}{\Gamma, A \vdash \Sigma}$ and $\frac{\Gamma \vdash A, A, \Sigma}{\Gamma \vdash A, \Sigma}$. Also known as **factoring** in automated theorem proving systems using resolution.
- **Exchange**, where two members on the same side of a sequent may be swapped. Symbolically: $\frac{\Gamma_1, A, \Gamma_2, B, \Gamma_3 \vdash \Sigma}{\Gamma_1, B, \Gamma_2, A, \Gamma_3 \vdash \Sigma}$ and $\frac{\Gamma \vdash \Sigma_1, A, \Sigma_2, B, \Sigma_3}{\Gamma \vdash \Sigma_1, B, \Sigma_2, A, \Sigma_3}$. (This is also known as the *permutation rule*.)

A logic without any of the above structural rules would interpret the sides of a sequent as pure sequences; with exchange, they are multisets; and with both contraction and exchange they are sets.

A famous structural rule is known as **cut**. Considerable effort is spent by proof theorists in showing that cut rules are superfluous in various logics. More precisely, what is shown is that cut is only (in a sense) a tool for abbreviating proofs, and does not add to the theorems that can be proved. The successful 'removal' of cut rules, known as *cut elimination*, is directly related to the philosophy of *computation as normalization* (see Curry–Howard correspondence); it often gives a good indication of the complexity of deciding a given logic.

Tautology (rule of inference)

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In propositional logic, **tautology** is one of two commonly used rules of replacement.^{[1][2]} The rules are used to eliminate redundancy in disjunctions and conjunctions when they occur in logical proofs. They are:

The principle of **idempotency of disjunction**:

$$(P \vee P) \Leftrightarrow P$$

and the principle of **idempotency of conjunction**:

$$(P \& P) \Leftrightarrow P$$

Where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a logical proof with."

Relation to tautology

The rule gets its name from the fact that the concept of the rule is the same as the tautologous statements *If "p and p" is true then "p" is true.* and *If "p or p" is true then "p" is true.* This type of tautology is called idempotency. Although the rule is the expression of a particular tautology, this is a bit misleading, as every rule of inference can be expressed as a tautology and vice-versa.

Formal notation

Theorems are those logical formulas ϕ where $\vdash \phi$ is the conclusion of a valid proof,^[3] while the equivalent semantic consequence $\models \phi$ indicates a tautology.

The *tautology* rule may be expressed as a sequent:

$$P \vee P \vdash P$$

and

$$P \wedge P \vdash P$$

where \vdash is a metalogical symbol meaning that P is a syntactic consequence of $P \vee P$, in the one case, $P \wedge P$ in the other, in some logical system;

or as a rule of inference:

$$\frac{P \vee P}{\therefore P}$$

and

$$\frac{P \wedge P}{\therefore P}$$

where the rule is that wherever an instance of " $P \vee P$ " or " $P \wedge P$ " appears on a line of a proof, it can be replaced with " P ";

or as the statement of a truth-functional tautology or theorem of propositional logic. The principle was stated as a theorem of propositional logic by Russell and Whitehead in *Principia Mathematica* as:

$$(P \vee P) \rightarrow P$$

and

$$(P \wedge P) \rightarrow P$$

where P is a proposition expressed in some logical system.

References

- [1] Copi and Cohen
- [2] Moore and Parker
- [3] Logic in Computer Science, p. 13

Transposition (logic)

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In propositional logic, **transposition**[Wikipedia:Verifiability](#)^[1] is a valid rule of replacement that permits one to switch the antecedent with the consequent of a conditional statement in a logical proof if they are also both negated. It is the inference from the truth of "A implies B" the truth of "Not-B implies not-A", and conversely.^{[2][3]} It is very closely related to the rule of inference modus tollens. It is the rule that:

$$(P \rightarrow Q) \Leftrightarrow (\neg Q \rightarrow \neg P)$$

Where " \Leftrightarrow " is a metalogical symbol representing "can be replaced in a proof with."

Formal notation

The *transposition* rule may be expressed as a sequent:

$$(P \rightarrow Q) \vdash (\neg Q \rightarrow \neg P)$$

where \vdash is a metalogical symbol meaning that $(\neg Q \rightarrow \neg P)$ is a syntactic consequence of $(P \rightarrow Q)$ in some logical system;

or as a rule of inference:

$$\frac{P \rightarrow Q}{\therefore \neg Q \rightarrow \neg P}$$

where the rule is that wherever an instance of " $P \rightarrow Q$ " appears on a line of a proof, it can be replaced with " $\neg Q \rightarrow \neg P$ ";

or as the statement of a truth-functional tautology or theorem of propositional logic. The principle was stated as a theorem of propositional logic by Russell and Whitehead in *Principia Mathematica* as:

$$(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$$

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

Traditional logic

Form of transposition

In the inferred proposition, the consequent is the contradictory of the antecedent in the original proposition, and the antecedent of the inferred proposition is the contradictory of the consequent of the original proposition. The symbol for material implication signifies the proposition as a hypothetical, or the "if-then" form, e.g. "if P then Q ".

The biconditional statement of the rule of transposition (\leftrightarrow) refers to the relation between hypothetical (\rightarrow) *propositions*, with each proposition including an antecedent and consequential term. As a matter of logical inference, to transpose or convert the terms of one proposition requires the conversion of the terms of the propositions on both sides of the biconditional relationship. Meaning, to transpose or convert $(P \rightarrow Q)$ to $(Q \rightarrow P)$ requires that the other proposition, $(\neg Q \rightarrow \neg P)$, be transposed or converted to $(\neg P \rightarrow \neg Q)$. Otherwise, to convert the terms of one proposition and not the other renders the rule invalid, violating the sufficient condition and necessary condition of the terms of the propositions, where the violation is that the changed proposition commits the fallacy of denying the antecedent or affirming the consequent by means of illicit conversion

The truth of the rule of transposition is dependent upon the relations of sufficient condition and necessary condition in logic.

Sufficient condition

In the proposition "If P then Q ", the occurrence of ' P ' is sufficient reason for the occurrence of ' Q '. ' P ', as an individual or a class, materially implicates ' Q ', but the relation of ' Q ' to ' P ' is such that the converse proposition "If Q then P " does not necessarily have sufficient condition. The rule of inference for sufficient condition is *modus ponens*, which is an argument for conditional implication:

Premise (1): If P , then Q

Premise (2): P

Conclusion: Therefore, Q

Necessary condition

Since the converse of premise (1) is not valid, all that can be stated of the relationship of 'P' and 'Q' is that in the absence of 'Q', 'P' does not occur, meaning that 'Q' is the necessary condition for 'P'. The rule of inference for necessary condition is *modus tollens*:

Premise (1): If P, then Q

Premise (2): not Q

Conclusion: Therefore, not P

Grammatically speaking

A grammatical example traditionally used by logicians contrasting sufficient and necessary conditions is the statement "If there is fire, then oxygen is present". An oxygenated environment is necessary for fire or combustion, but simply because there is an oxygenated environment does not necessarily mean that fire or combustion is occurring. While one can infer that fire stipulates the presence of oxygen, from the presence of oxygen the converse "If there is oxygen present, then fire is present" cannot be inferred. All that can be inferred from the original proposition is that "If oxygen is not present, then there cannot be fire".

Relationship of propositions

The symbol for the biconditional (" \leftrightarrow ") signifies the relationship between the propositions is both necessary and sufficient, and is verbalized as "if and only if", or, according to the example "If P then Q 'if and only if' if not Q then not P".

Necessary and sufficient conditions can be explained by analogy in terms of the concepts and the rules of immediate inference of traditional logic. In the categorical proposition "All S is P", the subject term 'S' is said to be distributed, that is, all members of its class are exhausted in its expression. Conversely, the predicate term 'P' cannot be said to be distributed, or exhausted in its expression because it is indeterminate whether every instance of a member of 'P' as a class is also a member of 'S' as a class. All that can be validly inferred is that "Some P are S". Thus, the type 'A' proposition "All P is S" cannot be inferred by conversion from the original 'A' type proposition "All S is P". All that can be inferred is the type "A" proposition "All non-P is non-S" (Note that $(P \rightarrow Q)$ and $(\sim Q \rightarrow \sim P)$ are both 'A' type propositions). Grammatically, one cannot infer "all mortals are men" from "All men are mortal". An 'A' type proposition can only be immediately inferred by conversion when both the subject and predicate are distributed, as in the inference "All bachelors are unmarried men" from "All unmarried men are bachelors".

Transposition and the method of contraposition

In traditional logic the reasoning process of transposition as a rule of inference is applied to categorical propositions through contraposition and obversion,^[4] a series of immediate inferences where the rule of obversion is first applied to the original categorical proposition "All S is P"; yielding the obverse "No S is non-P". In the obversion of the original proposition to an 'E' type proposition, both terms become distributed. The obverse is then converted, resulting in "No non-P is S", maintaining distribution of both terms. The No non-P is S" is again obverted, resulting in the [contrapositive] "All non-P is non-S". Since nothing is said in the definition of contraposition with regard to the predicate of the inferred proposition, it is permissible that it could be the original subject or its contradictory, and the predicate term of the resulting 'A' type proposition is again undistributed. This results in two contrapositives, one where the predicate term is distributed, and another where the predicate term is undistributed.^[5]

Differences between transposition and contraposition

Note that the method of transposition and contraposition should not be confused. Contraposition is a type of immediate inference in which from a given categorical proposition another categorical proposition is inferred which has as its subject the contradictory of the original predicate. Since nothing is said in the definition of contraposition with regard to the predicate of the inferred proposition, it is permissible that it could be the original subject or its contradictory. This is in contradistinction to the form of the propositions of transposition, which may be material implication, or a hypothetical statement. The difference is that in its application to categorical propositions the result of contraposition is two contrapositives, each being the obvert of the other,^[6] i.e. "No non-P is S" and "All non-P is non-S". The distinction between the two contrapositives is absorbed and eliminated in the principle of transposition, which presupposes the "mediate inferences"^[7] of contraposition and is also referred to as the "law of contraposition".^[8]

Transposition in mathematical logic

See Transposition (mathematics), Set theory

Proof

<i>Proposition</i>	<i>Derivation</i>
$P \rightarrow Q$	Given
$\neg P \vee Q$	Material implication
$Q \vee \neg P$	Commutativity
$\neg Q \rightarrow \neg P$	Material implication

References

- [1] Moore and Parker
- [2] Brody, Bobuch A. "Glossary of Logical Terms". *Encyclopedia of Philosophy*. Vol. 5–6, p. 76. Macmillan, 1973.
- [3] Copi, Irving M. *Symbolic Logic*. 5th ed. Macmillan, 1979. See the Rules of Replacement, pp. 39-40.
- [4] Stebbing, 1961, p. 65-66. For reference to the initial step of contraposition as obversion and conversion, see Copi, 1953, p. 141.
- [5] See Stebbing, 1961, pp. 65-66. Also, for reference to the immediate inferences of obversion, conversion, and obversion again, see Copi, 1953, p. 141.
- [6] See Stebbing, 1961, p. 66.
- [7] For an explanation of the absorption of obversion and conversion as "mediate inferences" see: Copi, Irving. *Symbolic Logic*. pp. 171-174, MacMillan, 1979, fifth edition.
- [8] Prior, A.N. "Logic, Traditional". *Encyclopedia of Philosophy*, Vol.5, Macmillan, 1973.

Further reading

- Brody, Bobuch A. "Glossary of Logical Terms". *Encyclopedia of Philosophy*. Vol. 5-6, p. 61. Macmillan, 1973.
- Copi, Irving. *Introduction to Logic*. MacMillan, 1953.
- Copi, Irving. *Symbolic Logic*. MacMillan, 1979, fifth edition.
- Prior, A.N. "Logic, Traditional". *Encyclopedia of Philosophy*, Vol.5, Macmillan, 1973.
- Stebbing, Susan. *A Modern Introduction to Logic*. Harper, 1961, Seventh edition

External links

- Improper Transposition (<http://www.fallacyfiles.org/imptrans.html>) (Fallacy Files)

Universal generalization

Transformation rules	
Propositional calculus	
Rules of inference	
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption 	
Rules of replacement	
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology 	
Predicate logic	
Universal generalization	
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation 	

In predicate logic, **generalization** (also **universal generalization**,^{[1][2][3]} **GEN**) is a valid inference rule. It states that if $\vdash P(x)$ has been derived, then $\vdash \forall x P(x)$ can be derived.

Generalization with hypotheses

The full generalization rule allows for hypotheses to the left of the turnstile, but with restrictions. Assume Γ is a set of formulas, φ a formula, and $\Gamma \vdash \varphi(y)$ has been derived. The generalization rule states that $\Gamma \vdash \forall x \varphi(x)$ can be derived if y is not mentioned in Γ and x does not occur in φ .

These restrictions are necessary for soundness. Without the first restriction, one could conclude $\forall x P(x)$ from the hypothesis $P(y)$. Without the second restriction, one could make the following deduction:

1. $\exists z \exists w (z \neq w)$ (Hypothesis)
2. $\exists w (y \neq w)$ (Existential instantiation)
3. $y \neq x$ (Existential instantiation)
4. $\forall x (x \neq x)$ (Faulty universal generalization)

This purports to show that $\exists z \exists w (z \neq w) \vdash \forall x (x \neq x)$, which is an unsound deduction.

Example of a proof

Prove: $\forall x (P(x) \rightarrow Q(x)) \rightarrow (\forall x P(x) \rightarrow \forall x Q(x))$.

Proof:

Number	Formula	Justification
1	$\forall x (P(x) \rightarrow Q(x))$	Hypothesis
2	$\forall x P(x)$	Hypothesis
3	$(\forall x (P(x) \rightarrow Q(x))) \rightarrow (P(y) \rightarrow Q(y))$	Universal instantiation
4	$P(y) \rightarrow Q(y)$	From (1) and (3) by Modus ponens
5	$(\forall x P(x)) \rightarrow P(y)$	Universal instantiation
6	$P(y)$	From (2) and (5) by Modus ponens
7	$Q(y)$	From (6) and (4) by Modus ponens
8	$\forall x Q(x)$	From (7) by Generalization
9	$\forall x (P(x) \rightarrow Q(x)), \forall x P(x) \vdash \forall x Q(x)$	Summary of (1) through (8)
10	$\forall x (P(x) \rightarrow Q(x)) \vdash \forall x P(x) \rightarrow \forall x Q(x)$	From (9) by Deduction theorem
11	$\vdash \forall x (P(x) \rightarrow Q(x)) \rightarrow (\forall x P(x) \rightarrow \forall x Q(x))$	From (10) by Deduction theorem

In this proof, Universal generalization was used in step 8. The Deduction theorem was applicable in steps 10 and 11 because the formulas being moved have no free variables.

References

- [1] Copi and Cohen
- [2] Hurley
- [3] Moore and Parker

Universal instantiation

Transformation rules
Propositional calculus
Rules of inference
<ul style="list-style-type: none"> • <i>Modus ponens</i> • <i>Modus tollens</i> • Biconditional introduction • Biconditional elimination • Conjunction introduction • Simplification • Disjunction introduction • Disjunction elimination • Disjunctive syllogism • Hypothetical syllogism • Constructive dilemma • Destructive dilemma • Absorption
Rules of replacement
<ul style="list-style-type: none"> • Associativity • Commutativity • Distributivity • Double negation • De Morgan's laws • Transposition • Material implication • Material equivalence • Exportation • Tautology
Predicate logic
Universal generalization
<ul style="list-style-type: none"> • Universal instantiation • Existential generalization • Existential instantiation

In predicate logic **universal instantiation**^{[1][2][3]} (**UI**, also called **universal specification**, and sometimes confused with *Dictum de omni*) is a valid rule of inference from a truth about each member of a class of individuals to the truth about a particular individual of that class. It is generally given as a quantification rule for the universal quantifier but it can also be encoded in an axiom. It is one of the basic principles used in quantification theory.

Example: "All dogs are mammals. Fido is a dog. Therefore Fido is a mammal."

In symbols the rule as an axiom schema is

$$\forall x A(x) \Rightarrow A(a/x),$$

for some term a and where $A(a/x)$ is the result of substituting a for all occurrences of x in A .

And as a rule of inference it is

from $\vdash \forall x A$ infer $\vdash A(a/x)$,

with $A(a/x)$ the same as above.

Irving Copi noted that universal instantiation "...follows from variants of rules for 'natural deduction', which were devised independently by Gerhard Gentzen and Stanislaw Jaskowski in 1934."^[4]

Quine

Universal Instantiation and Existential generalization are two aspects of a single principle, for instead of saying that ' $(x=x)$ ' implies 'Socrates is Socrates', we could as well say that the denial 'Socrates \neq Socrates' implies ' $(\exists x(x \neq x))$ '. The principle embodied in these two operations is the link between quantifications and the singular statements that are related to them as instances. Yet it is a principle only by courtesy. It holds only in the case where a term names and, furthermore, occurs referentially.^[5]

References

- [1] Copi and Cohen
 - [2] Hurley
 - [3] Moore and Parker
 - [4] pg. 71. Symbolic Logic; 5th ed.
 - [5] Quine,W.V.O., Quintessence, Extensionalism, Reference and Modality, P366
-

Theorems in Propositional Logic

Case analysis

For the rule of inference of propositional logic which expresses Case analysis, see Disjunction elimination.

Case analysis is one of the most general and applicable methods of analytical thinking, depending only on the division of a problem, decision or situation into a sufficient number of separate cases. Analysing each such case individually may be enough to resolve the initial question. The principle of case analysis is invoked in the celebrated remark of Sherlock Holmes, to the effect that when one has eliminated the impossible, what remains must be true, however unlikely that seems.

Connections with logical principles

The logical roots of the Holmes remark speak to the principle of excluded middle. That indicates the importance to case analysis of logical disjunction: stringing together propositions with the logical connective "*or*". Medical diagnosis can indeed follow the Holmes pattern, with a patient's symptom possibly caused by a number of conditions: the patient suffers from *A* or *B* or ... or illness *I*; see differential diagnosis. Deductive logic is applied to reducing the number of cases; see case-based reasoning.

A canonical statement of case analysis in the sentential calculus is:

"If a statement *P* implies a statement *Q*, and a statement *R* also implies *Q*, and either *P* or *R* is true, then *Q* must be true."

$$(((P \rightarrow Q) \wedge (R \rightarrow Q)) \wedge (P \vee R)) \rightarrow Q$$

Exhaustive analysis

The most important issue in this style of case analysis is that the cases should be collectively *exhaustive*: everything is covered. The condition that they should be *exclusive*, while convenient, is not to be assumed lightly; for example a patient's liver problem might be caused by hepatitis *and* abuse of alcohol, with one factor not ruling out the other. This points up the distinction between exclusive or, and logical disjunction which is the default meaning of 'or' (in logic, mathematics and science) and which is non-exclusive. Case analysis of the non-overlapping kind is a special case, only.

That being said, in computer programming, case analysis presents itself in a form best adapted to exclusive cases. In simple terms, the requirement is to have a list of actions, so that 'if *X* = 1 do *P*, if *X* = 2 do *Q*, if *X* = 3 do *R*' can be given as quite unambiguous instructions. The value of *X* here is computed according to what case one is in.

Other terminology

Case-by-case analysis is a more specific term for such a pinning-down of cases. It assumes a situation in which a thorough-going case analysis can be completed: all cases covered and resolved. This is not always realistic. Other forms of case analysis are best case analysis and worst case analysis, scenarios for the optimist and pessimist, respectively.

Two names for approaches that take complete case-by-case analysis as not meeting the needs of the topic under consideration are casuistry, most often in ethics, and the case study method used in business schools.

Consequentia mirabilis

Consequentia mirabilis (Latin for "admirable consequence"), also known as **Clavius's Law**, is used in traditional and classical logic to establish the truth of a proposition from the inconsistency of its negation.^[1] It is thus similar to *reductio ad absurdum*, but it can prove a proposition true using just its negation. It states that if a proposition is a consequence of its negation, then it is true, for consistency. It can thus be demonstrated without using any other principle, but that of consistency.

In formal notation: $(\neg A \rightarrow A) \rightarrow A$ which is equivalent to $(\neg\neg A \vee A) \rightarrow A$.

Consequentia mirabilis was a pattern of argument popular in 17th century Europe that first appeared in a fragment of Aristotle's *Protrepticus*: "If we ought to philosophise, then we ought to philosophise; and if we ought not to philosophise, then we ought to philosophise (i.e. in order to justify this view); in any case, therefore, we ought to philosophise."

The most famous example is perhaps the Cartesian *cogito ergo sum*: Even if one can question the validity of the thinking, no one can deny that they are thinking.

References

[1] Sainsbury, Richard. *Paradoxes*. Cambridge University Press, 2009, p. 128.

Contraposition

For contraposition in the field of traditional logic, see Contraposition (traditional logic).

For contraposition in the field of symbolic logic, see Transposition (logic).

In logic, **contraposition** is a law, which says that a conditional statement is logically equivalent to its **contrapositive**. The contrapositive of the statement has its antecedent and consequent inverted and flipped: the contrapositive of $P \rightarrow Q$ is thus $\neg Q \rightarrow \neg P$. For instance, the proposition "All bats are mammals" can be restated as the conditional "If something is a bat, then it is a mammal". Now, the law says that statement is identical to the contrapositive "If something is not a mammal, then it is not a bat."

The contrapositive can be compared with three other relationships between conditional statements:

- **Inversion (the inverse):** $\neg P \rightarrow \neg Q$

"If something is not a bat, then it is not a mammal." Unlike the contrapositive, the inverse's truth value is not at all dependent on whether or not the original proposition was true, as evidenced here. The inverse here is clearly not true.

- **Conversion (the converse):** $Q \rightarrow P$.

"If something is a mammal, then it is a bat." The converse is actually the contrapositive of the inverse and so always has the same truth value as the inverse, which is not necessarily the same as that of the original proposition.

- **Negation:**

"There exists a bat that is not a mammal." If the negation is true, the original proposition (and by extension the contrapositive) is untrue. Here, of course, the negation is untrue.

Note that if $P \rightarrow Q$ is true and we are given that Q is false, $\neg Q$, it can logically be concluded that P must be false, $\neg P$. This is often called the *law of contrapositive*, or the *modus tollens* rule of inference.

Simple proof using Euler diagrams

Consider the Euler diagram on the right. It appears clear that if something is in A, it must be in B, as well. We can rephrase all A is (in) B as

$$(A \rightarrow B)$$

It is also clear that anything that is **not** within B can **not** be within A, either. This statement,

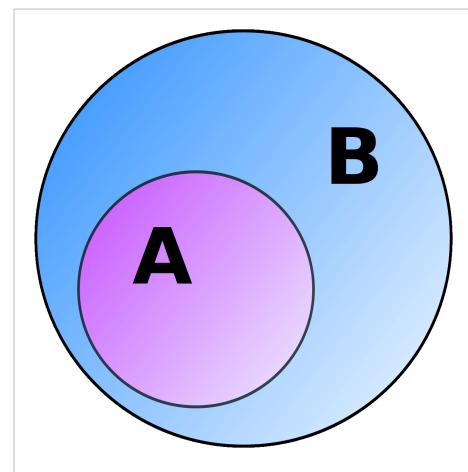
$$(\neg B \rightarrow \neg A)$$

is the contrapositive. Therefore we can say that

$$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$$

Practically speaking, this may make life much easier when trying to prove something. For example, if we want to prove that every girl in the United States (A) is blonde (B), we can either try to directly prove $(A \rightarrow B)$ by checking all girls in the United States to see if they are all blonde. Alternatively, we can try to prove $(\neg B \rightarrow \neg A)$ by checking all non-blonde girls to see if they are all outside the US. This means that if we find at least one non-blonde girl within the US, we will have disproved $(\neg B \rightarrow \neg A)$, and equivalently $(A \rightarrow B)$.

To conclude, for any statement where A implies B, then *not* B always implies *not* A. Proving or disproving either one of these statements automatically proves or disproves the other. They are fully equivalent.



Formal definition

A proposition Q is implicated by a proposition P when the following relationship holds:

$$(P \rightarrow Q)$$

In vernacular terms, this states that, "if P , then Q ", or, "if *Socrates is a man*, then *Socrates is human*." In a conditional such as this, P is the antecedent, and Q is the consequent. One statement is the **contrapositive** of the other only when its antecedent is the negated consequent of the other, and vice versa. The contrapositive of the example is

$$(\neg Q \rightarrow \neg P).$$

That is, "If not- Q , then not- P ", or, more clearly, "If Q is not the case, then P is not the case." Using our example, this is rendered "If *Socrates is not human*, then *Socrates is not a man*." This statement is said to be *contraposed* to the original and is logically equivalent to it. Due to their logical equivalence, stating one effectively states the other; when one is true, the other is also true. Likewise with falsity.

Strictly speaking, a contraposition can only exist in two simple conditionals. However, a contraposition may also exist in two complex conditionals, if they are similar. Thus, $\forall x(Px \rightarrow Qx)$, or "All P s are Q s," is contraposed to $\forall x(\neg Qx \rightarrow \neg Px)$, or "All non- Q s are non- P s."

Simple proof by definition of a conditional

In first-order logic, the conditional is defined as:

$$A \rightarrow B \iff \neg A \vee B$$

We have:

$$\begin{aligned} \neg A \vee B &\iff \neg A \vee (\neg \neg B) \\ &\iff \neg(\neg B) \vee \neg A \\ &\iff \neg B \rightarrow \neg A \end{aligned}$$

Simple proof by contradiction

Let:

$$(A \rightarrow B) \wedge \neg B$$

It is given that, if A is true, then B is true, and it is also given that B is not true. We can then show that A must not be true by contradiction. For, if A were true, then B would have to also be true (given). However, it is given that B is not true, so we have a contradiction. Therefore, A is not true (assuming that we are dealing with concrete statements that are either true or not true):

$$(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$$

We can apply the same process the other way round:

$$(\neg B \rightarrow \neg A) \wedge A$$

We also know that B is either true or not true. If B is not true, then A is also not true. However, it is given that A is true; so, the assumption that B is not true leads to contradiction and must be false. Therefore, B must be true:

$$(\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)$$

Combining the two proved statements makes them logically equivalent:

$$(A \rightarrow B) \iff (\neg B \rightarrow \neg A)$$

More rigorous proof of the equivalence of contrapositives

Logical equivalence between two propositions means that they are true together or false together. To prove that contrapositives are logically equivalent, we need to understand when material implication is true or false.

$$(P \rightarrow Q)$$

This is only false when P is true and Q is false. Therefore, we can reduce this proposition to the statement "False when P and not- Q " (i.e. "True when it is not the case that P and not- Q "):

$$\neg(P \wedge \neg Q)$$

The elements of a conjunction can be reversed with no effect (by commutativity):

$$\neg(\neg Q \wedge P)$$

We define R as equal to " $\neg Q$ ", and S as equal to $\neg P$ (from this, $\neg S$ is equal to $\neg\neg P$, which is equal to just P):

$$\neg(R \wedge \neg S)$$

This reads "It is not the case that (R is true and S is false)", which is the definition of a material conditional. We can then make this substitution:

$$(R \rightarrow S)$$

When we swap our definitions of R and S , we arrive at the following:

$$(\neg Q \rightarrow \neg P)$$

Comparisons

name	form	description
implication	if P then Q	first statement implies truth of second
inverse	if not P then not Q	negation of both statements
converse	if Q then P	reversal of both statements
contrapositive	if not Q then not P	reversal and negation of both statements
negation	P and not Q	contradicts the implication

Examples

Take the statement "All red objects have color." This can be equivalently expressed as "If an object is red, then it has color."

- The **contrapositive** is "If an object does not have color, then it is not red." This follows logically from our initial statement and, like it, it is evidently true.
- The **inverse** is "If an object is not red, then it does not have color." An object which is blue is not red, and still has color. Therefore in this case the inverse is false.
- The **converse** is "If an object has color, then it is red." Objects can have other colors, of course, so, the converse of our statement is false.
- The **negation** is "There exists a red object that does not have color." This statement is false because the initial statement which it negates is true.

In other words, the contrapositive is logically equivalent to a given conditional statement, though not sufficient for a biconditional.

Similarly, take the statement "All quadrilaterals have four sides," or equivalently expressed "If a polygon is a quadrilateral, then it has four sides."

- The **contrapositive** is "*If a polygon does not have four sides, then it is not a quadrilateral.*" This follows logically, and as a rule, contrapositives share the truth value of their conditional.
- The **inverse** is "*If a polygon is not a quadrilateral, then it does not have four sides.*" In this case, unlike the last example, the inverse of the argument is true.
- The **converse** is "*If a polygon has four sides, then it is a quadrilateral.*" Again, in this case, unlike the last example, the converse of the argument is true.
- The **negation** is "*There is at least one quadrilateral that does not have four sides.*" This statement is clearly false.

Since the statement and the converse are both true, it is called a biconditional, and can be expressed as "**A polygon is a quadrilateral if, and only if, it has four sides.**" (The phrase *if and only if* is sometimes abbreviated *iff*.) That is, having four sides is both necessary to be a quadrilateral, and alone sufficient to deem it a quadrilateral.

Truth

- If a statement is true, then its contrapositive is true (and vice versa).
- If a statement is false, then its contrapositive is false (and vice versa).
- If a statement's inverse is true, then its converse is true (and vice versa).
- If a statement's inverse is false, then its converse is false (and vice versa).
- If a statement's negation is false, then the statement is true (and vice versa).
- If a statement (or its contrapositive) and the inverse (or the converse) are both true or both false, it is known as a logical biconditional.

Application

Because the **contrapositive** of a statement always has the same truth value (truth or falsity) as the statement itself, it can be a powerful tool for proving mathematical theorems via proof by contradiction, as in the proof of the irrationality of the square root of 2. By the definition of a rational number, the statement can be made that "*If $\sqrt{2}$ is rational, then it can be expressed as an irreducible fraction*". This statement is **true** because it is a restatement of a true definition. The contrapositive of this statement is "*If $\sqrt{2}$ cannot be expressed as an irreducible fraction, then it is not rational*". This contrapositive, like the original statement, is also **true**. Therefore, if it can be proven that $\sqrt{2}$ cannot be expressed as an irreducible fraction, then it must be the case that $\sqrt{2}$ is not a rational number. A similar, but not identical tool for proving mathematical theorems is the proof by contraposition.

Double negation

In propositional logic, **double negation** is the theorem that states that "If a statement is true, then it is not the case that the statement is not true." This is expressed by saying that a proposition A is logically equivalent to *not (not- A)*, or by the formula $A \equiv \sim(\sim A)$ where the sign \equiv expresses logical equivalence and the sign \sim expresses negation.^[1]

Like the law of the excluded middle, this principle is considered to be a law of thought in classical logic,^[2] but it is disallowed by intuitionistic logic.^[3] The principle was stated as a theorem of propositional logic by Russell and Whitehead in *Principia Mathematica* as:

$$\ast 4 \cdot 13. \vdash . p \equiv \sim(\sim p)^{[4]}$$

"This is the principle of double negation, *i.e.* a proposition is equivalent of the falsehood of its negation."

The *principium contradictiones* of modern logicians (particularly Leibnitz and Kant) in the formula A is not *not-A*, differs entirely in meaning and application from the Aristotelian proposition [*i.e.* Law of Contradiction: *not (A and not- A) i.e. $\sim(A \& \sim A)$, or not ((B is A) and (B is not- A))*]. This latter refers to the relation between an affirmative and a negative judgment. According to Aristotle, one judgment [B is judged to be an A] contradicts another [B is judged to be a *not-A*]. The later proposition [A is not *not-A*] refers to the relation between subject and predicate in a single judgment; the predicate contradicts the subject. Aristotle states that one judgment is false when another is true; the later writers [Leibniz and Kant] state that a judgment is in itself and absolutely false, because the predicate contradicts the subject. What the later writers desire is a principle from which it can be known whether certain propositions are in themselves true. From the Aristotelian proposition we cannot immediately infer the truth or falsehood of any particular proposition, but only the impossibility of believing both affirmation and negation at the same time.^[5]

Footnotes

[1] Or alternate symbolism such as $A \leftrightarrow \neg(\neg A)$ or Kleene's *49°: $A \sim \neg\neg A$ (Kleene 1952:119; in the original Kleene uses an elongated tilde \sim for logical equivalence, approximated here with a "lazy S".)

[2] Hamilton is discussing Hegel in the following: "In the more recent systems of philosophy, the universality and necessity of the axiom of Reason has, with other logical laws, been controverted and rejected by speculators on the absolute.[*On principle of Double Negation as another law of Thought*, see Fries, *Logik*, §41, p. 190; Calker, *Denklehre oder Logic und Dialektik*, §165, p. 453; Beneke, *Lehrbuch der Logic*, §64, p. 41.]" (Hamilton 1860:68)

[3] The ° of Kleene's formula *49° indicates "the demonstration is not valid for both systems [classical system and intuitionistic system]", Kleene 1952:101.

[4] PM 1952 reprint of 2nd edition 1927 pages 101-102, page 117.

[5] Sigwart 1895:142-143

References

- William Hamilton, 1860, *Lectures on Metaphysics and Logic, Vol. II. Logic*; Edited by Henry Mansel and John Veitch, Boston, Gould and Lincoln. Available online from googlebooks.
- Christoph Sigwart, 1895, *Logic: The Judgment, Concept, and Inference; Second Edition, Translated by Helen Dendy*, Macmillan & Co. New York. Available online from googlebooks.
- Stephen C. Kleene, 1952, *Introduction to Metamathematics*, 6th reprinting with corrections 1971, North-Holland Publishing Company, Amsterdam NY, ISBN 0 7204 2103 9.
- Stephen C. Kleene, 1967, *Mathematical Logic*, Dover edition 2002, Dover Publications, Inc, Mineola N.Y. ISBN 0-486-42533-9 (pbk.)
- Alfred North Whitehead and Bertrand Russell, *Principia Mathematica* to *56, 2nd edition 1927, reprint 1962, Cambridge at the University Press, London UK, no ISBN or LCCN.

Frege's theorem

In metalogic and metamathematics, **Frege's theorem** is a metatheorem which states that the Peano axioms of arithmetic can be derived in second-order logic from Hume's principle. It was first proven, informally, by Gottlob Frege in his *Die Grundlagen der Arithmetik* (Foundations of Arithmetic), published in 1884, and proven more formally in his *Grundgesetze der Arithmetik* (Basic Laws of Arithmetic), published in two volumes, in 1893 and 1903. The theorem was re-discovered by Crispin Wright in the early 1980s and has since been the focus of significant work. It is at the core of the philosophy of mathematics known as neo-logicism.

Frege's theorem in propositional logic

In propositional logic, Frege's theorems refers to this tautology:

$$(P \rightarrow (Q \rightarrow R)) \rightarrow ((P \rightarrow Q) \rightarrow (P \rightarrow R))$$

External links

- Stanford Encyclopedia of Philosophy:
 - "Frege's Theorem and Foundations for Arithmetic^[1]" — by Edward Zalta.

References

[1] <http://plato.stanford.edu/entries/frege-logic/>

Idempotency of entailment

Idempotency of entailment is a property of logical systems that states that one may derive the same consequences from many instances of a hypothesis as from just one. This property can be captured by a structural rule called **contraction** and in such systems one may say that entailment is idempotent if and only if contraction is an admissible rule.

Rule of Contraction: from

A,C,C → B

is derived

A,C → B.

Or in sequent calculus notation,

$$\frac{\Gamma, C, C \vdash B}{\Gamma, C \vdash B}$$

Law of excluded middle

This article uses forms of logical notation. For a concise description of the symbols used in this notation, see List of logic symbols.

In logic, the **law of excluded middle** (or the **principle of excluded middle**) is the third of the three classic laws of thought. It states that for any proposition, either that proposition is true, or its negation is true.

The law is also known as the **law (or principle) of the excluded third**, in Latin *principium tertii exclusi*. Yet another Latin designation for this law is *tertium non datur*: "no third (possibility) is given".

The earliest known formulation is Aristotle's principle of non-contradiction, first proposed in *On Interpretation*,^[1] where he says that of two contradictory propositions (i.e. where one proposition is the negation of the other) one must be true, and the other false.^[2] He also states it as a principle in the *Metaphysics* book 3, saying that it is necessary in every case to affirm or deny,^[3] and that it is impossible that there should be anything between the two parts of a contradiction.^[4] The principle was stated as a theorem of propositional logic by Russell and Whitehead in *Principia Mathematica* as:

$$\ast 2 \cdot 11. \vdash . p \vee \sim p .$$

The principle should not be confused with the principle of bivalence, which states that every proposition is either true or false, and has only a semantical formulation.

Classic laws of thought

The principle of excluded middle, along with its complement, the law of contradiction (the second of the three classic laws of thought), are correlates of the law of identity (the first of these laws). Because the principle of identity intellectually partitions the Universe into exactly two parts: "self" and "other", it creates a dichotomy wherein the two parts are "mutually exclusive" and "jointly exhaustive". The principle of contradiction is merely an expression of the mutually exclusive aspect of that dichotomy, and the principle of excluded middle is an expression of its jointly exhaustive aspect.

Analogous laws

Some systems of logic have different but analogous laws. For some finite n -valued logics, there is an analogous law called the **law of excluded $n+1$ th**. If negation is cyclic and " \vee " is a "max operator", then the law can be expressed in the object language by $(P \vee \sim P \vee \sim \sim P \vee \dots \vee \sim \dots \sim P)$, where " $\sim \dots \sim$ " represents $n-1$ negation signs and " $\vee \dots \vee$ " $n-1$ disjunction signs. It is easy to check that the sentence must receive at least one of the n truth values (and not a value that is not one of the n).

Other systems reject the law entirely.

Examples

For example, if P is the proposition:

Socrates is mortal.

then the law of excluded middle holds that the logical disjunction:

Either Socrates is mortal, or it is not the case that Socrates is mortal.

is true by virtue of its form alone. That is, the "middle" position, that Socrates is neither mortal nor not-mortal, is excluded by logic, and therefore either the first possibility (*Socrates is mortal*) or its negation (*it is not the case that Socrates is mortal*) must be true.

An example of an argument that depends on the law of excluded middle follows.^[5] We seek to prove that there exist two irrational numbers a and b such that

a^b is rational.

It is known that $\sqrt{2}$ is irrational (see proof). Consider the number

$$\sqrt{2}^{\sqrt{2}}.$$

Clearly (excluded middle) this number is either rational or irrational. If it is rational, the proof is complete, and

$$a = \sqrt{2} \text{ and } b = \sqrt{2}.$$

But if $\sqrt{2}^{\sqrt{2}}$ is irrational, then let

$$a = \sqrt{2}^{\sqrt{2}} \text{ and } b = \sqrt{2}.$$

Then

$$a^b = \left(\sqrt{2}^{\sqrt{2}}\right)^{\sqrt{2}} = \sqrt{2}^{(\sqrt{2} \cdot \sqrt{2})} = \sqrt{2}^2 = 2,$$

and 2 is certainly rational. This concludes the proof.

In the above argument, the assertion "this number is either rational or irrational" invokes the law of excluded middle. An intuitionist, for example, would not accept this argument without further support for that statement. This might come in the form of a proof that the number in question is in fact irrational (or rational, as the case may be); or a finite algorithm that could determine whether the number is rational or not.

The Law in non-constructive proofs over the infinite

The above proof is an example of a *non-constructive* proof disallowed by intuitionists:

The proof is nonconstructive because it doesn't give specific numbers a and b that satisfy the theorem but only two separate possibilities, one of which must work. (Actually $a = \sqrt{2}^{\sqrt{2}}$ is irrational but there is no known easy proof of that fact.) (Davis 2000:220)

By *non-constructive* Davis means that "a proof that there actually are mathematic entities satisfying certain conditions would have to provide a method to exhibit explicitly the entities in question." (p. 85). Such proofs presume the existence of a totality that is complete, a notion disallowed by intuitionists when extended to the *infinite*—for them the infinite can never be completed:

In classical mathematics there occur *non-constructive* or *indirect* existence proofs, which intuitionists do not accept. For example, to prove *there exists an n such that P(n)*, the classical mathematician may deduce a contradiction from the assumption for all n , not $P(n)$. Under both the classical and the intuitionistic logic, by reductio ad absurdum this gives *not for all n, not P(n)*. The classical logic allows this result to be transformed into *there exists an n such that P(n)*, but not in general the intuitionistic... the classical meaning, that somewhere in the completed infinite totality of the natural numbers there occurs an n such that $P(n)$, is not available to him, since he does not conceive the natural numbers as a completed totality.^[6] (Kleene 1952:49–50)

Indeed, Hilbert and Brouwer both give examples of the law of excluded middle extended to the infinite. Hilbert's example: "the assertion that either there are only finitely many prime numbers or there are infinitely many" (quoted in Davis 2000:97); and Brouwer's: "Every mathematical species is either finite or infinite." (Brouwer 1923 in van Heijenoort 1967:336).

In general, intuitionists allow the use of the law of excluded middle when it is confined to discourse over finite collections (sets), but not when it is used in discourse over infinite sets (e.g. the natural numbers). Thus intuitionists absolutely disallow the blanket assertion: "For all propositions P concerning infinite sets D : P or $\sim P$ " (Kleene 1952:48).

For more about the conflict between the intuitionists (e.g. Brouwer) and the formalists (Hilbert) see *Foundations of mathematics and Intuitionism*.

Putative counterexamples to the law of excluded middle include the liar paradox or Quine's Paradox. Certain resolutions of these paradoxes, particularly Graham Priest's dialetheism as formalised in LP, have the law of excluded middle as a theorem, but resolve out the Liar as both true and false. In this way, the law of excluded middle is true, but because truth itself, and therefore disjunction, is not exclusive, it says next to nothing if one of the disjuncts is paradoxical, or both true and false.

History

Aristotle

Aristotle wrote that ambiguity can arise from the use of ambiguous names, but cannot exist in the facts themselves:

It is impossible, then, that "being a man" should mean precisely "not being a man", if "man" not only signifies something about one subject but also has one significance. ... And it will not be possible to be and not to be the same thing, except in virtue of an ambiguity, just as if one whom we call "man", and others were to call "not-man"; but the point in question is not this, whether the same thing can at the same time be and not be a man in name, but whether it can be in fact. (*Metaphysics* 4.4, W.D. Ross (trans.), GBWW 8, 525–526).

Aristotle's assertion that "...it will not be possible to be and not to be the same thing", which would be written in propositional logic as $\neg(P \wedge \neg P)$, is a statement modern logicians could call the law of excluded middle ($P \vee \neg P$), as distribution of the negation of Aristotle's assertion makes them equivalent, regardless that the former claims that no statement is *both* true and false, while the latter requires that any statement is *either* true or false.

However, Aristotle also writes, "since it is impossible that contradictories should be at the same time true of the same thing, obviously contraries also cannot belong at the same time to the same thing" (Book IV, CH 6, p. 531). He then proposes that "there cannot be an intermediate between contradictories, but of one subject we must either affirm or deny any one predicate" (Book IV, CH 7, p. 531). In the context of Aristotle's traditional logic, this is a remarkably precise statement of the law of excluded middle, $P \vee \neg P$.

Leibniz

Its usual form, "Every judgment is either true or false" [footnote 9]..."(from Kolmogorov in van Heijenoort, p. 421) footnote 9: "This is Leibniz's very simple formulation (see *Nouveaux Essais*, IV,2)...." (ibid p 421)

Bertrand Russell and *Principia Mathematica*

Bertrand Russell asserts a distinction between the "law of excluded middle" and the "law of noncontradiction". In *The Problems of Philosophy*, he cites three "Laws of Thought" as more or less "self-evident" or "a priori" in the sense of Aristotle:

1. Law of identity: "Whatever is, is."
2. Law of noncontradiction: "Nothing can both be and not be."
3. **Law of excluded middle:** "Everything must either be or not be."

These three laws are samples of self-evident logical principles... (p. 72)

It is correct, at least for bivalent logic—i.e. it can be seen with a Karnaugh map—that Russell's Law (2) removes "the middle" of the inclusive-or used in his law (3). And this is the point of Reichenbach's demonstration that some believe the *exclusive*-or should take the place of the *inclusive*-or.

About this issue (in admittedly very technical terms) Reichenbach observes:

The tertium non datur

29. $(x)[f(x) \vee \neg f(x)]$

is not exhaustive in its major terms and is therefore an inflated formula. This fact may perhaps explain why some people consider it unreasonable to write (29) with the inclusive-'or', and want to have it written with the sign of the *exclusive*'-or'

30. $(x)[f(x) \oplus \neg f(x)]$, where the symbol " \oplus " signifies exclusive-or^[7]

in which form it would be fully exhaustive and therefore nomological in the narrower sense.
(Reichenbach, p. 376)

In line (30) the "(x)" means "for all" or "for every", a form used by Russell and Reichenbach; today the symbolism is usually $\forall x$. Thus an example of the expression would look like this:

- (pig): $(Flies(pig) \oplus \neg Flies(pig))$
- (For all instances of "pig" seen and unseen): ("Pig does fly" or "Pig does not fly" but not both simultaneously)

A formal definition from *Principia Mathematica*

Principia Mathematica (PM) defines the law of excluded middle formally:

*2.1 : $\neg p \vee p$ (*PM* p. 101) Example: Either it is true that "this is red", or it is true that "this is not red". Hence it is true that "this is red or this is not red". (See below for more about how this is derived from the primitive axioms).

So just what is "truth" and "falsehood"? At the opening *PM* quickly announces some definitions:

Truth-values. The "truth-values" of a proposition is *truth* if it is true and *falsehood* if it is false* [*This phrase is due to Frege]...the truth-value of " $p \vee q$ " is truth if the truth-value of either p or q is truth, and is falsehood otherwise ... that of " $\neg p$ " is the opposite of that of p ..." (p. 7-8)

This is not much help. But later, in a much deeper discussion, ("Definition and systematic ambiguity of Truth and Falsehood" Chapter II part III, p. 41 ff) *PM* defines truth and falsehood in terms of a relationship between the "a" and the "b" and the "percipient". For example "This 'a' is 'b'" (e.g. "This 'object a' is 'red'") really means "'object a' is a sense-datum" and "'red' is a sense-datum", and they "stand in relation" to one another and in relation to "I". Thus what we really mean is: "I perceive that 'This object a is red'" and this is an undeniable-by-3rd-party "truth".

PM further defines a distinction between a "sense-datum" and a "sensation":

That is, when we judge (say) "this is red", what occurs is a relation of three terms, the mind, and "this", and "red". On the other hand, when we perceive "the redness of this", there is a relation of two terms, namely the mind and the complex object "the redness of this" (pp. 43–44).

Russell reiterated his distinction between "sense-datum" and "sensation" in his book *The Problems of Philosophy* (1912) published at the same time as *PM* (1910–1913):

Let us give the name of "sense-data" to the things that are immediately known in sensation: such things as colours, sounds, smells, hardnesses, roughnesses, and so on. We shall give the name "sensation" to the experience of being immediately aware of these things... The colour itself is a sense-datum, not a sensation. (p. 12)

Russell further described his reasoning behind his definitions of "truth" and "falsehood" in the same book (Chapter XII *Truth and Falsehood*).

Consequences of the law of excluded middle in *Principia Mathematica*

From the law of excluded middle, formula $\Box 2.1$ in *Principia Mathematica*, Whitehead and Russell derive some of the most powerful tools in the logician's argumentation toolkit. (In *Principia Mathematica*, formulas and propositions are identified by a leading asterisk and two numbers, such as " $\Box 2.1$ ".)

$\Box 2.1 \sim p \vee p$ "This is the Law of excluded middle" (*PM*, p. 101).

The proof of $\Box 2.1$ is roughly as follows: "primitive idea" 1.08 defines $p \rightarrow q = \sim p \vee q$. Substituting p for q in this rule yields $p \rightarrow p = \sim p \vee p$. Since $p \rightarrow p$ is true (this is Theorem 2.08, which is proved separately), then $\sim p \vee p$ must be true.

$\Box 2.11 p \vee \sim p$ (Permutation of the assertions is allowed by axiom 1.4)

$\Box 2.12 p \rightarrow \sim(\sim p)$ (Principle of double negation, part 1: if "this rose is red" is true then it's not true that "this rose is not-red" is true".)

$\Box 2.13 p \vee \sim\{\sim(\sim p)\}$ (Lemma together with 2.12 used to derive 2.14)

$\Box 2.14 \sim(\sim p) \rightarrow p$ (Principle of double negation, part 2)

$\Box 2.15 (\sim p \rightarrow q) \rightarrow (\sim q \rightarrow p)$ (One of the four "Principles of transposition". Similar to 1.03, 1.16 and 1.17. A very long demonstration was required here.)

$\Box 2.16 (p \rightarrow q) \rightarrow (\sim q \rightarrow \sim p)$ (If it's true that "If this rose is red then this pig flies" then it's true that "If this pig doesn't fly then this rose isn't red.")

$\Box 2.17 (\sim p \rightarrow \sim q) \rightarrow (q \rightarrow p)$ (Another of the "Principles of transposition".)

$\Box 2.18 (\sim p \rightarrow p) \rightarrow p$ (Called "The complement of *reductio ad absurdum*. It states that a proposition which follows from the hypothesis of its own falsehood is true" (*PM*, pp. 103–104).)

Most of these theorems—in particular $\Box 2.1$, $\Box 2.11$, and $\Box 2.14$ —are rejected by intuitionism. These tools are recast into another form that Kolmogorov cites as "Hilbert's four axioms of implication" and "Hilbert's two axioms of negation" (Kolmogorov in van Heijenoort, p. 335).

Propositions $\Box 2.12$ and $\Box 2.14$, "double negation": The intuitionist writings of L. E. J. Brouwer refer to what he calls "the principle of the reciprocity of the multiple species, that is, the principle that for every system the correctness of a property follows from the impossibility of the impossibility of this property" (Brouwer, *ibid*, p. 335).

This principle is commonly called "the principle of double negation" (*PM*, pp. 101–102). From the law of excluded middle ($\Box 2.1$ and $\Box 2.11$), *PM* derives principle $\Box 2.12$ immediately. We substitute $\sim p$ for p in 2.11 to yield $\sim p \vee \sim(\sim p)$, and by the definition of implication (i.e. 1.01 $p \rightarrow q = \sim p \vee q$) then $\sim p \vee \sim(\sim p) = p \rightarrow \sim(\sim p)$. QED (The derivation of 2.14 is a bit more involved.)

Criticisms

Many modern logic systems reject the law of excluded middle, replacing it with the concept of negation as failure. That is, there is a third possibility: the truth of a proposition is unknown. The principle of negation-as-failure is used as a foundation for autoepistemic logic, and is widely used in logic programming. In these systems, the programmer is free to assert the law of excluded middle as a true fact; it is not built-in *a priori* into these systems.

Mathematicians such as L. E. J. Brouwer and Arend Heyting contested the usefulness of the law of excluded middle in the context of modern mathematics^[8]

Stéphane Lupasco (1900–1988) has also substantiated the logic of the included middle, showing that it constitutes "a true logic, mathematically formalized, multivalent (with three values: A, non-A, and T) and non-contradictory".^[9]

Quantum mechanics is said to be an exemplar of this logic, through the superposition of "yes" and "no" quantum states; the included middle is also mentioned as one of the three axioms of transdisciplinarity, without which reality cannot be understood.^[10]

Footnotes

- [1] Geach p. 74
- [2] *On Interpretation*, c. 9
- [3] *Metaphysics* 2, 996b 26–30
- [4] *Metaphysics* 7, 1011b 26–27
- [5] This well-known example of a non-constructive proof depending on the law of excluded middle can be found in many places, for example:
Megill, Norm. *Metamath: A Computer Language for Pure Mathematics*, footnote on p. 17, (<http://us.metamath.org/index.html#book>) and Davis 2000:220, footnote 2.
- [6] In a comparative analysis (pp. 43–59) of the three "-isms" (and their foremost spokesmen)—Logicism (Russell and Whitehead), Intuitionism (Brouwer) and Formalism (Hilbert)—Kleene turns his thorough eye toward intuitionism, its "founder" Brouwer, and the intuitionists' complaints with respect to the law of excluded middle as applied to arguments over the "completed infinite".
- [7] The original symbol as used by Reichenbach is an upside down V, nowadays used for AND. The AND for Reichenbach is the same as that used in Principia Mathematica -- a "dot" cf p. 27 where he shows a truth table where he defines "a.b". Reichenbach defines the exclusive-or on p. 35 as "the negation of the equivalence". One sign used nowadays is a circle with a + in it, i.e. \oplus (because in binary, $a \oplus b$ yields modulo-2 addition -- addition without carry). Other signs are \neq (not identical to), or \neq (not equal to).
- [8] "Proof and Knowledge in Mathematics" by Michael Detlefsen (http://books.google.co.uk/books?id=uUC30fqhdIAC&pg=PA138&dq=the+principle+of+excluded+middle+criticism+of&hl=en&ei=tzXUTfy-I8rysgaU1vTiAg&sa=X&oi=book_result&ct=result&resnum=3&ved=0CDYQ6AEwAg#v=onepage&q=the principle of excluded middle criticism of&f=false)
- [9] Basarab Nicolescu, 2010, "Methodology of Transdisciplinarity - Levels of Reality, Logic of the Included Middle and Complexity"
Transdisciplinary Journal of Engineering and Science, vol 2010, p.31 (http://www.theatlas.org/index.php?option=com_phocadownload&view=category&id=21:engineering-science&download=181:methodology-of-transdisciplinarity-levels-of-reality-logic-of-the-included-middle-and-complexity&Itemid=157)
- [10] Basarab Nicolescu, 2010, "Methodology of Transdisciplinarity - Levels of Reality, Logic of the Included Middle and Complexity"
Transdisciplinary Journal of Engineering and Science, vol 2010, p.31 (http://www.theatlas.org/index.php?option=com_phocadownload&view=category&id=21:engineering-science&download=181:methodology-of-transdisciplinarity-levels-of-reality-logic-of-the-included-middle-and-complexity&Itemid=157)

References

- Aquinas, Thomas, "Summa Theologica", Fathers of the English Dominican Province (trans.), Daniel J. Sullivan (ed.), vols. 19–20 in Robert Maynard Hutchins (ed.), *Great Books of the Western World*, Encyclopædia Britannica, Inc., Chicago, IL, 1952. Cited as GB 19–20.
- Aristotle, "Metaphysics", W.D. Ross (trans.), vol. 8 in Robert Maynard Hutchins (ed.), *Great Books of the Western World*, Encyclopædia Britannica, Inc., Chicago, IL, 1952. Cited as GB 8. 1st published, W.D. Ross (trans.), *The Works of Aristotle*, Oxford University Press, Oxford, UK.
- Martin Davis 2000, *Engines of Logic: Mathematicians and the Origin of the Computer*, W. W. Norton & Company, NY, ISBN 0-393-32229-7 pbk.
- Dawson, J., *Logical Dilemmas, The Life and Work of Kurt Gödel*, A.K. Peters, Wellesley, MA, 1997.
- van Heijenoort, J., *From Frege to Gödel, A Source Book in Mathematical Logic, 1879–1931*, Harvard University Press, Cambridge, MA, 1967. Reprinted with corrections, 1977.
- Luitzen Egbertus Jan Brouwer, 1923, *On the significance of the principle of excluded middle in mathematics, especially in function theory* [reprinted with commentary, p. 334, van Heijenoort]
- Andrei Nikolaevich Kolmogorov, 1925, *On the principle of excluded middle*, [reprinted with commentary, p. 414, van Heijenoort]
- Luitzen Egbertus Jan Brouwer, 1927, *On the domains of definitions of functions*, [reprinted with commentary, p. 446, van Heijenoort] Although not directly germane, in his (1923) Brouwer uses certain words defined in this paper.
- Luitzen Egbertus Jan Brouwer, 1927(2), *Intuitionistic reflections on formalism*, [reprinted with commentary, p. 490, van Heijenoort]
- Stephen C. Kleene 1952 original printing, 1971 6th printing with corrections, 10th printing 1991, *Introduction to Metamathematics*, North-Holland Publishing Company, Amsterdam NY, ISBN 0-7204-2103-9.

- Kneale, W. and Kneale, M., *The Development of Logic*, Oxford University Press, Oxford, UK, 1962. Reprinted with corrections, 1975.
- Alfred North Whitehead and Bertrand Russell, *Principia Mathematica to *56*, Cambridge at the University Press 1962 (Second Edition of 1927, reprinted). Extremely difficult because of arcane symbolism, but a must-have for serious logicians.
- Bertrand Russell, *The Problems of Philosophy, With a New Introduction by John Perry*, Oxford University Press, New York, 1997 edition (first published 1912). Very easy to read: Russell was a wonderful writer.
- Bertrand Russell, *The Art of Philosophizing and Other Essays*, Littlefield, Adams & Co., Totowa, NJ, 1974 edition (first published 1968). Includes a wonderful essay on "The Art of drawing Inferences".
- Hans Reichenbach, *Elements of Symbolic Logic*, Dover, New York, 1947, 1975.
- Tom Mitchell, *Machine Learning*, WCB McGraw-Hill, 1997.
- Constance Reid, *Hilbert*, Copernicus: Springer-Verlag New York, Inc. 1996, first published 1969. Contains a wealth of biographical information, much derived from interviews.
- Bart Kosko, *Fuzzy Thinking: The New Science of Fuzzy Logic*, Hyperion, New York, 1993. Fuzzy thinking at its finest. But a good introduction to the concepts.
- David Hume, *An Inquiry Concerning Human Understanding*, reprinted in Great Books of the Western World Encyclopædia Britannica, Volume 35, 1952, p. 449 ff. This work was published by Hume in 1758 as his rewrite of his "juvenile" *Treatise of Human Nature: Being An attempt to introduce the experimental method of Reasoning into Moral Subjects Vol. I, Of The Understanding* first published 1739, reprinted as: David Hume, *A Treatise of Human Nature*, Penguin Classics, 1985. Also see: David Applebaum, *The Vision of Hume*, Vega, London, 2001: a reprint of a portion of *An Inquiry* starts on p. 94 ff

External links

- "Contradiction" entry (<http://plato.stanford.edu/entries/contradiction/>) in the Stanford Encyclopedia of Philosophy

Law of identity

This article uses forms of logical notation. For a concise description of the symbols used in this notation, see List of logic symbols.

In logic, the **law of identity** is the first of the three classical laws of thought. It states that: "each thing is the same with itself and different from another": "A is A and not $\sim A$ ". By this it is meant that each thing (be it a universal or a particular) is composed of its own unique set of characteristic qualities or features, which the ancient Greeks called its essence. Consequently, things that have the same essence are the same thing, while things that have different essences are different things.^[1] In its symbolic representation:("A is A"), the first element of the proposition represents the subject (thing) and the second element, the predicate (its essence), with the copula "is" signifying the relation of "identity".^[2] Further, since a definition is an expression of the essence of that thing with which the linguistic term is associated, it follows that it is through its definition that the identity of a thing is established.^[3] For example, in the definitive proposition:^[4]"A lawyer is a person qualified and authorized to practice law", the subject (lawyer) and the predicate (person qualified and authorized to practice law) are declared to be one and the same thing (identical). Consequently, the Law of Identity prohibits us from rightfully calling anything other than "a person qualified and authorized to practice law" a "lawyer".

In logical discourse, violations of the Law of Identity (LOI) result in the informal logical fallacy known as equivocation.^[5] That is to say, we cannot use the same term in the same discourse while having it signify difference senses or meanings – even though the different meanings are conventionally prescribed to that term. In everyday language, violations of the LOI introduce ambiguity into the discourse, making it difficult to form an interpretation at the desired level of specificity.

History

The earliest recorded use of the law appears to occur in Plato's dialogue Theaetetus (185a), wherein Socrates attempts to establish that what we call "sounds" and "colours" are two different classes of thing:

Socrates: How about sounds and colours: in the first place you would admit that they both exist?

Theaetetus: Yes.

Socrates: And that either of them is **different from the other, and the same with itself?**

Theaetetus: Certainly.

Socrates: And that both are two and each of them one?

Theaetetus: Yes.

Aristotle takes recourse to the law of identity - though he does not identify it as such - in an attempt to negatively demonstrate the law of non-contradiction. However, in doing so, he shows that the law of non-contradiction is not the more fundamental of the two:

"First then this at least is obviously true, that the word 'be' or 'not be' has a definite meaning, so that not everything will be 'so and not so'. Again, if 'man' has one meaning, let this be 'two-footed animal'; by having one meaning I understand this:-if 'man' means 'X', then if A is a man 'X' will be what 'being a man' means for him. (It makes no difference even if one were to say a word has several meanings, if only they are limited in number; for to each definition there might be assigned a different word. For instance, we might say that 'man' has not one meaning but several, one of which would have one definition, viz. 'two-footed animal', while there might be also several other definitions if only they were limited in number; for a peculiar name might be assigned to each of the definitions. If, however, they were not limited but one were to say that the word has an infinite number of meanings, obviously reasoning would be impossible; for not to have one meaning is to have no meaning, and if words have no meaning our reasoning with one another, and indeed with ourselves, has been annihilated; for it is

impossible to think of anything if we do not think of one thing; but if this is possible, one name might be assigned to this thing." - (*Metaphysics*, Book IV, Part 4)

Both Thomas Aquinas (*Met.* IV., lect. 6) and Duns Scotus (*Quaest. sup. Met.* IV., Q. 3) follow Aristotle. Antonius Andreas, the Spanish disciple of Scotus (d. 1320) argues that the first place should belong to the law "Every Being is a Being" (*Omne Ens est Ens*, Qq. in *Met.* IV., Q. 4), but the late scholastic writer Francisco Suarez (*Disp. Met.* III., § 3) disagreed, also preferring to follow Aristotle.

Another possible allusion to the same principle may be found in the writings of Nicholas of Cusa (1431-1464) where he says:

... there cannot be several things exactly the same, for in that case there would not be several things, but the same thing itself. Therefore all things both agree with and differ from one another. [6]

Gottfried Wilhelm Leibniz claimed that the law of Identity, which he expresses as 'Everything is what it is,' is the first primitive truth of reason which is affirmative, and the law of noncontradiction, is the first negative truth (*Nouv. Ess.* IV., 2, § i), arguing that "the statement that a thing is what it is, is prior to the statement that it is not another thing" (*Nouv. Ess.* IV., 7, § 9). Wilhelm Wundt credits Gottfried Leibniz with the symbolic formulation, "A is A". [7]

George Boole, in the introduction to his treatise *An Investigation of the Laws of Thought*, made the following observation with respect to the nature of language and those principles that must inhere naturally within them, if they are to be intelligible:

"There exist, indeed, certain general principles founded in the very nature of language, by which the use of symbols, which are but the elements of scientific language, is determined. To a certain extent these elements are arbitrary. Their interpretation is purely conventional: we are permitted to employ them in whatever sense we please. But this permission is limited by two indispensable conditions, first, that from the sense once conventionally established we never, in the same process of reasoning, depart; secondly, that the laws by which the process is conducted be founded exclusively upon the above fixed sense or meaning of the symbols employed."

John Locke (*Essay Concerning Human Understanding* IV. vii. iv. ("Of Maxims") says:

... whenever the mind with attention considers any proposition, so as to perceive the two ideas signified by the terms, and affirmed or denied one of the other to be the same or different; it is presently and infallibly certain of the truth of such a proposition; and this equally whether these propositions be in terms standing for more general ideas, or such as are less so: e.g., whether the general idea of Being be affirmed of itself, as in this proposition, "whatsoever is, is"; or a more particular idea be affirmed of itself, as "a man is a man"; or, "whatsoever is white is white" ...

African Spir proclaims the law of identity as the fundamental law of knowledge, which is opposed to the changing appearance of the empirical reality. [8]

References

- [1] "Two things are called one, when the definition which states the essence of one is indivisible from another definition which shows us the other (though in itself every definition is divisible)." [Aristotle's *Metaphysics*, Book VI, Part 4 (c) - Translated by W. D. Ross]
- [2] "Each thing itself, then, and its essence are one and the same in no merely accidental way, as is evident both from the preceding arguments and because to know each thing, at least, is just to know its essence, so that even by the exhibition of instances it becomes clear that both must be one." [Aristotle's *Metaphysics*, Book VII, Part 6 - Translated by W. D. Ross]
- [3] "For if a definition is an expression signifying the essence of the thing and the predicates contained therein ought also to be the only ones which are predicated of the thing in the category of essence; and genera and differentiae are so predicated in that category: it is obvious that if one were to get an admission that so and so are the only attributes predicated in that category, the expression containing so and so would of necessity be a definition; for it is impossible that anything else should be a definition, seeing that there is not anything else predicated of the thing in the category of essence." [Aristotle's *Topics*, Book VII, Part 1 - Translated by W. A. Pickard-Cambridge]
- [4] A definitive proposition is that wherein one term is the definition of the other, e.g., "Hope is the looking with pleasure into the future."
- [5] Things are said to be named 'equivocally' when, though they have a common name, the definition corresponding with the name differs for each.

- [6] De Venatione Sapientiae, 23.
- [7] La philosophie éternelle ou traditionnelle, la métaphysique, la logique, la raison et l'intelligence (<http://perso.orange.fr/thomiste/eternelb.htm>)
- [8] *Forschung nach der Gewissheit in der Erkenntnis der Wirklichkeit*, Leipzig, J.G. Findel, 1869 and *Denken und Wirklichkeit: Versuch einer Erneuerung der kritischen Philosophie*, Leipzig, J. G. Findel, 1873.

Law of noncontradiction

This article uses forms of logical notation. For a concise description of the symbols used in this notation, see List of logic symbols.

In classical logic, the **law of non-contradiction** (LNC) (or the **law of contradiction** (PM) or the **principle of non-contradiction** (PNC), or the **principle of contradiction**) is the second of the three classic laws of thought. It states that contradictory statements cannot both be true in the same sense at the same time, e.g. the two propositions "*A is B*" and "*A is not B*" are mutually exclusive.

The principle was stated as a theorem of propositional logic by Russell and Whitehead in *Principia Mathematica* as:

$$\ast 3 \cdot 24. \vdash . \sim (p. \sim p)$$

The law of noncontradiction, along with its complement, the law of excluded middle (the third of the three classic laws of thought), are correlates of the law of identity (the first of the three laws). Because the law of identity partitions its logical Universe into exactly two parts, it creates a dichotomy wherein the two parts are "mutually exclusive" and "jointly exhaustive". The law of noncontradiction is merely an expression of the mutually exclusive aspect of that dichotomy, and the law of excluded middle, an expression of its jointly exhaustive aspect.

Interpretations

One difficulty in applying the law of noncontradiction is ambiguity in the propositions. For instance, if time is not explicitly specified as part of the propositions A and B, then A may be B at one time, and not at another. A and B may in some cases be made to sound mutually exclusive linguistically even though A may be partly B and partly not B at the same time. However, it is impossible to predicate of the same thing, at the same time, and in the same sense, the absence and the presence of the same fixed quality.

Eastern philosophy

The law of noncontradiction is found in ancient Indian logic as a meta-rule in the *Shrauta Sutras*, the grammar of Pāṇini,^[1] and the *Brahma Sutras* attributed to Vyasa. It was later elaborated on by medieval commentators such as Madhvacharya.

Heraclitus

According to both Plato and Aristotle, Heraclitus was *said* to have denied the law of noncontradiction. This is quite likely if, as Plato pointed out, the law of noncontradiction does not hold for changing things in the world. If a philosophy of Becoming is not possible without change, then (the potential of) what is to become must already exist in the present object. In "*We step and do not step into the same rivers; we are and we are not*", both Heraclitus's and Plato's object simultaneously must, in some sense, be both what it now is and have the potential (dynamis) of what it might become.

Unfortunately, so little remains of Heraclitus' aphorisms that not much about his philosophy can be said with certainty. He seems to have held that strife of opposites is universal both within and without, therefore *both* opposite existents or qualities must simultaneously exist, although in some instances in different respects. "*The road up and down are one and the same*" implies either the road leads both ways, or there can be no road at all. This is the logical

complement of the law of noncontradiction. According to Heraclitus, change, and the constant conflict of opposites is the universal logos of nature.

Protagoras

Personal subjective perceptions or judgments can only be said to be true at the same time in the same respect, in which case, the law of noncontradiction must be applicable to personal judgments. The most famous saying of Protagoras is: "*Man is the measure of all things: of things which are, that they are, and of things which are not, that they are not*".^[2] However, Protagoras was referring to things that are used by or in some way related to humans. This makes a great difference in the meaning of his aphorism. Properties, social entities, ideas, feelings, judgements, etc. originate in the human mind. However, Protagoras has never suggested that man must be the measure of stars, or the motion of the stars.

Parmenides

Parmenides, employed an ontological version of the law of noncontradiction to prove that being is and to deny the void, change, and motion. He also similarly disproved contrary propositions. In his poem On Nature, he said,

the only routes of inquiry there are for thinking:
the one that [it] is and that [it] cannot not be
is the path of Persuasion (for it attends upon truth)
the other, that [it] is not and that it is right that [it] not be,
this I point out to you is a path wholly inscrutable
for you could not know what is not (for it is not to be accomplished)
nor could you point it out... For the same thing is for thinking and for being

The nature of the 'is' or what-is in Parmenides is a highly contentious subject. Some have taken it to be whatever exists, some to be whatever is or can be the object of scientific inquiry.^[3]

Socrates

In Plato's early dialogues, Socrates uses the elenctic method to investigate the nature or definition of ethical concepts such as justice or virtue. Elenctic refutation depends on a dichotomous thesis, one that may be divided into exactly two mutually exclusive parts, only one of which may be true. Then Socrates goes on to demonstrate the contrary of the commonly accepted part using the law of noncontradiction. According to Gregory Vlastos,^[4] the method has the following steps:

1. Socrates' interlocutor asserts a thesis, for example "Courage is endurance of the soul", which Socrates considers false and targets for refutation.
2. Socrates secures his interlocutor's agreement to further premises, for example "Courage is a fine thing" and "Ignorant endurance is not a fine thing".
3. Socrates then argues, and the interlocutor agrees, that these further premises imply the contrary of the original thesis, in this case it leads to: "courage is not endurance of the soul".
4. Socrates then claims that he has shown that his interlocutor's thesis is false and that its negation is true.

Plato's synthesis

Plato's version of the law of noncontradiction states that "*The same thing clearly cannot act or be acted upon in the same part or in relation to the same thing at the same time, in contrary ways*" (*The Republic* (436b)). In this, Plato carefully phrases three axiomatic restrictions on *action* or reaction: 1) in the same part, 2) in the same relation, 3) at the same time. The effect is to momentarily create a frozen, timeless state, somewhat like figures frozen in action on the frieze of the Parthenon.^[5]

This way, he accomplishes two essential goals for his philosophy. First, he logically separates the Platonic world of constant change^[6] from the formally knowable world of momentarily fixed physical objects.^{[7][8]} Second, he provides the conditions for the dialectic method to be used in finding definitions, as for example in the *Sophist*. So Plato's law of noncontradiction is the empirically derived necessary starting point for all else he has to say.

In contrast, Aristotle reverses Plato's order of derivation. Rather than starting with *experience*, Aristotle begins *a priori* with the law of noncontradiction as the fundamental axiom of an analytic philosophical system.^[9] This axiom then necessitates the fixed, realist model. Now, he starts with much stronger logical foundations than Plato's non-contrariety of action in reaction to conflicting demands from the three parts of the soul.

Aristotle's contribution

The traditional source of the law of noncontradiction is Aristotle's *Metaphysics* where he gives three different versions.^[10]

1. ontological: "It is impossible that the same thing belong and not belong to the same thing at the same time and in the same respect." (1005b19-20)
2. psychological: "No one can believe that the same thing can (at the same time) be and not be." (1005b23-24)
3. logical: "The most certain of all basic principles is that contradictory propositions are not true simultaneously." (1011b13-14)

Aristotle attempts several proofs of this law. He first argues that every expression has a single meaning (otherwise we could not communicate with one another). This rules out the possibility that by "to be a man", "not to be a man" is meant. But "man" means "two-footed animal" (for example), and so if anything is a man, it is necessary (by virtue of the meaning of "man") that it must be a two-footed animal, and so it is impossible at the same time for it *not* to be a two-footed animal. Thus "it is not possible to say truly at the same time that the same thing is and is not a man" (*Metaphysics* 1006b 35). Another argument is that anyone who believes something cannot believe its contradiction (1008b).

Why does he not just get up first thing and walk into a well or, if he finds one, over a cliff? In fact, he seems rather careful about cliffs and wells.^[11]

Avicenna gives a similar argument:

Anyone who denies the law of non-contradiction should be beaten and burned until he admits that to be beaten is not the same as not to be beaten, and to be burned is not the same as not to be burned.^{[12][13]}

Leibniz and Kant

Leibniz and Kant adopted a different statement, by which the law assumes an essentially different meaning. Their formula is A is not not-A; in other words it is impossible to predicate of a thing a quality which is its contradictory. Unlike Aristotle's law this law deals with the necessary relation between subject and predicate in a single judgment. For example, in Gottlob Ernst Schulze's *Aenesidemus*, it is asserted, "... nothing supposed capable of being thought may contain contradictory characteristics." Whereas Aristotle states that one or other of two contradictory propositions must be false, the Kantian law states that a particular kind of proposition is in itself necessarily false. On the other hand there is a real connection between the two laws. The denial of the statement A is not-A presupposes some knowledge of what A is, i.e. the statement A is A. In other words a judgment about A is implied.

Kant's analytical judgments of propositions depend on presupposed concepts which are the same for all people. His statement, regarded as a logical principle purely and apart from material facts, does not therefore amount to more than that of Aristotle, which deals simply with the significance of negation^[citation needed].

Modern logics

Traditionally, in Aristotle's classical logical calculus, in evaluating any proposition there are only two possible truth values, "true" and "false." An obvious extension to classical two-valued logic is a many-valued logic for more than two possible values. In logic, a many- or multi-valued logic is a propositional calculus in which there are more than two values. Those most popular in the literature are three-valued (e.g., Łukasiewicz's and Kleene's), which accept the values "true", "false", and "unknown", finite-valued with more than three values, and the infinite-valued (e.g. fuzzy logic and probability logic) logics.

Dialetheism

Graham Priest advocates the view that *under some conditions*, some statements can be both true and false simultaneously, or may be true and false at different times. Applied universally, without specified conditions or axiomatic restrictions, this dialetheism will cause every statement, to explode, to become true. Dialetheism arises from formal logical paradoxes, such as the Liar's paradox and Russell's paradox.

Alleged impossibility of its proof or denial

As is true of all axioms of logic, the law of non-contradiction is alleged to be neither verifiable nor falsifiable, on the grounds that any proof or disproof must use the law itself prior to reaching the conclusion. In other words, in order to verify or falsify the laws of logic one must resort to logic as a weapon, an act which would essentially be self-defeating.^[14] Since the early 20th century, certain logicians have proposed logics that deny the validity of the law. Collectively, these logics are known as "paraconsistent" or "inconsistency-tolerant" logics. But not all paraconsistent logics deny the law, since they are not necessarily completely agnostic to inconsistencies in general. Graham Priest advances the strongest thesis of this sort, which he calls "dialetheism".

In several axiomatic derivations of logic,^[15] this is effectively resolved by showing that $(P \vee \neg P)$ and its negation are constants, and simply defining TRUE as $(P \vee \neg P)$ and FALSE as $\neg(P \vee \neg P)$, without taking a position as to the principle of bivalence or the law of excluded middle.

Some, such as David Lewis, have objected to paraconsistent logic on the ground that it is simply impossible for a statement and its negation to be jointly true.^[16] A related objection is that "negation" in paraconsistent logic is not really *negation*; it is merely a subcontrary-forming operator.^[17]

Notes

[1] (cf.)

[2] (80B1 DK). According to Plato's *Theaetetus*, section 152a. (<http://www.perseus.tufts.edu/cgi-bin/ptext?lookup=Plat.+Theaet.+152a>)

[3] Curd, Patricia, "Presocratic Philosophy", *The Stanford Encyclopedia of Philosophy* (Summer 2011 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/sum2011/entries/presocratics/>

[4] Gregory Vlastos, 'The Socratic Elenchus', *Oxford Studies in Ancient Philosophy I*, Oxford 1983, 27–58.

[5] James Danaher, *The Laws of Thought* (<http://www.the-philosopher.co.uk/lawsofthought.htm>) "The restrictions Plato places on the laws of thought (i.e., "in the same respect," and "at the same time,") are an attempt to isolate the object of thought by removing it from all other time but the present and all respects but one."

[6] Plato's Divided Line describes the four Platonic worlds

[7] *Cratylus*, starting at 439e (<http://www.perseus.tufts.edu/hopper/text?doc=Plat.+Crat.+439e&fromdoc=Perseus:text:1999.01.0172>)

[8] "A thing which is F at one time, or in one way, or in one relation, or from one point of view, will be all too often not-F, at another time, in another way" ("Metaphysical Paradox" in Gregory Vlastos, *Platonic Studies*, p.50)

[9] Similarly, Kant remarked that Newton "by no means dared to prove this law *a priori*, and therefore appealed rather to experience" (*Metaphysical Foundations*, 4:449)

[10] p.487

[11] 1008b, trans. Lawson-Tancred

[12] Avicenna, Metaphysics, I; commenting on Aristotle, Topics I.11.105a4–5.

[13] Of course, if Avicenna was burned on the one hand while frozen on the other, he would have soon admitted that it is not the same to be *both* burned and frozen or *neither*.

- [14] S.M. Cohen, *Aristotle on the Principle of Non-Contradiction* (<http://faculty.washington.edu/smcohen/AristotlePNC.pdf>) "Aristotle's solution in the Posterior Analytics is to distinguish between *episteme* (scientific knowledge) and *nous* (intuitive intellect). First principles, such as PNC, are not objects of scientific knowledge - since they are not demonstrable - but are still known, since they are grasped by *nous*".
- [15] Steven Wolfram, A New Kind Of Science, ISBN 1-57955-008-8
- [16] See Lewis (1982).
- [17] See Slater (1995), Béziau (2000).

References

- Benardete, Seth (1989). *Socrates' Second Sailing: On Plato's Republic*. University of Chicago Press.
- Lawson-Tancred, H (1998). *Aristotle's Metaphysics*. Penguin.
- Łukasiewicz, Jan (1910 (in Polish)), "On the Principle of Contradiction in Aristotle", *Review of Metaphysics* 24: 485–509

External links

- S.M. Cohen, "Aristotle on the Principle of Non-Contradiction" (<http://faculty.washington.edu/smcohen/AristotlePNC.pdf>), *Canadian Journal of Philosophy*, Vol. 16, No. 3
- James Danaher, "The Laws of Thought" (<http://www.the-philosopher.co.uk/lawsofthought.htm>), *The Philosopher*, Vol. LXXXII No. 1
- Paula Gottlieb, "Aristotle on Non-contradiction" (<http://plato.stanford.edu/entries/aristotle-noncontradiction/>) (Stanford Encyclopedia of Philosophy)
- Laurence Horn, "Contradiction" (<http://plato.stanford.edu/entries/contradiction/>) (Stanford Encyclopedia of Philosophy)
- Graham Priest and Francesco Berto, "Dialetheism" (<http://plato.stanford.edu/entries/dialetheism/>) (Stanford Encyclopedia of Philosophy)
- Graham Priest and Koji Tanaka, "Paraconsistent logic" (<http://plato.stanford.edu/entries/logic-paraconsistent/>) (Stanford Encyclopedia of Philosophy)
- Peter Suber, "Non-Contradiction and Excluded Middle" (<http://www.earlham.edu/~peters/courses/logsyst/pnc-pem.htm>), Earlham College

Monotonicity of entailment

Monotonicity of entailment is a property of many logical systems that states that the hypotheses of any derived fact may be freely extended with additional assumptions. In sequent calculi this property can be captured by an inference rule called **weakening**, or sometimes **thinning**, and in such systems one may say that entailment is monotone if and only if the rule is admissible. Logical systems with this property are occasionally called *monotonic logics* in order to differentiate them from non-monotonic logics.

Weakening rule

To illustrate, starting from the natural deduction sequent:

$$\Gamma \vdash C$$

weakening allows one to conclude:

$$\Gamma, A \vdash C$$

Non-monotonic logics

In most logics, weakening is either an inference rule or a metatheorem if the logic doesn't have an explicit rule. Notable exceptions are:

- Strict logic or relevant logic, where every hypothesis must be necessary for the conclusion.
- Linear logic which disallows arbitrary contraction in addition to arbitrary weakening.
- Bunched implications where weakening is restricted to additive composition.
- Various types of default reasoning.
- Abductive reasoning, the process of deriving the most likely explanations of the known facts.
- Reasoning about knowledge, where statements specifying that something is not known need to be retracted when that thing is learned.

Principle of explosion

The **principle of explosion**, (Latin: *ex falso quodlibet* or *ex contradictione sequitur quodlibet*, "from a contradiction, anything follows") or the **principle of Pseudo-Scotus**, is the law of classical logic, intuitionistic logic and similar logical systems, according to which any statement can be proven from a contradiction.^[1] That is, once a contradiction has been asserted, any proposition (or its negation) can be inferred from it.

As a demonstration of the principle, consider two contradictory statements - "All lemons are yellow" and "Not all lemons are yellow", and suppose (for the sake of argument) that both are simultaneously true. If that is the case, anything can be proven, e.g. "Santa Claus exists", by using the following argument:

- 1) We know that "All lemons are yellow" as it is defined to be true.
- 2) Therefore the statement that ("All lemons are yellow" OR "Santa Claus exists") must also be true, since the first part is true.
- 3) However, if "Not all lemons are yellow" (and this is also defined to be true), Santa Claus must exist - otherwise statement 2 would be false. It has thus been "proven" that Santa Claus exists. The same could be applied to any assertion, including the statement "Santa Claus does not exist".

Symbolic representation

The principle of explosion can be expressed in the following way (where " \vdash " symbolizes the relation of logical consequence):

$$\{\phi, \neg\phi\} \vdash \psi$$

or

$$\perp \rightarrow P.$$

This can be read as, "If one claims something is both true (ϕ) and not true ($\neg\phi$), one can logically derive *any* conclusion (ψ)."

Arguments for explosion

An informal, descriptive, argument is given above. In more formal terms, there are two kinds of argument for the principle of explosion, semantic and proof-theoretic.

The semantic argument

The first argument is *semantic* or *model-theoretic* in nature. A sentence ψ is a *semantic consequence* of a set of sentences Γ only if every model of Γ is a model of ψ . But there is no model of the contradictory set $\{\phi, \neg\phi\}$. A fortiori, there is no model of $\{\phi, \neg\phi\}$ that is not a model of ψ . Thus, vacuously, every model of $\{\phi, \neg\phi\}$ is a model of ψ . Thus ψ is a semantic consequence of $\{\phi, \neg\phi\}$.

The proof-theoretic argument

The second type of argument is *proof-theoretic* in nature. Consider the following derivations:

1. $\phi \wedge \neg\phi$
assumption
2. ϕ
from (1) by conjunction elimination
3. $\neg\phi$
from (1) by conjunction elimination
4. $\phi \vee \psi$
from (2) by disjunction introduction
5. ψ
from (3) and (4) by disjunctive syllogism
6. $(\phi \wedge \neg\phi) \rightarrow \psi$
from (5) by conditional proof (discharging assumption 1)

This is just the symbolic version of the informal argument given above, with ϕ standing for "all lemons are yellow" and ψ standing for "Santa Claus exists". From "all lemons are yellow and not all lemons are yellow" (1), we infer "all lemons are yellow" (2) and "not all lemons are yellow" (3); from "all lemons are yellow" (2), we infer "all lemons are yellow or Santa Claus exists" (4); and from "not all lemons are yellow" (3) and "all lemons are yellow or Santa Claus exists" (4), we infer "Santa Claus exists" (5). Hence, if all lemons are yellow and not all lemons are yellow, then Santa Claus exists.

Or:

1. $\phi \wedge \neg\phi$
hypothesis
2. ϕ
from (1) by conjunction elimination
3. $\neg\phi$
from (1) by conjunction elimination
4. $\neg\psi$
hypothesis
5. ϕ
reiteration of (2)
6. $\neg\psi \rightarrow \phi$
from (4) to (5) by deduction theorem
7. $(\neg\phi \rightarrow \neg\neg\psi)$
from (6) by contraposition
8. $\neg\neg\psi$
from (3) and (7) by modus ponens
9. ψ
from (8) by double negation elimination
10. $(\phi \wedge \neg\phi) \rightarrow \psi$
from (1) to (9) by deduction theorem

Or:

1. $\phi \wedge \neg\phi$

- assumption
- 2. $\neg\psi$
 - assumption
 - 3. ϕ
 - from (1) by conjunction elimination
 - 4. $\neg\phi$
 - from (1) by conjunction elimination
 - 5. $\neg\neg\psi$
 - from (3) and (4) by reductio ad absurdum (discharging assumption 2)
 - 6. ψ
 - from (5) by double negation elimination
 - 7. $(\phi \wedge \neg\phi) \rightarrow \psi$
 - from (6) by conditional proof (discharging assumption 1)

Addressing the principle

Paraconsistent logics have been developed that allow for sub-contrary forming operators. Model-theoretic paraconsistent logicians often deny the assumption that there can be no model of $\{\phi, \neg\phi\}$ and devise semantical systems in which there are such models. Alternatively, they reject the idea that propositions can be classified as true or false. Proof-theoretic paraconsistent logics usually deny the validity of one of the steps necessary for deriving an explosion, typically including disjunctive syllogism, disjunction introduction, and reductio ad absurdum.

Use

The metamathematical value of the principle of explosion is that for any logical system where this principle holds, any derived theory which proves \perp (or an equivalent form, $\phi \wedge \neg\phi$) is worthless because *all* its statements would become theorems, making it impossible to distinguish truth from falsehood. That is to say, the principle of explosion is an argument for the law of non-contradiction in classical logic, because without it all truth statements become meaningless.

References

- [1] Carnielli, W. and Marcos, J. (2001) "Ex contradictione non sequitur quodlibet" (<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.70>) *Proc. 2nd Conf. on Reasoning and Logic* (Bucharest, July 2000)

Proof by contradiction

In logic, **proof by contradiction** is a form of proof that establishes the truth or validity of a proposition by showing that the proposition's being false would imply a contradiction. Proof by contradiction is also known as **indirect proof**, **apagogical argument**, **proof by assuming the opposite**, and **reductio ad impossibilem**. It is a particular kind of the more general form of argument known as *reductio ad absurdum*.

G. H. Hardy described proof by contradiction as "one of a mathematician's finest weapons", saying "It is a far finer gambit than any chess gambit: a chess player may offer the sacrifice of a pawn or even a piece, but a mathematician offers the game."^[1]

Examples

Irrationality of the square root of 2

A classic proof by contradiction from mathematics is the proof that the square root of 2 is irrational. If it were rational, it could be expressed as a fraction a/b in lowest terms, where a and b are integers, at least one of which is odd. But if $a/b = \sqrt{2}$, then $a^2 = 2b^2$. Therefore a^2 must be even. Because the square of an odd number is odd, that in turn implies that a is even. This means that b must be odd because a/b is in lowest terms.

On the other hand, if a is even, then a^2 is a multiple of 4. If a^2 is a multiple of 4 and $a^2 = 2b^2$, then $2b^2$ is a multiple of 4, and therefore b^2 is even, and so is b .

So b is odd and even, a contradiction. Therefore the initial assumption—that $\sqrt{2}$ can be expressed as a fraction—must be false.

The length of the hypotenuse

The method of proof by contradiction has also been used to show that for any non-degenerate right triangle, the length of the hypotenuse is less than the sum of the lengths of the two remaining sides. The proof relies on the Pythagorean theorem. Letting c be the length of the hypotenuse and a and b the lengths of the legs, the claim is that $a + b > c$.

The claim is negated to assume that $a + b \leq c$. Squaring both sides results in $(a + b)^2 \leq c^2$ or, equivalently, $a^2 + 2ab + b^2 \leq c^2$. A triangle is non-degenerate if each edge has positive length, so it may be assumed that a and b are greater than 0. Therefore, $a^2 + b^2 < a^2 + 2ab + b^2 \leq c^2$. The transitive relation may be reduced to $a^2 + b^2 < c^2$. It is known from the Pythagorean theorem that $a^2 + b^2 = c^2$. This results in a contradiction since strict inequality and equality are mutually exclusive. The latter was a result of the Pythagorean theorem and the former the assumption that $a + b \leq c$. The contradiction means that it is impossible for both to be true and it is known that the Pythagorean theorem holds. It follows that the assumption that $a + b \leq c$ must be false and hence $a + b > c$, proving the claim.

No least positive rational number

Consider the proposition, P : "there is no smallest rational number greater than 0". In a proof by contradiction, we start by assuming the opposite, $\neg P$: that there is a smallest rational number, say, r .

Now $r/2$ is a rational number greater than 0 and smaller than r . (In the above symbolic argument, " $r/2$ is the smallest rational number" would be Q and " r (which is different from $r/2$) is the smallest rational number" would be $\neg Q$.) But that contradicts our initial assumption, $\neg P$, that r was the *smallest* rational number. So we can conclude that the original proposition, P , must be true — "there is no smallest rational number greater than 0".

Other

For other examples, see proof that the square root of 2 is not rational (where indirect proofs different from the above one can be found) and Cantor's diagonal argument.

In mathematical logic

In mathematical logic, the proof by contradiction is represented as:

If

$$S \cup \{P\} \vdash F$$

then

$$S \vdash \neg P.$$

or

If

$$S \cup \{\neg P\} \vdash F$$

then

$$S \vdash P.$$

In the above, P is the proposition we wish to disprove respectively prove; and S is a set of statements, which are the premises—these could be, for example, the axioms of the theory we are working in, or earlier theorems we can build upon. We consider P , or the negation of P , in addition to S ; if this leads to a logical contradiction F , then we can conclude that the statements in S lead to the negation of P , or P itself, respectively.

Note that the set-theoretic union, in some contexts closely related to logical disjunction (or), is used here for sets of statements in such a way that it is more related to logical conjunction (and).

A particular kind of indirect proof assumes that some object doesn't exist, and then proves that this would lead to a contradiction; thus, such an object must exist. Although it is quite freely used in mathematical proofs, not every school of mathematical thought accepts this kind of argument as universally valid. See further Nonconstructive proof.

Notation

Proofs by contradiction sometimes end with the word "Contradiction!". Isaac Barrow and Baermann used the notation Q.E.A., for "*quod est absurdum*" ("which is absurd"), along the lines of Q.E.D., but this notation is rarely used today.^[2] A graphical symbol sometimes used for contradictions is a downwards zigzag arrow "lightning" symbol (U+21AF: \bot), for example in Davey and Priestley.^[3] Others sometimes used include a pair of opposing arrows (as $\rightarrow\leftarrow$ or $\Rightarrow\Leftarrow$), struck-out arrows ($\not\leftrightarrow$), a stylized form of hash (such as U+2A33: \top), or the "reference mark" (U+203B: \approx).^{[4][5]} The "up tack" symbol (U+22A5: \perp) used by philosophers and logicians (see contradiction) also appears, but is often avoided due to its usage for orthogonality.

References

- [1] G. H. Hardy, *A Mathematician's Apology*; Cambridge University Press, 1992. ISBN 9780521427067. p. 94 (<http://books.google.com/books?id=beImvXUGD-MC&pg=PA94>).
- [2] Hartshorne on QED and related (<http://robin.hartshorne.net/QED.html>)
- [3] B. Davey and H.A. Priestley, Introduction to lattices and order, Cambridge University Press, 2002.
- [4] The Comprehensive LaTeX Symbol List, pg. 20. <http://www.ctan.org/tex-archive/info/symbols/comprehensive/symbols-a4.pdf>
- [5] Gary Hardegree, *Introduction to Modal Logic*, Chapter 2, pg. II–2. <http://people.umass.edu/gmhwww/511/pdf/c02.pdf>

Further reading

- Franklin, James (2011). *Proof in Mathematics: An Introduction* (<http://www.maths.unsw.edu.au/~jim/proofs.html>). chapter 6: Kew. ISBN 978-0-646-54509-7.

External links

- Proof by Contradiction (<http://zimmer.csufresno.edu/~larryc/proofs/proofs.contradict.html>) from Larry W. Cusick's How To Write Proofs (<http://zimmer.csufresno.edu/~larryc/proofs/proofs.html>)

Reductio ad absurdum

Reductio ad absurdum (Latin: "reduction to absurdity"), also known as *argumentum ad absurdum* (Latin: argument to absurdity), is a common form of argument which seeks to demonstrate that a statement is true by showing that a false, untenable, or absurd result follows from its denial, or in turn to demonstrate that a statement is false by showing that a false, untenable, or absurd result follows from its acceptance. First appearing in classical Greek philosophy (the Latin term derives from the Greek "*εἰς ἀτοπὸν ἀπαγωγή*" or *eis atopon apagoge*, "reduction to the impossible", for example in Aristotle's *Prior Analytics*), this technique has been used throughout history in both formal mathematical and philosophical reasoning, as well as informal debate.

The "absurd" conclusion of a *reductio ad absurdum* argument can take a range of forms:

Rocks have weight, otherwise we would see them floating in the air.

Society must have laws, otherwise there would be chaos.

There is no smallest positive rational number, because if there were, it could be divided by two to get a smaller one.

The first example above argues that the denial of the assertion would have a ridiculous result, against the evidence of our senses. The second argues that the denial would have an untenable result: unacceptable, unworkable or unpleasant for society. The third is a mathematical proof by contradiction, arguing that the denial of the assertion would result in a logical contradiction (there is a smallest rational number and yet there is a rational number smaller than it).

Greek philosophy

This technique is used throughout Greek philosophy, beginning with Presocratic philosophers. The earliest Greek example of a *reductio* argument is supposedly in fragments of a satirical poem attributed to Xenophanes of Colophon (c.570 – c.475 BC). Criticizing Homer's attribution of human faults to the Greek gods, he says that humans also believe that the gods' bodies have human form. But if horses and oxen could draw, they would draw the gods with horse and oxen bodies. The gods can't have both forms, so this is a contradiction. Therefore the attribution of other human characteristics to the gods, such as human faults, is also false.

The earlier dialogs of Plato (424 – 348 BC), relating the debates of his teacher Socrates, raised the use of *reductio* arguments to a formal dialectical method, now called the *Socratic method*. Typically Socrates' opponent would make an innocuous claim, then Socrates by a step-by-step train of reasoning, bringing in other background assumptions, would make the person admit that the assertion resulted in an absurd or contradictory conclusion, forcing him to abandon his assertion. The technique was also a focus of the work of Aristotle (384 – 322 BC).

The principle of non-contradiction

Aristotle clarified the connection between contradiction and falsity in his principle of non-contradiction. This states that an assertion cannot be both true and false. Therefore if the contradiction of an assertion ($\text{not-}P$) can be derived logically from the assertion (P) it can be concluded that a false assumption has been used. This technique, called *proof by contradiction* has formed the basis of *reductio ad absurdum* arguments in formal fields like logic and mathematics.

The principle of non-contradiction has seemed absolutely undeniable to most philosophers. However a few philosophers such as Heraclitus and Hegel have accepted contradictions. The discovery of contradictions at the foundations of mathematics at the beginning of the 20th century, such as Russell's paradox, have led a few philosophers such as Graham Priest to reject the principle of non-contradiction, giving rise to theories such as dialethism and paraconsistent logic which accept that some statements can be both true and false.

Straw Man argument

A *false* argument similar to *reductio ad absurdum* often seen in polemical debate is the *straw man* logical fallacy. A straw man argument attempts to refute a given proposition by showing that a slightly different or inaccurate form of the proposition (the "straw man") has an absurd, unpleasant, or ridiculous consequence, relying on the audience not to notice that the argument does not actually apply to the original proposition. For example, in a 1977 appeal of a U.S. bank robbery conviction, a prosecuting attorney said in his closing argument

I submit to you that if you can't take this evidence and find these defendants guilty on this evidence then we might as well open all the banks and say, 'Come on and get the money, boys', because we'll never be able to convict them.

The prosecutor was tacitly equating the failure to convict the defendants in one particular trial with the inability to convict *any* bank robbers, a situation with self-evident unpleasant consequences but very little connection with the outcome of the trial.

References

Article Sources and Contributors

Algebra of sets *Source:* <https://en.wikipedia.org/w/index.php?oldid=572929989> *Contributors:* Alex10023, Arthur Rubin, AugPi, Byelf2007, Charles Matthews, Corruptcopper, DonMTobin, Doshell, Esse, Giliam, Hans Adler, Isnow, Jackzhp, Jamelan, Jochen Burghardt, Josephpetty100, Juan Marquez, Kupirijo, Macrakis, Matt Popat, MegaSloth, Oleg Alexandrov, Palitras, Paul August, Policron, Salix alba, Set theorist, Slipstream, Specs112, Splintercellguy, Tcampas42, The Anome, The enemies of god, Tobias Bergemann, Trovatore, WhisperToMe, Wile E. Heresiarch, Woohookity, Yahia.barie, 48 anonymous edits

Boolean algebra (structure) *Source:* <https://en.wikipedia.org/w/index.php?oldid=576485070> *Contributors:* 2A01:E35:8B38:8680:4844:33B:4E17:C113, 2D, ABCD, Aguitarheroperson, Alai, Albmont, Allhutch, Ancheta Wis, Andreawbel, 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, Ciacci, Clarepawling, Colin Rowat, Condem, Constructive editor, Conversion script, Cronholm144, Cullinane, Cybercobra, D.Lazard, Dai mingjie, David Eppstein, Dcoetzee, Delusion23, DesertSteve, Dysprosia, ERcheck, Ed Poor, Eduardoporches, Eequor, Elias, Elwywa, Elwikipedista, Enoksr, Escher26, Fibonacci, Fredrik, Freidle, Funny Yet Tasty, Fwehrung, GOD, GTBacchus, Gauss, Giftlite, Gonzalecg, Graham87, Gregbard, Hairy Dude, Hans Adler, Helder.wiki, Heron, Honx, Ht689rg90, Hugo Herbelin, Igny, Ilmari Karonen, Ilya, Incnis Mrsi, Irmy, Isaac Rabinovitch, Ivan Bajlo, Izehar, Jarble, Jeronimo, Jiri 1984, Jitse Niesen, JoanneB, Jochen Burghardt, Joebeone, Johnleemk, Kojiti fb, Jon Awbrey, JorgeGG, Josh Cherry, Julian Mendez, Justin Johnson, KHamsun, KSmrq, KamasaK, Kephr, Kirachinnmoku, Kliparrot, Kompik, KrakatoaKatie, Kzollman, Lambiam, Lethe, LiHelpa, Linas, MER-C, MSGJ, Macrakis, Magioladitis, Magister Mathematicae, Mani1, MarSch, Markus Krötzsch, Masashi49, May, Maximus Rex, Meco, Michael Hardy, Mosekslein, Msh210, Nagytibi, Nortexoid, OKPerson, Omnicron18, Paul August, Pce3@ij.net, Perry Bebbington, Pit, Plclarke, Pleasantville, Plugwash, Poccil, Policron, Pvemburg, Qwertys, Robinh, Roman, Ruakh, Sagaciousuk, Salix alba, Samtheboy, Samuel, Sandman, SaxTeacher, Scythe33, Shellreef, SixWingedSraph, Slipstream, Sommacal alfons, Sopoforic, Staeker, StuRat, Tagremover, Talkstosocks, Tarquin, Taw, TedDunning, Tellyaddict, The Tetrast, Thecheesyskid, Tijfo098, Timwi, Tobias Bergemann, Toby Bartels, Trovatore, Ukepat, Uncle G, Vaughan Pratt, Visor, Voodoo, Wiki alf, WimdeValk, Woohookity, Wrs1864, XJaM, Yamamoto Ichiro, Zero sharp, Zundark, Јованић, Александар, 213 .. 28 anonymous edits

Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=577987919> *Contributors:* Abecedarius, Andrewa, Arthur Rubin, Autonoff, BaseballDetective, Brad7777, CBM, CMBJ, CRGreathouse, Ched, Cheerios69, Classicaeleon, D.Lazard, Daniel998, David Eppstein, Delusion23, Dianmaa, ESkog, Emijl, Francvs, Gamewizard71, Gary King, Ghaly, Gregbard, Hairy Dude, Hans Adler, Hugo Herbelin, Incnis Mrsi, ItsZippy, Iannoriel, Izno, Jiri 1984, Jmjereemy, Jmencisom, JohnBlackburne, Jrtayloriv, Khazar, LZ6387, Lambiam, LanaEditArticles, LuluQ, Magioladitis, Mattheus Kastor, MetaNest, Michael Hardy, Michiel Helvenstein, Mindmatrix, Minusia, Muammar Gaddafi, Neelix, Northamerica1000, Oleg Alexandrov, Oxonienses, Paul August, Pgallert, Pomte, Proxyma, Rlwood1, Robert Thyder, Rotsor, Shevek1981, Sofia karampataki, Soler97, StuRat, Telfordbuck, The Rahul Jain, Thorwald, Tijfo098, TonyBrooke, Trovatore, Vaughan Pratt, Waldhorn, Wavelength, William Avery, Wolfman122, Wvbaily, 102 anonymous edits

Field of sets *Source:* <https://en.wikipedia.org/w/index.php?oldid=551325898> *Contributors:* Arthur Rubin, Bart133, Charles Matthews, DaughterofSun, DaveGorman, David Shay, Deltahedron, Gala.martin, Giftlite, Hans Adler, Jarble, Kiefer.Wolfowitz, Kuratowski's Ghost, Mathematrucker, Mhss, Mike Dillon, Oleg Alexandrov, Paul August, R'n'B, Rich Farmbrough, Salix alba, Stotr, That Guy, From That Show!, Tijfo098, Tobias Bergemann, Touriste, Trovatore, William Elliot, Yahia.barie, 24 anonymous edits

List of logic symbols *Source:* <https://en.wikipedia.org/w/index.php?oldid=574414219> *Contributors:* Abdull, Alphathon, Ancheta Wis, Anonymous Dissident, Arthur Rubin, Benwing, BrendanLarvor, CBM, Cailean8, Cartesian1, Chira, Cybercobra, Da nuke, DePiep, Dead10ck, DemosDemon, Derekleungszei, Dirac1933, Donner60, Dqb124, Emijl, Erelen, Eric76, GeorgBauhaus, George Richard Leeming, Grafen, Gregbard, Hello71, Ihatemregister, InverseHypercube, Isaac Dupree, It Is Me Here, JJ Harrison, Jbalint, Jitse Niesen, Julian Mendez, KSchutte, KSmrq, Koaf, Kumioko (renamed), Lagelspeil, Lambiam, Laranedo, Melab-1, Melchoir, Mhss, Mindmatrix, Mkouklis, Msdcancer, Oscarquintana, PWilkinson, Paul August, Peter.C, Petershank, Philosophy.dude, Pioto, Quondum, Rfontiveros, Rm1271, Robocracy, RockMagnetist, Saeed.Veradi, SpeedyGonsales, The Mol Man, Trovatore, Vanisac, Wavelength, 84 anonymous edits

Logical connective *Source:* <https://en.wikipedia.org/w/index.php?oldid=565331538> *Contributors:* Abdull, Ahoerstemeier, Andres, Anonymous Dissident, Arcanedude91, Arthur Rubin, AugPi, AxelBoldt, BAICAN XXX, BD2412, Bjankuloski06en, Bookandcoffee, Boothinator, CBM, CRGreathouse, Chalst, Charm, Chris the speller, Christian List, Christian424, Conversion script, Cullinane, Danger, Darkvix, David Eppstein, DavidCary, Djk3, Dmcq, Download, Dule1818, DuncanHill, Dysprosia, El Caro, Elwikipedista, EMj, Entropeter, Filemon, Francvs, Freshenesz, Gaius Cornelius, Giftlite, GoatGuy, Graham87, Greenfernagle, Gregbard, Hans Adler, Hiding, Hriber, Hugo Herbelin, Huku-chan, Incnis Mrsi, Indolering, InvictaHOG, JHunterJ, JRSPl, JRSpiggs, Jdm64, Jeff G., Jiy, Johnuniq, Jon Awbrey, Joriki, Julian Mendez, Justin W Smith, Kaldari, Kappa, Kbdkn71, Lambiam, Letranova, Loupetter, Luis Felipe Schenone, Masquatto, Mate2code, Melab-1, Mhiji, Michael Hardy, MilesAgain, Mindmatrix, Mortense, Nahaj, Nishkid64, Neelclerc, Ojigiri, Owarihajimari, Paul August, Philogo, Pinethicket, Policron, Racconish, Rfl, RichardF, Rick Norwood, Risk one, Rmhermen, Sam Hocevar, Sbisolo, Seabucy, Sergio01, Shanes, Siroxo, Skippydo, Spoon!, Star767, Stevertigo, SurrealWarrior, Suruena, TakuyaMurata, Tijfo098, Timboat, Tolmaion, Tommor7835, Trovatore, Weaktofu, Wiki Wikardo, Yintan, ZeroOne, ZuluPapa5, 97 , 27 , 11 anonymous edits

Necessity and sufficiency *Source:* <https://en.wikipedia.org/w/index.php?oldid=577650705> *Contributors:* Abdull, Acorley, Agquarek, Akakios, Alansohn, Alastair Haines, Aranel, Arthur Rubin, Attleboro, AugPi, Black Falcon, CBM, Charles Matthews, Chris Roy, Cibic, CommonsDelinquer, Constructive editor, Cybercobra, DUStof.zaidan, DabMachine, Dante Alighieri, Dcoetze, Dgh725, Drphilharmonic, EeepEeep, Empiro2, Epastore, EpicDream86, Evercat, FT2, Flash94, Fredlighthall, Gakmo, Gandalfxiv, Giftlite, Gregbard, Gwib, Habaneroman, Hanlon1755, Igor Yalovecky, JMSwtlk, Javalenok, Jeiki Rebirth, Joonasl, K95, KYPark, Karho.Yau, Kblino, Kenneth M Burke, Kku, Krubo, Kzollman, Lamro, LiHelpa, Machine Elf 1735, Madmardigan53, Markhurd, Matuku, Maunus, Mayumashu, Michael Hardy, Miracle Pen, Motorneuron, NorsemanII, Orie0505, Pagrashitak, Pasixxx, Patrick, Paul August, PaulTanenbaum, Paxfile, Penpen, Pmanderson, Polluks, Quiddity, R'n'B, Raeky, Richard001, Rschwieb, RubySS, SFinside, Shreevatsa, Skomorokh, Spinspark, Spitfire, Steve2011, TakuyaMurata, That Guy, From That Show!, The Anome, The Gnome, Tim!, Tiroche, Tobias Bergemann, Tomisti, Uncle G, Vulakh, Wolfgang hofkirchner, Wshun, Yoak, Yonitdm, 115 anonymous edits

Propositional calculus *Source:* <https://en.wikipedia.org/w/index.php?oldid=577354663> *Contributors:* 478jjz, ABCD, Adriaticus, Almit39, Ancheta Wis, Andeggs, Applemeister, Arthur Rubin, AugPi, Ayda D, BAxelrod, Benwing, BobDrzyzgula, Bomazi, Bookandcoffee, Brad7777, BrokenSegue, CBM, Chalst, Charles Matthews, Chewings72, Christian Nassif-Haynes, Cmdrjameson, Conversion script, Created Equal, Creidieki, Cruise, Cwchng, Cybercobra, Djk3, Doezycty, Dysprosia, Elwikipedista, Epsilon0, EricBright, Evercat, Extrapiroamidale, Francvs, FranksValli, Gadfium, Gaius Cornelius, Garyzx, GeePriest, General Reader, George100, Giftlite, Glenn Willen, Graham87, Grauw, GreatWhiteNortherner, Gregbard, Gubbubu, Hairy Dude, Hanlon1755, Hans Adler, Harold F, Huku-chan, Hut 8.5, Iamthedeus, Ihope127, Imz, Incnis Mrsi, JJ Harrison, JRSpiggs, JakeVortex, Jan Hidders, Japanese Scarabin, Jason Quinn, Jean Santeuil, Jimmyleaveland, Jmath666, Jochen Burghardt, JohnBlackburne, Jon Awbrey, Joriki, Jpbown, Julian Mendez, JulianMendez, Justin Johnson, Kbdkn71, Kku, Koaf, Kutulu, Lambiam, Lethe, LiDaobing, Lightbearer, Limes, Loadmaster, Looxix, Lynxmb, Marc Venot, Materialscientist, MathMartin, Mets501, Mhss, Michael Hardy, Michael Meyling, MilesAgain, Mindmatrix, Minesweeper, Mistercupcake, MrKoplin, Mrellisdee, N4nojohn, Newbyguesses, Nightstallion, Nortexoid, Ohconfucius, Oleg Alexandrov, Omnipaediast, Our Fathers, Paroswiki, Paul August, Pejman47, Philogo, PhotoBox, PlatypineArchbow, Porcher, Qwertys, R'n'B, Rdanneskjold, Red Winged Duck, Reuben.cornel, Reyk, Rich Farmbrough, Rick Norwood, Rorro, Rossami, Russell C, Sibley, Salgueiro, Santiago Saint James, Sapphic, Scientizle, Sdorrance, ShelfSkewed, Shivakumar2009, Srnce, Stefan.vatev, Svick, Taneli HUUSKONEN, Tannkrem, Tarquin, TechnoGuyRob, Teply, TestSubject528491, The Anome, The One I Love, The Tetrast, Thinker1221, Tijfo098, Tom5 TDotO, Tompw, Trevor Andersen, Trovatore, Tzartzam, Unica111, Urhixidur, VanishedUserABC, Varuna, Vaughan Pratt, Voidxor, Waldir, Wideshanks, Wolfman122, Wvbaily, Wysprgr2005, Xee, Xenfreak, Zero sharp, Дарко Максимовић, 199 anonymous edits

Truth function *Source:* <https://en.wikipedia.org/w/index.php?oldid=576618022> *Contributors:* 777sms, Alba, Ancheta Wis, Andreweskew, Anonymous Dissident, Arthur Rubin, Banazir, Banno, Byelf2007, Cargoking, Charles Matthews, Dionyziz, Download, Drift chambers, Francvs, Gamewizard71, Gene Nygaard, Gregbard, Grumpyyoungman01, Incnis Mrsi, Jochen Burghardt, Jon Awbrey, Kwoyeu, Lgeffen, Mhss, Michael Hardy, Nortexoid, Otelemuyen, Quondum, RichardF, Robert Weemeyer, Scribo, T.c.w7468, Tabletop, Tijfo098, TubularWorld, Volfy, 28 anonymous edits

2-valued morphism *Source:* <https://en.wikipedia.org/w/index.php?oldid=529412928> *Contributors:* BD2412, Epasseto, Gzabers, Hans Adler, Iridescent, Jesse V., Onopearls, PhiRho, Pietdesomere, Ste4k, Trovatore, 3 anonymous edits

Bitwise operation *Source:* <https://en.wikipedia.org/w/index.php?oldid=577237700> *Contributors:* 0x0309, @modi, Acetate, Ahoerstemeier, Ak301, AlistairMcMillan, Amnirax, Andreas Kaufmann, AtrumVentus, AxelBoldt, Baa, BlindWanderer, Bluumoose, Brookie, CRGreathouse, Cal27, Cburnett, Ceran, Chico75, ClearQ, David Eppstein, DavidCary, Dcoetze, Delirius, Dennis714, Discospinster, Distalzou, Dthomsen8, ESkog, Edgarhs, Forderud, Fraggle81, FrenchIsAwesome, Frosted14, Fuebar, Furykef, Giftlite, Glen Pepicelli, Guillefc, Guy Macon, Hackster78, Heckledpie, Hooperbloob, Hu12, Ian.thomson, Incnis Mrsi, Intelliproject, Iste Praetor, Jajabinks97, Jarble, Jayeshsenjaliya, JeR, Jimwilliams57, Jochen Burghardt, Johnny honestly, Jonathan de Boyne Pollard, Jopazhani, Jveldeb, Kbdkn71, Kevin.dickerson, Kinema, Klutz, Kubanczyk, Locke411, Lowelian, Mate2code, Matthiaspaul, Mikeblas, Moskvax, Mr link, Muntoo, N5iln, Nathan Hamblen, Nayuki, NewEnglandYankee, Numb03, Oalders, Oli Filth, Olwgorithm, Paul August, Pilatus, Plainjane335, Plugwash, Quixplusone, Rbakker99, RedYeti, Rimcob, RomanXNS, RoyBoy, Sekelsenmat, Seraphim, Set theorist, Sharkqwy, Sillygwailo, SimonTrew, Sjakkalle, Snjrdn, Spoon!, Suruna, Tagremover, Teehee123, Tim Starling, Timeroott, Bergemann, Torszmoskus, Troller Trolling Rodriguez, Trovatore, Tslocum, Tubyyb, Turnstep, Vadmiuum, Vareyn, Visor, WQDW412, Wapcaplet, Wavelength, XP1, Yageroy, Yintan, ZZninepluralZalpha, Zenzhong8383, ZeroOne, Zvn, 268 anonymous edits

Boolean data type *Source:* <https://en.wikipedia.org/w/index.php?oldid=577429716> *Contributors:* A-Day, Aaron613, Adammck, Ahmed Fouad(the lord of programming), Amine Brikci N, AndyKali, Andyluciano, Ashrentum, BIL, Btx40, Bucketsofg, Burble, Causa sui, Cburnett, CesarB, Charles Matthews, Chris Mason, Cmcqueen1975, Ctxppc, Cybercobra, Cyclomedia,

DanBishop, Darguz Parsilvan, Datrukup, Db099221, Decltype, DevastatorIIC, Donner60, Doug Bell, EdC, Enochlau, Epbr123, Fanf, Furrykef, Gah4, Gail, Gracenotes, Greenrd, Grendelkhan, Grick, Guy Macon, Götz, Hasegeli, Henning Makholm, Hiiiiiiiiiiiiiiiiiiii, Hitherbrian, Hqb, JLaTondre, Jafet, Jamelan, Jamespurs, Jerryobject, Joeinwap, Jokes Free4Me, Jorge Stolfi, Jpaulm, JustAGal, Kbrose, Kcordina, Kindall, Kylecbenton, LOTRrules, Lambiam, Lesser Cartographies, Letter M, Machine Elf 1735, Mark Renier, Max Schwarz, Mboverload, Melchoir, Metron4, Mfc, Mikeblas, Minority Report, Mirag3, Miym, NevilleDNZ, Nickha0, Nikai, Nyktos, Ospalh, Paddy3118, Pointillist, Praetorian42, Qwazyer, R. S. Shaw, RainbowOfLight, Raise exception, Renku, Richardcavell, Rising, Rjwilmsi, Royboycrashfan, Salix alba, Sam Pointon, SamSim, Sbisolo, Seaphoto, Sgould, Shirik, SimonTrew, SoulComa, Spankthercrumpet, Spoon!, SqIPac, Staaki, Superm401, Tabletop, The Anonymouse, The Noodl Incident, Thecheesykid, Thiseye, Tijfo098, Tobias Bergemann, Tom Jenkins, Tony Sidaway, Tonyfaull, Torc2, Trovatore, Vrenator, Wknigh94, Wlievens, Zero Thrust, Zondor, Zorakoid, 177 anonymous edits

Boolean expression *Source:* <https://en.wikipedia.org/w/index.php?oldid=542432703> *Contributors:* Andreas Kaufmann, Bobo192, BorgHunter, Giraffedata, Grylliida, Michael Sloane, Miguel Andrade, NameIsRon, Patrick, R'n'B, R000t, Rune X2, Sct72, StuRat, Tamfang, Timcrall, Trovatore, 16 anonymous edits

Boolean satisfiability problem *Source:* <https://en.wikipedia.org/w/index.php?oldid=577916192> *Contributors:* 151.99.218.xxx, 2001:4DD0:FF00:907A:226:82FF:FEB3:399A, A3 nm, Adrians wikiname, Ahalwright, Alex R S, Andre.holzner, AndrewHowse, Ap, Artem M. Pelenitsy, B4hand, Ben Standeven, Bender2k14, Bovineboy2008, Brion VIBBER, CBM, CRGreathouse, Cachedio, Canadadiune, Cedtrio, Chalst, ChangChienFu, Chinju, Chris the speller, ChrisGualtieri, Clement Cherlin, Conversion script, Creidieki, DBeyer, DFRussia, Damian Yerrick, Dan Gluck, David.Monnaux, Dcoetzee, Dejan Jovanović, Dennis714, Diego Moya, Djhulme, Doomdayx, Dratman, Drbrznejev, Dwheeler, Dysprosia, Electron9, Elias, Enniles, Everyking, Fancieryu, Favonian, FlashSheridan, Fratrep, Gdr, Giftlite, Graham87, Gregbard, Guarani.py, Gwern, Hans Adler, Hattes, Hh, Hohohob, Igor Markov, J. Finkelstein, Jan Hidders, Janto, Javalenok, Jecar, Jimbreed, Jochen Burghardt, Jok2000, Jon Awbrey, Jowa fan, Jpbowen, Julian Mendez, Juliusbier, Karada, Karl-Henner, Kingpin13, LC, Leibniz, Linas, Localzuk, Logan, LokiClock, LouSheffer, Luuva, Magnus, Mamling, Markhurd, MathMartin, Max13, McCart42, Mellum, Mets501, Mhym, Michael Hardy, Michael Shields, Milimetr88, Miym, Mjuarez, Mn100, Mqasem, MrX, Msoos, Musatovatattdotnet, Mutilin, Naddy, Nameless23, Naturalog, Nealmcb, Night Gyr, Nilmerg, Obradovic Goran, Oerjan, Oliver Kullmann, Orange Suede Sofa, PerryTachett, Piyush Sriva, PoeticVerse, PsyberS, Quaternion, RG2, Rafaelgm, Saforest, Sam Hocevar, Sarah90, Schneelocke, Sergei, Simon04, SiraRaven, Siteswapper, Sl, Sravan11k, SurrealWarrior, The Anome, Tigerauna, Tijfo098, Tim Starling, Timwi, Tizio, Twanyl, Twiki-walsh, Vaxquis, Vegasprof, Weichaoliu, Wik, Wiki.Tango.Foxtrot, Wvbailey, Yaxy2k, Ylloh, Yury.chebiryal, Ywro, Zander, Zarrapastro, Zeno Gantner, ZeroOne, Stanislav, 178 anonymous edits

Boolean-valued model *Source:* <https://en.wikipedia.org/w/index.php?oldid=544134059> *Contributors:* BD2412, CBM, Charles Matthews, Giftlite, Gregbard, Kiefer.Wolfowitz, Marcos, Mets501, Mhss, Michael Hardy, Newbyguesses, R.e.b., Ryan Reich, TakuyaMurata, Tobias Bergemann, Trovatore, Zero sharp, 5 anonymous edits

Booleo *Source:* <https://en.wikipedia.org/w/index.php?oldid=553361412> *Contributors:* Duckniish, Gregbard, Jesse V., Jranbrandt, LilHelpa, MrNiceGuy1113, Nemesis63, Shevek1981, Wcherowi, 2 anonymous edits

Chaff algorithm *Source:* <https://en.wikipedia.org/w/index.php?oldid=481480548> *Contributors:* Andreas Kaufmann, Banes, Bsilverthorn, CBM, David Eppstein, Hermel, Jpbowen, McCart42, Mets501, NavarroJ, Pgr94, Salix alba, Stephan Schulz, Tizio, 4 anonymous edits

Correlation immunity *Source:* <https://en.wikipedia.org/w/index.php?oldid=544310328> *Contributors:* Apokrif, Michael Hardy, Ner102, Pascal.Tesson, Rjwilmsi, 6 anonymous edits

Davis–Putnam algorithm *Source:* <https://en.wikipedia.org/w/index.php?oldid=551278706> *Contributors:* Acipsen, Algebraist, Andreas Kaufmann, Ary29, C S, CBM, CRGreathouse, David Eppstein, Fawcett5, Freak42, Fuenfundachtzig, Gdm, Gregbard, Hans Adler, Iannigb, Jon Awbrey, Jpbowen, Linas, McCart42, Michael Hardy, Mo ainm, MostlyListening, Pgr94, R'n'B, Rrigr, Salgueiro, Silverfish, Simeon, Stephan Schulz, Tijfo098, Tizio, 7 anonymous edits

DPLL algorithm *Source:* <https://en.wikipedia.org/w/index.php?oldid=571139181> *Contributors:* 2404:2000:2000:5:0:0:C1, Ale jr, Andreas Kaufmann, Arbor, Barrkel, Bergsten, CBM, CavalloRazzo, CharlesGillingham, ChrisGualtieri, Creidieki, Gregbard, Jpbowen, Karl Streetmann, Linas, Mets501, Michael Hardy, Ntmatter, Phil Boswell, Rocketrod1960, Rodney Topor, STHayden, Salgueiro, SamBuss, Simeon, Skapur, SocratesJedi, Stephan Schulz, Tijfo098, Tizio, TomT0m, Vegaswikian, Юлия Безрукова, 24 anonymous edits

Formula game *Source:* <https://en.wikipedia.org/w/index.php?oldid=392119594> *Contributors:* Alai, Bearcat, Complex (de), Deutschgirl, ForgeGod, Gregbard, Michael Hardy

Join (sigma algebra) *Source:* <https://en.wikipedia.org/w/index.php?oldid=506222577> *Contributors:* Linas, Tsirel

Logic alphabet *Source:* <https://en.wikipedia.org/w/index.php?oldid=576241870> *Contributors:* Algotr, Alksentrs, BD2412, Cactus.man, Danger, EAderhold, Ettrig, Gamewizard71, Giftlite, Gregbard, Kenyon, LittleWink, Mate2code, Michael Hardy, Nick Number, Oleg Alexandrov, PamD, R'n'B, ResidueOfDesign, Rich Farmbrough, Romicron, Saeed.Veradi, Slyplace, Smerdis, Topbanana, Trovatore, WilliamBHall, 12 anonymous edits

Logic redundancy *Source:* <https://en.wikipedia.org/w/index.php?oldid=550535495> *Contributors:* Cburnett, Chris the speller, Giftlite, Mart22n, Michael Hardy, Nurg, TeamX, The Thing That Should Not Be, WimdeValk, 5 anonymous edits

Logical matrix *Source:* <https://en.wikipedia.org/w/index.php?oldid=568770678> *Contributors:* -Midorihana-, Aksi great, AugPi, BABBUØ, Bare In Mind, Blanchardb, Breggen, Brigit Zilwaukee, Brusegadi, Buchanan's Navy Sec, C.Fred, CBM, Carlosuarez46, Cerberus0, Cliff, Closedmouth, DABBUØ, DEBBUØ, David Eppstein, Deyvid Setti, DoubleBlue, Doubtentry, Education Is The Basis Of Law And Order, El C, Floquenbeam, Flower Mound Belle, Happy-melon, Hut 8.5, InverseHypercube, Jeffrey O. Gustafson, Jheiv, Jon Awbrey, JzG, Kimmy007, Lambiam, Marsboat, MaxSem, Mets501, Mrs. Lovett's Meat Puppets, Nihilires, Oleg Alexandrov, Overstay, Paul August, Poke Salat Annie, Preveiling Opinion Of Dominant Opinion Group, RABBÜ, RABBUØ, REBBÜ, REBBUØ, RxS, Seb26, Slakr, TeaDrinker, Tijfo098, Unco Guid, Wknigh94, Wolf of the Steppes, Yolanda Zilwaukee, Михајло Анђелковић, 16 anonymous edits

Logical value *Source:* <https://en.wikipedia.org/w/index.php?oldid=217011121> *Contributors:* Alansohn, Andrewaskew, Apokrif, ArkadiuszGlowaLaskowski, BD2412, Bjankuloski06en, Byelf2007, CBM, CRGreathouse, Card Zero, Chalst, Charles Matthews, Cyro, Dbtfz, EmilJ, ErikDunsing, FatherBrain, Frap, Frege, Giftlite, Gregbard, Gveret Tered, Hans Adler, Incnis Mrsi, Jackmass, Jon Awbrey, Josh Parris, Kephir, Kuru, LBehounek, Letranova, Matchups, Mate2code, Mav, Mayumashu, Mhss, Michael Hardy, Nakon, Noamz, Octahedron80, Patrick, R'n'B, Rami radwan, Rich Farmbrough, Senu, Sethmahoney, Simeon, SlimVirgin, Solomonfromfinland, T-9000, TakuyaMurata, Thabick, Thecameraguy12345678, Tijfo098, Tobias Bergemann, Toby Bartels, Tomisti, Trovatore, WhyBeNormal, Wvbailey, XcepticZP, Zagothal, 28 anonymous edits

Modal algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=545321861> *Contributors:* EmilJ, Mhss

Petrick's method *Source:* <https://en.wikipedia.org/w/index.php?oldid=544299564> *Contributors:* Andrei Stroe, Fresheneesz, Harry0xBd, Jay Uv., Kulp, Michael Hardy, Mindmatrix, Njoutram, Oleg Alexandrov, Paul August, Timeroot, Tizio, Trovatore, Willem, Wolfmanx122, 13 anonymous edits

Product term *Source:* <https://en.wikipedia.org/w/index.php?oldid=555736427> *Contributors:* Alai, CBM, Materialsscientist, Mets501, Oleg Alexandrov, Trovatore, 3 anonymous edits

Propositional formula *Source:* <https://en.wikipedia.org/w/index.php?oldid=576709164> *Contributors:* Adelpine, Arch dude, BD2412, Billinghurst, Bookandcoffee, CBM, Chris the speller, ChrisCork, Colonies Chris, Djihed, Filemon, Giftlite, Golbez, Gregbard, Hairy Dude, Happy-melon, Hmains, Iridescent, Jaded-view, Jochen Burghardt, Jon Awbrey, Julian Mendez, Kbdank71, Kevin Gorman, Klparrot, Kwiki, Lambiam, LlHelpa, Linas, Mark viking, Michael Hardy, Mild Bill Hiccup, Mindmatrix, Muhammad Hamza, Neuralwarp, Neurolysis, Nick Number, Open2universe, PWilkinson, PhnomPencil, R'n'B, Raise exception, Rjwilmsi, Tabletop, The Evil IP address, Timrollpickering, Tobias Bergemann, WRK, Wizard191, Wolfmanx122, Woohookitty, Wvbailey, 17 anonymous edits

Stone duality *Source:* <https://en.wikipedia.org/w/index.php?oldid=548861954> *Contributors:* BeteNoir, Charles Matthews, Chinju, D6, Deltahedron, Erudescorp, Gauge, Giftlite, Honestosewater, Lethe, Marcosaedro, Markus Krötzsch, Mets501, Michael Hardy, Michael Kinyon, Nick Number, Noisy, Sodin, Tijfo098, Trovatore, Vanish2, Viridae, Vivacissamamente, Zundark, 7 anonymous edits

Stone functor *Source:* <https://en.wikipedia.org/w/index.php?oldid=570562135> *Contributors:* AtheWeatherman, Bearcat, Blanchardb, John Z, Porton, Shawn in Montreal, TenPoundHammer, 8 anonymous edits

True quantified Boolean formula *Source:* <https://en.wikipedia.org/w/index.php?oldid=541465038> *Contributors:* Charles Matthews, Creidieki, David Eppstein, Drae, Edward, EmilJ, ForgeGod, Giraffedata, Gregbard, Ilia Kr, Karmastan, Khazar2, Kusma, Lambiam, Michael Fourman, Michael Hardy, Milimetr88, Misterspock1, Miym, Neile, Radagast83, RobinK, Spug, Twin Bird, 23 anonymous edits

Truth value *Source:* <https://en.wikipedia.org/w/index.php?oldid=576054871> *Contributors:* Alansohn, Andrewaskew, Apokrif, ArkadiuszGlowaLaskowski, BD2412, Bjankuloski06en, Byelf2007, CBM, CRGreathouse, Card Zero, Chalst, Charles Matthews, Cyro, Dbtfz, EmilJ, ErikDunsing, FatherBrain, Frap, Frege, Giftlite, Gregbard, Gveret Tered, Hans Adler, Incnis Mrsi, Jackmass, Jon Awbrey, Josh Parris, Kephir, Kuru, LBehounek, Letranova, Matchups, Mate2code, Mav, Mayumashu, Mhss, Michael Hardy, Nakon, Noamz, Octahedron80, Patrick, R'n'B, Rami radwan, Rich Farmbrough, Senu, Sethmahoney, Simeon, SlimVirgin, Solomonfromfinland, T-9000, TakuyaMurata, Thabick, Thecameraguy12345678, Tijfo098, Tobias Bergemann, Toby Bartels, Tomisti, Trovatore, WhyBeNormal, Wvbailey, XcepticZP, Zagothal, 28 anonymous edits

Vector logic *Source:* <https://en.wikipedia.org/w/index.php?oldid=575735278> *Contributors:* Almadana, Josve05a, Michael Hardy, 14 anonymous edits

Binary decision diagram *Source:* <https://en.wikipedia.org/w/index.php?oldid=575615209> *Contributors:* Ajo Mama, Amccosta, Andreas Kaufmann, Andris, AshtonBenson, Behandeeem, Bigfootbigfoot, Bluemoose, Bobke, Boute, Brighterorange, Britlak, Calabe1992, Charles Matthews, ChrisGualtieri, David Eppstein, David.Monniaux, Denim1965, Derek farn, Dirk Beyer, Divy.dv, EmilJ, Greenrd, GregorB, Gtrmap, Happyuk, Hermel, Heron, Hitanshu D, J04n, Jason Recliner, Esq., Jay.Here, Jcarroll, Jkl, KSchutte, Kakesson, Karlbrace, Karlkt, Karsten Strehl, Kinu, Laudaka, Lone boatman, LouScheffer, Matumio, McCant42, Mdd, Michael Hardy, Nouiz, Ort43v, Pce3@ij.net, Qwertyus, Rohit.nadig, Rorro, Ruud Koot, Ryan Clark, Sade, SailorH, Sam Hocevar, Sirutan, Sun Creator, Taw, Tijfo098, Trivialist, Trovatore, Twri, Uli, YHoshua, YamnTM, 122 anonymous edits

Implication graph *Source:* <https://en.wikipedia.org/w/index.php?oldid=409100328> *Contributors:* Altenmann, CBM, David Eppstein, DavidCBryant, GregorB, Thisisraja, Twri, Vadmium, 3 anonymous edits

Karnaugh map *Source:* <https://en.wikipedia.org/w/index.php?oldid=577993669> *Contributors:* AHA.SOLAX, Abed pacino, Acerperi, Ademkader, Aeozza, Aervanath, AgadaUrbanit, Alessandro.goullart, Amplitude101, Ancheta Wis, Anomie, Arichnad, Augustojd, Avoided, B-Con, B.Zsolt, BL, Biker Biker, BillNace, Bogdangiusca, Bonzo, Bovlb, Bryan Derksen, Bukwoy, CARPON, CRGreathouse, Caesar, Carrige, Cburnett, Ceklock, Chadloeder, Chan siuman, CharlotteWebb, ChyranandChloe, Ckape, Colin Marquardt, Cremepuff222, Cyberjoac, Czar Kirk, Czarkoff, Danim, Dannamite, Dcarter, Delaszk, Dionyziz, Discospinster, Dmenet, Ebojyr, EuroCarGT, Eyreland, FreddieRic, Freewol, Fresheneesz, Fritzpoll, Fubar Obfusc, Furykef, FxBit, GRAHAMUK, Gantoniu, Geiriani, Giftlite, Goat-see, Grunt, Gulliveig, HLwiKi, Hana Adler, HereToHelp, Heron, Hex4def6, Hhedeshed, Icigic, Imyourfoot, InverseHypercube, Julian, Jake Wartenberg, Jitse Niesen, Jk2q3jrkls, Jmangold, Jmgonzalez, Jochen Burghardt, JoeFloyd, Jokes Free4Me, Jonnie5, JustAGal, Justin Johnson, Justin W Smith, Kartano, Katragadda465, Kelum.kosala, Kenyon, Kvtoelker, LA2, LeCire, Leuko, Ligulem, Locriani, Macjohn2, Macrakis, MagnaMopups, MarSch, MartinPackerIBM, Mate2code, Mdcoope3, Mdd, MeltBanana, Mhayes46, Michael Hardy, Michael.Pohoreski, Mike Segal, Miym, Mobius, Moonshiner, Mrphious, Murtasa, Naddy, Nbeverly, Ndgyu, Nigel, Njoutram, Norlik, Nveitch, Oblivious, Obradovic Goran, Omegatron, Paul Murray, Pdebone, Pearle, Phyzone, PierreAbbat, Pinethicket, Pitel, Plugwash, R'n'B, RG2, RUL3R, RazoreICE, RazoreRobin, Reywas92, Rjd0060, Rocketrod1960, Rohanmittal, RolandYoung, Rrfwki, SamB, Seav, Seprax, Shidh, Sigra, Sinan Taifour, Snowolf, SpaceFlight89, Spudpuppy, SpuriousQ, Svick, SwisterTwister, Tardis, Texture, The gulyan89, Thumperward, Thunder Wolf, Tkynerd, Trovatore, Tullywinters, TunLuek, Unstable-Element, Usmanrazaz, Utility Knife, Vaceituno, Villarinho, Vobrez, Voomoo, Vrenator, Widr, Wikiklrc, WimdeValk, Winhunter, Wolfmanx122, Writer130, Wtshmanski, Wvbailey, Yaxim, Yim, ZeroOne, Zundark, 335 anonymous edits

Propositional directed acyclic graph *Source:* <https://en.wikipedia.org/w/index.php?oldid=561376261> *Contributors:* Aagtbdoua, Andreas Kaufmann, BD2412, Brighterorange, DRap, Dennis714, Mvinyals, Nbarth, RUN, Selket, Tijfo098, Trovatore, 4 anonymous edits

Quine–McCluskey algorithm *Source:* <https://en.wikipedia.org/w/index.php?oldid=577146512> *Contributors:* AgadaUrbanit, Akshayrsrinivasan, Alansohn, Alessandro.goullart, Allan McInnes, Amccosta, Andionita, Andrew Bunn, AugPi, BBar, CARPON, Ceklock, Charles Matthews, Chetvorno, ChrisGualtieri, CiaPan, Clementina, Cybercobra, Dar-Ape, Dcoetzee, Dfass, Durova, Dusa.adrian, Dusadrian, Dysprosia, Elanthiel, Fresheneesz, Giftlite, Gilliam, Gregbard, Gulliveig, Gulyan89, Gzornenplatz, HHadavi, Huntscorpio, Hv, Infrangible, Iridescent, James Foster, Jay Uv., Jim.henderson, Jkl, John of Reading, Johnibby, Jon Awbrey, Kobold, Lithiumhead, Loopers920, McCart42, Mhss, Michael Hardy, Modify, Narfananor, Njoutram, Noeckel, OneWorld22, Piet Delport, Potopi, Pqrstuv, QuiteUnusual, RJFJR, Ra2007, Ralph Corderoy, RedLunchBag, Revision17, Roachmeister, Romanski, Salgueiro, Simoneau, Skryskalla, TakuyaMurata, Trovatore, Two Bananas, Wlk13rh3nry, Wh1chwh1tch, Wikibofh, WimdeValk, Woohookitty, Ysangkok, 142 anonymous edits

Reed–Muller expansion *Source:* <https://en.wikipedia.org/w/index.php?oldid=564919131> *Contributors:* DavidCBryant, Fyrael, Jason Recliner, Esq., Macha, Macrakis, RobinK, 3 anonymous edits

Truth table *Source:* <https://en.wikipedia.org/w/index.php?oldid=575840699> *Contributors:* 65.68.87.xxx, Abd, Achal Singh, Addihockey10, Aeroknight, AgadaUrbanit, Aitias, Akuindo, Alansohn, Ancheta Wis, Andres, Annina.kari, Antandrus, Antonelli, ArnoldReinhold, Aston Martian, AugPi, Avaya1, BD2412, Banno, Bdesham, Beetstra, Bergin, Bgwhite, Billymac00, Bluemoose, Bookandcoffee, Bryan Derksen, 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, Cybercobra, Danlev, Dcoetze, Delirium, DesertSteve, Dicklyon, Djayjp, Dr Smith, Dysprosia, Flowerpotman, Fox89, Frances, Fresheneesz, Gaicacara, Gary King, Gershwinrb, Ghettoblaster, Giftlite, Graham87, Gregbard, Gschadow, Hans Adler, Hephasto, Heron, Holly golightly, InverseHypercube, IfxId4, JDP90, JVz, Jason Quinn, JimWae, Jimfbleak, Jk2q3jrkls, Johnbrownbody, Jon Awbrey, Jonsafari, Joyous!, Julian Mendez, Justin Johnson, K ganju, Kbdk71, Kiril Simeonovski, Kitty Davis, KnightRider, KnowledgeOfSelf, Krawi, Kyle Barbour, Larry Sanger, Lee, Lethe, Letranova, Liftarn, Lights, Logan, Lst27, LutzL, Markhurd, Mastrausa, Mav, Mdd, Mts501, Mhss, Michael Hardy, Millermk, Mindmatrix, Nakke, Noosphere, Nortexoid, Olaf, Oleg Alexandrov, On This Continent, Oreopriest, Pakaran, Parikshit Narkhede, Patrick, Paul August, Policron, Pooryorick, Quad4rax, Quinto, Qwfp, R27smith200245, Racconish, RedWolf, Rich Farmbrough, Rofthorax, Santiago Saint James, Schnits, Sevillana, Sir48, Slazenger, SpuriousQ, Starylon, TBloemink, Tangotango, Tarquin, The Rhymesmith, The Tetras, Tijfo098, Trovatore, Utren, Vadmium, Vegetator, Vilarage, Wavelength, Wbm1058, Webmaestro, WikiPuppies, WimdeValk, Wolfmanx122, Wshun, Wstorr, XTerminator2000, Xxp0r, 407 anonymous edits

Venn diagram *Source:* <https://en.wikipedia.org/w/index.php?oldid=577034314> *Contributors:* 041744, 2help, 47b, AVand, Acondolu, Adam Zábranský, Addshore, AdjustShift, Aillema, Alansohn, Alchemistmatt, Alister MacLeod, Almacha, AnonGuy, Anonymous Dissident, Anshuk, Antonio Lopez, Art LaPella, Arthur Rubin, Arunsingh16, AugPi, AySz88, Azurengar, BMF81, Bachcell, Badgernet, Barneca, BarrelProof, Bbukh, Belinrahs, Ben Ben, Bhanks, Blainster, BoomerAB, Brent Gulanowski, Brick Thrower, Bryan Derksen, Bubbha, CBM, CWenger, Cadadril, Calvin 1998, Camembert, Capricorn42, Carewsr, Ceyockey, Chadparker, Chalst, Chan siuman, Charles Matthews, Charleskiss, Ched, Chetvorno, Christopherlin, Classicaeon, Cmglee, Cntrns, ConradPino, Crissov, CyberCrone, DBaba, DVdm, Dagme, Danger, Dangerousnerd, Daniel5127, Dariusz wozniak, DarylNickerson, Dave.Dunford, David Eppstein, Ddon, Deb, Den fjärrtrade ankan, DennisDaniels, Derschmitty, Diagram01, Dina, Discospinster, Disucks, Docey, Doprendek, Doradus, Doug, Dream out loud, Duncumcumming, Duoduo, Ebelerular, EdH, Edonovan, Elecbullet, Equendil, Excirial, Falcon8765, Fanyavizuri, Favonian, Fbfifriday, Frank Romein, Freshbakedpie, Fumblebruschi, GEGranato, GLPeterson, Gail, Gawaxay, GemStapleton, Georg-Johann, Ghetteaux, Giftlite, Giro720, Gogo Dodo, Gregbard, Ground Zero, Gwernol, Hannes Röst, Hans Adler, Happy-melon, Hede2000, Heron, Hu Gadarn, Hu12, Hyacinth, Illia Connell, Infovarius, Insanity Incarnate, Interiot, Isnow, IfxId4, J.delanoy, JForget, Jae481, JamesBWatson, Jbourjif, Jeff Muscato, Jennavecia, Jeodesic, Jhbdl, Jhenderson777, Jimaginator, Jimjanjak, Jmunge, JodyB, Joelm, John Broughton, John Cline, Johnfin, Jon Awbrey, Jonnytaylor, Justin Johnson, Katalaveno, Katieh5584, Kevin B12, Khullah, Khvalamde, Kjetil, Kompik, Kopophex, Krunk, Kvng, Kwantus, Lambiam, LennartBolks, Liam Skoda, Ligulem, Ltickett, Luckyluke, Lumingz, MER-C, Madsci guy, Manco Capac, Marek69, Marekchik, Masgatotkaca, Mate2code, Matthew Fennell, MattieTK, Mdd, Megaman em, Melchoir, Melcombe, Mhss, Michael Hardy, Midnightcomm, Mikeblew, Mindmatrix, MisterSheik, Mmarchin, MoatazKob, Mojgan mirzaie, Mojo Hand, MrOllie, Narayan3210, NawlinWiki, Nbarth, Neverquick, Nightstallion, Numbermaniac, Nyttend, ONEder Boy, Oleg Alexandrov, Owenozier, Parent5446, Patstuart, Paul August, Paul Magnusson, Paul Matthews, Pelge, Persian Poet Gal, Peter Rodgers, Phancy Physicist, Pharaoh of the Wizards, PhilKnight, Philip Trueman, Phoenixrod, Pinkbeast, Plantperson, PleaseStand, Pmc, Pointillist, Policron, Potatomonk, Quantar, RDBrown, Raven in Orbit, Reaper Eternal, Richard001, Richie, Rjwilmsi, Robinh, Roeeyaron, Rojomoke, Ronz, Rpresser, Rrnr, Rsppeer, RupertMillard, Russeltarr, Ruud Koot, SMC89, Salix alba, Sarfa, Shbowside, SchuminWeb, Schutz, SeanWillard, Shenne, Shuiipzv3, Shyamgadhia28, SilverStar, SimonArrott, SiobhanHansa, Smjg, SoWhy, SomeStranger, Sozertsat, Squiddy, Stephenb, Steverapaport, Stevertigo, Stractive, Super Sam, Tarobon, Tarotcards, Tarquin, Tentinator, Terrysolid, Teivilo, Texture, The Anome, The Thing That Should Not Be, Theaterfreak64, Tide rolls, Timwi, Toa0707, ToasteIL, Todds1, Tomwsulcer, Trovatore, Trusilver, Ttwo, Uncle Dick, Unitknow2, Unschoool, Vanished user 34958, Verne Equinox, Waldir, Wdflake, Werdna, West.andrew.g, Wikipediaatrix, Wvbailey, Xander, Yero, Yintan, Youssefian, ZimZalaBim, Zipzipip, Zmag, Zscout370, Ταοοὐλα, Ο ὀλότρος, 587 anonymous edits

Ampeck *Source:* <https://en.wikipedia.org/w/index.php?oldid=486672982> *Contributors:* A:-)Brunuš, Adrianwn, ArnoldReinhold, BD2412, Bjankuloski06en, Bookandcoffee, Branzillo, Bryan Derksen, CBM, Cameronc, Chris the speller, Colin Marquardt, Cícero, Dega180, Dijxtra, Dogah, Dreamyshade, Edward, EmilJ, Emvee, Francvs, Furykef, Gamewizard71, GeorgBauhaus, Gregbard, Grendelhan, Gubbub, Hans Adler, Hermense, Hut 8.5, Igor Yalovecky, Igotta Lemma, Iolar, IronMaidenRocks, Iserdo, IfxId4, JMRyan, Jallan, Jean-Frédéric, Jerzy, Jon Awbrey, Kbdk71, Krymzon, Lantern Leatherhead, Loadmaster, Mate2code, Maxim Razin, Meisam, Melchoir, Memento Vivere, Metadjin, Mhss, Mindmatrix, MrOllie, Nbarth, Nesarose, Nortexoid, Ohnoitsjamie, Pangur Ban My Cat, Paul August, Policron, Prestonmag, Pukkie, Queen of the Dishpan, Qwertyus, RokerHRO, Santiago Saint James, Sietse Snel, SirPeebles, SocratesJedi, Stevertigo, Stimp, The Tetrast, Tijfo098, Trovatore, Tyomitch, UU, Urhixidur, Vujke, WarrenPlatts, Webber Jocky, 59 anonymous edits

Balanced boolean function *Source:* <https://en.wikipedia.org/w/index.php?oldid=545743743> *Contributors:* Allens, AvicAWB, Bearcat, Fetchcomms, Govind285, Gzorg, Int80, Lobopizza, M.r.ebrahimi, Prakhar098, Qetuth, Ryanguyaneseboy, Snow Blizzard, Uncle Milty, Xezbeth, 3 anonymous edits

Bent function *Source:* <https://en.wikipedia.org/w/index.php?oldid=511814479> *Contributors:* Anonymous Dissident, David Eppstein, Gzorg, Marozols, Mate2code, Nageh, Ntsimp, Phil Boswell, Rich Farmbrough, Rjwilmsi, Will Orrick

Boolean algebras canonically defined *Source:* <https://en.wikipedia.org/w/index.php?oldid=562163237> *Contributors:* Arthur Rubin, Assulted Peanut, Barticus88, Bte99, CBM, D6, Dabomb87, David Eppstein, EmilJ, Fratrep, Gregbard, Hairy Dude, Hans Adler, Headbomb, Irmy, Jeepday, Jon Awbrey, KSmrq, Lambiam, Linas, Michael Hardy, Michael.Deckers, Nick Number, Paul August, Pcap, Pmanderson, R'n'B, Rjwilmsi, Robertvan1, Salix alba, Set theorist, Srice13, The Tetrast, Tijfo098, Tobias Bergemann, Tommy2010, Vaughan Pratt, Woohookitty, Zundark, 26 anonymous edits

Boolean function *Source:* <https://en.wikipedia.org/w/index.php?oldid=575343139> *Contributors:* Allanthebaws, AltiusBimm, Arthena, Ayda D, BigrTex, Bjankuloski06en, Charles Matthews, CharlesC, Chillout003, Ciphergoth, Clemwang, Eassin, Farisori, Gadfium, Gene.arboit, Giftlite, Gregbard, Hans Adler, Int80, Jiri 1984, Jok2000, Jon Awbrey, Kumiko (renamed), Liquidborn, Loadmaster, Mate2code, Matt Crypto, Mhss, Michael Hardy, Michael Snow, Mindmatrix, Murtasa, Nageh, Neilc, Ner102, Ntsimp, Oleg Alexandrov, Omnipaedista, Patrick, Pce3@ij.net, Poa, Policron, Qwertyus, SDS, Shyguy92, Sivan.rosenfeld, Spinningspark, Steveprutz, Tatrgel, TheKoG, Theorist2, TreasuryTag, Trovatore, Trusilver, Twri, Waldir, WikiPuppies, 45 anonymous edits

Boolean-valued function *Source:* <https://en.wikipedia.org/w/index.php?oldid=530986513> *Contributors:* AK456, Arthur Rubin, AutocraticUzbek, BD2412, Bibliomaniac15, BrigitZilwaukee, Buchanan'sNavySec, C.Fred, CBM, CRGreathouse, ChesterCountyDude, ChrisTheSpeller, Closedmouth, Coredesat, DelawareValleyGirl, DoubleBlue, Doubtentry, Dougher, ElCFlower, MoundBelle, Giftlite, GogoDodo, Gregbard, HansAdler, Hut8.5, IcharusIxiion, Jamelan, JeffreyO.Gustafson, Jim62sch, JonAwbrey, JzG, Linas, Marsboat, MathMartin, Mhss, Mrs.Lovett'sMeatPuppets, NavyPierre, OlegAlexandrov, Omnipaedista, Overstay, PokeSalatAnnie, Salixalba, Samppi111, Sdorrance, Seb26, Slakr, SoutheastPennPoppa, Tijfo098, TobyBartels, Trovatore, Versageek, VivaLaInformationRevolution!, WimdeValk, WolfoftheSteppes, YolandaZilwaukee, 11anonymous edits

Conditioned disjunction *Source:* <https://en.wikipedia.org/w/index.php?oldid=577876317> *Contributors:* BD2412, CBM, DavidWBrooks, Elonka, Gregbard, GregorB, Lambiam, MatthewKastor, Nortexoid

Converse implication *Source:* <https://en.wikipedia.org/w/index.php?oldid=544842594> *Contributors:* BD2412, CBM, DanteCardosoPintoDeAlmeida, DavidEppstein, Francvs, Gregbard, GregorB, HansAdler, Kbdank71, Mate2code, Meisam, Urhixidur, 8anonymous edits

Converse nonimplication *Source:* <https://en.wikipedia.org/w/index.php?oldid=546749136> *Contributors:* CBM, DavidEppstein, EmilJ, Francvs, Gregbard, GregorB, Kbdank71, Mate2code, Metadjinn, MichaelHardy, MinawaYukino, RichFarmbrough, Roshan220195, Stormbay, SupperNope, Tikurion, Vegaswikian, 3anonymous edits

Evasive Boolean function *Source:* <https://en.wikipedia.org/w/index.php?oldid=545859781> *Contributors:* DavidEppstein, Mate2code, MichaelHardy, MildBillHiccup, MuffledThud, SivanRosenfeld, Xmn, וְנִילָה

Exclusive or *Source:* <https://en.wikipedia.org/w/index.php?oldid=577722589> *Contributors:* 16@r, Acwilson9, Aeons, Aeoma, Amire80, Andre.holzner, AnonMoos, AxelBoldt, BD2412, Barryb, Beamishboy, Bjankuloski06en, BlackTerror, Blotwell, Bob.v.R, Boing! said Zebedee, Bookandcoffee, Btx40, C.A.Russell, CBM, CRGreathouse, Centrx, CharlesMatthews, CommonsDelinker, Creidieki, Cybercobra, DaNuke, Daverocks, DavidEppstein, Dcoetze, DePiep, Deflective, Delos, DennisMalandro, DerHexer, DerekR.Bullamore, Dijxtra, Dnas, DopefishJustin, Download, DrJolo, Dwheeler, Eassin, Elemeno, ErikZachte, Feb30th1712, Fibonacci, FiremanBiff, FlashSheridan, Foobar, Francvs, Fredrik, Fresheneesz, Furykef, Gerbrand, Giftlite, GolemUnity, Gongshow, Goodnightmush, GraemeLeggett, Graue, Gregbard, Gscshoyru, Gurch, H2o8polo, Hamaryns, HansAdler, Helix84, Henrygb, Heron, Hfastedge, Horaceli, Htmlland, Hvn0413, Hwasungmars, IMSO, IanManka, IgorYalovecky, Imroy, Indefatigable, Inductiveload, Irizedent, Iwwwwii, Jafet, Jarble, JesseViviano, Jfrascencio, Jjbeard, Jobarts, JokesFree4Me, JonAwbrey, Jsmethers, JustinWSmith, Karada, KayDekker, Kbdank71, Knucmo2, Kwamikagami, L3prador, LOL, LarryV, LedgendGamer, LingKahJai, Loadmaster, Looxix, Lousyd, Malcohol, Manderr, Marksmon, Marmor10, Mate2code, Midito, Meisam, Melab-1, Melchoir, Merovingian, MichaelHardy, Mike92591, Miserlou, Mjaked, Munibert, N-double-u, Najeeb1010, Nickdc, Nixdorf, Nthomas, Ohnoitsjamie, Olathe, OlegAlexandrov, Origin415, Pandamonnia, Patstuart, PaulAugust, PetScarecrow, Philopedia, Plugwash, Priceman86, Pulu, PurplePlatypus, Qooooooo, Quackor, Quuxpluseone, R'n'B, R.S.Shaw, RG2, Ram434, Reisio, RichardB.Frost, Richlumaiu, RockMagNetist, Rosenkreuz, RoxanneFeline, SJP, Samboy, Samppi111, SantiagoSaintJames, Sbisolo, Sderose, Separa, Shandris, SiBr4, Silverfox11202, Snafflekid, SnorlaxMonster, SocratesJedi, Spoon!, Square87, Stevertigo, Sugarskane, Supuhstar, TSO1D, Taak, Taka, TakuyaMurata, TedColes, TheTito, Themania, Thumperward, Tijfo098, TobiasBergemann, TobyBartels, Todd434, Tokek, Tony414, ToobMug, Trovatore, UU, VKokielov, VladikVP, Wafulz, Wavelength, Winterheart, Wjl, Xenfreak, Xorlogician, Ziyaeen, 애멜루지로, 239anonymous edits

False (logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=570480529> *Contributors:* Eyesnore, Gregbard, Hvn0413, IncnisMrsi, IronGargoyle, Lugia2453, Mcoupal, ShelfSkewed, TobyBartels, 19anonymous edits

Functional completeness *Source:* <https://en.wikipedia.org/w/index.php?oldid=575967711> *Contributors:* Abazgiri, AnchetaWis, CBM, CBright, Cnoguera, Dixtosa, Domster, EmilJ, FMasic, Francvs, Gregbard, Guppyfinsoup, HansAdler, Infwl, InverseHypercube, Jamesfisher, JoshuaIssac, Kaldari, Kbdank71, Krauss, LOL, MarSch, MichaelHardy, Nortexoid, PaulMurray, Paxsimius, Qwertys, R.e.s., RichF, SaraleeArrowoodViognier, SergeyMarchenko, Slrubenstein, Swpb, Tijfo098, 41anonymous edits

If and only if *Source:* <https://en.wikipedia.org/w/index.php?oldid=575485422> *Contributors:* ABCD, Abolen, Abyaly, Adcx, AdamConover, Adam78, Adjam, AgentPeppermint, Andres, AnonymousDissident, Ark, ArthurRubin, AstroHurricane001, BANZ111, BMF81, Brianjd, Bryanburgers, CBM, CRGreathouse, Camembert, CarrieVS, Cathfolant, Chalst, Chaszzbrown, Chinasaurs, Chmarkine, Chirico, Ciphers, Conversionscript, DamianYerrick, Danielpi, DanteAlighieri, DavidCary, Davkal, Delirium, Diberri, Dicklyon, Dominus, DopefishJustin, Dysprosia, EdwardZ.Yang, Egriffin, Ekevu, ElPolloDiablo, Elwikipedista, Enochlau, Eu.stefan, Evercat, Evildictator, Ex13, Eyu00, Forderud, Francvs, Geoffrey, Giftlite, GlennL.Gloin, GoOdCoNEnT, Gregbard, Helder.wiki, Henrygb, HolyKnight33, Iamtheadeus, Ichata, IgorYalovecky, IgorekSF, IncnisMrsi, Interiot, InverseHypercube, Itai, Jaquerie27, Jarble, JasonQuinn, Jewbacc, Jkelly, JonatanSwift, Joriki, JoshuaW, KarlDickman, KennethBurke, Kingturtle, KjellG, Kmddmk, LarrySanger, Ledge, Letranova, LiDaobing, Lillingen, Lotje, Master11218, MatthewWoodcraft, McKay, Melchoir, Mellum, Mets501, MichaelHardy, Mikespida, Mindmatrix, Minehava, MouseIsBack, Msh210, Msknathan, Mvc, Najoj, Nejatarn, Negatrapt, Neutrality, Nzll, OlegAlexandrov, Patrick, PaulAugust, Pearle, Peterwhy, Phorrest, PhotoBox, Picaroon, PoponARope, Postclock, Psychonaut, Quintus314, R.e.b., Rainwarrior, Rapsar, Rbonvall, Revolver, Rhialto, Ronjhones, Ropers, Ruakh, RuudKoot, RyanReich, SGBailey, Schneau, Serpent'sChoice, Shirifan, Shoeofdeath, Singularity, Skippydo, Smaug123, Stevenj, Stevertigo, Stillnotelf, Sugarfoot1001, Sunborn, Superfrank, Surfeited, Suruuna, Taak, Tarquin, TheMadBaron, Thumperward, Tijfo098, Timlevin, TinyplasticGreyKnight, TobiasBergemann, TobyBartels, TrippingTroubadour, Ttzz, Urdutex, Urhixidur, UserGoogol, Vanisheduser47736712, Velho, VickiRosenzweig, Voltagedrop, Warman06, WestwoodMatt, Widr, Wikiborg, Wikimichael22, WissenDürster, Wmahan, Woohookitty, Wshun, Wwwolf, Yuide, Zundark, ÅvarArnfjördBjarmason, ЛевДубовой, 165anonymous edits

Inclusion (Boolean algebra) *Source:* <https://en.wikipedia.org/w/index.php?oldid=567164022> *Contributors:* Macrakis

Indicative conditional *Source:* <https://en.wikipedia.org/w/index.php?oldid=567636548> *Contributors:* Aquafleur, ArthurRubin, Bfinn, CharlesMatthews, ChrisGualtieri, Copyman, Donjoe334, Dysprosia, Eluard, Elvarg, Fresheneesz, Gregbard, Hanlon1755, Isseaboor, Jgyorke, Jaymay, KSchutte, Kzollman, MachineElf1735, Mhss, MichaelHardy, MindShifts, Mwilde, OlegAlexandrov, Otr500, Pacogo7, Patrick, PaulAugust, PhilBoswell, Radgeek, Recentchanges, Ryguasu, ServiceableVillain, TakuyaMurata, ÅE, 27anonymous edits

Indicator function *Source:* <https://en.wikipedia.org/w/index.php?oldid=571974793> *Contributors:* Albmont, ArnoldReinhold, AxelBoldt, Banus, BiT, BoJacoby, CSTAR, Centrx, CharlesMatthews, Colinville, DavidEppstein, DavidHouse, Dcoetze, DigbyTantrum, Falcor84, Gauss, Giftlite, Helgus, Ht686rg90, Icarions, JonAwbrey, JordiBurguetCastell, LBehounek, Linas, Martpol, Maschen, MathMartin, Melcombe, Mets501, MichaelHardy, Mornegil, Mpdr1989, NickBush24, NightJaguar, ObradorovicGoran, Octahedron80, OlegAlexandrov, PV=nRT, Paintman, PaulAugust, Pcb21, PhilBoswell, Piotrus, Quietbritishjim, Raystorm, Rjwilmis, Rosh3000, Rutgerjanlange, Salixalba, Salspaugh, Sherbrooke, Smallpotato, Straightontillmorning, Sullivan.t.j, TheAnome, Trovatore, Wvbailey, Xetrov, Zleinft, 49anonymous edits

Logical NOR *Source:* <https://en.wikipedia.org/w/index.php?oldid=573427813> *Contributors:* A:-)Brunuš, Adrianwn, ArnoldReinhold, BD2412, Bjankuloski06en, Bookandcoffee, Branzillo, BryanDerksen, CBM, Cameronc, ChrisTheSpeller, ColinMarquardt, Cicero, Dega180, Dijxtra, Dogah, Dreamyshade, Edward, EmilJ, Emvee, Francvs, Freekh, GaiusCornelius, Giftlite, Gregbard, Grendelkhan, Gubbubu, HansAdler, Hermeneus, Hut8.5, IgorYalovecky, IgottaLemma, Iolar, IronMaidenRocks, Isredo, Ixfd64, JMRyan, Jallan, JeanFrédéric, Jerzy, JonAwbrey, Krymzon, Lanter Leatherhead, Loadmaster, Mate2code, MaximRazin, Meisam, Melchoir, MementoVivere, Metadjinn, Mhss, Mindmatrix, MrOllie, Nbarth, Nesarose, Nortexoid, Ohnoitsjamie, PangurBanMyCat, PaulAugust, Policron, Prestonmag, Pukkie, QueenoftheDishpan, Qwertys, RokerHRO, SantiagoSaintJames, SietseSnel, SirPeebles, SocratesJedi, Stevertigo, Stimp, TheTetra, Tijfo098, Trovatore, Tyomitch, UU, Urhixidur, Vujke, WarrenPlatts, WebberJocky, 59anonymous edits

Logical biconditional *Source:* <https://en.wikipedia.org/w/index.php?oldid=575008221> *Contributors:* IForTheMoney, 777sms, Adambiswanger1, Andycjp, AnonymousDissident, ArthurRubin, BAxelrod, BD2412, Bjankuloski06en, CBM, ChesterMarkel, CommonsDelinker, DEMcAdams, DavidCary, Discospinster, Dysprosia, Elwikipedista, Francvs, Freekh, GaiusCornelius, Giftlite, GoOdCoNEnT, Graymornings, Gregbard, Hanlon1755, Infovarius, InverseHypercube, Jdelaney, JSquish, Jarble, Jittat, JonAwbrey, JulianMendez, Kbdank71, Kuzmaka, Lambiam, Lethe, Letranova, Mate2code, MattGiua, Meisam, Melchoir, Mets501, Mijelliott, MilesAgain, OlegAlexandrov, Patrick, PaulAugust, Pine, Radagast83, Rainwarrior, Recentchanges, Robina, Shirulashem, TakuyaMurata, TinyplasticGreyKnight, Velho, WayneSlam, Wolfrock, WookieInHeat, 49anonymous edits

Logical conjunction *Source:* <https://en.wikipedia.org/w/index.php?oldid=578002875> *Contributors:* 2andrewknayev, ALE!, AdrienChen, Andres, AxelBoldt, B4hand, BD2412, BGSpaceAce, Bluemoose, Brockert, Bytner, Callowschoolboy, Cfайлde, Chazzzzbrown, ClarkMobarry, Classicaelecon, CommonsDelinker, Conversionscript, Crisdeda2000, Cybercobra, DEMcAdams, Daniel5127, Dijxtra, Dysprosia, EAderhold, EddEdmondson, Emvee, Enchanter, Enix150, Francvs, Fredrik, Fresheneesz, GaborLajos, Geometryguy, Giftlite, Goodralph, Graham87, Gregbard, Hehsaan, HansAdler, Happy-melon, Hede2000, Hotfeba, Jamesfisher, JasonQuinn, JitseNiesen, JoanneB, JonAwbrey, JoshuaF, JustinJohnson, Kbdank71, Lab-oratory, Lambiam, LeonardVertigel, Lethe, Limowreck, LingKahJai, Macrakis, Magmalex, Majorbrainy, Mate2code, Mdito, Meisam, Melchoir, MichaelHardy, Mindmatrix, Mintguy, Mjaked, MrOllie, NoNamesavailable, Oberiko, OnThisContinent, OrenO, Oxymoron83, PaulAugust, Poccil, Policron, PoorYorick, R'n'B, Richard001, Richie, RuiMalheiro, Rzelnik, SantiagoSaintJames, Scwarebang, Slacka123, SocratesJedi, Stevertigo, TakuyaMurata, TastyPoutine, TobyBartels, TomMorris, TrevorGoodyear, Trovatore, VKKokielov, Vanisheduser34958, Voodoo, Vujke, WarrenPlatts, Wikimol, Wolfrock, Yekrats, АлександраВв, 128anonymous edits

Logical disjunction *Source:* <https://en.wikipedia.org/w/index.php?oldid=560445825> *Contributors:* 2andrewknayev, AceticAcid, AlanUS, Althepal, Anyeverybody, Arachnocapitalist, AxelBoldt, B4hand, BD2412, Bact, Bbukh, BiT, Blinken, Bluemoose, BryanDerksen, Cxong, CBM, CarrieVS, Charlantino, CharlesMerriam, Classicaelecon, ColinMarquardt, CommonsDelinker, Conversionscript, Ctppc, Cybercobra, D.Daemonst, David65536, DesertSteve, Dijxtra, DocDaneeka, Dysprosia, EdC, Emvee, Enix150, Espetkov, Francvs, Fresheneesz, Fuebar, GaiusCornelius, Gamewizard71, Giftlite, Gregbard, Gringo300, Gwib, HansAdler, Hesperian, Ixfd64, Jnestorius, JonAwbrey, JulianMendez, JustinJohnson, Katieh5584, Kbdank71, Kowey, Kurykh, Kzollman, Laymanal, Lemminor, Lethe, Limowreck, LingKahJai, Macrakis, Magioladitis, Mandraxor, Mate2code, Matthewbeckler, Maxdamantus, Mdito, Meisam, Melchoir, MichaelHardy, Mindmatrix, Mintguy, Moulder, NickNumber, ObradorovicGoran, OlegAlexandrov, OnThisContinent, OrangeDog, PamD, Patrick, PaulAugust, Pengkeu, PhnomPencil, Pit, Poccil, Policron, PoorYorick, Recentchanges, RekishiEJ, Rumping, RuyPugliesi, SantiagoSaintJames, Scwarebang, Slacka123, SocratesJedi, Soler97, StephenCCarlson,

Supuhstar, TakuyaMurata, Tarquin, Thryduulf, Tobias Bergemann, Toby Bartels, Tom Morris, Tony Winter, Tony1, Tripsone, Trovatore, Vanished user 34958, Voodoo, Wernhervonbraun, World.suman, Xiao Li, ZeroOne, 96 anonymous edits

Logical equality Source: <https://en.wikipedia.org/w/index.php?oldid=544187712> Contributors: 2ndjpeg, Aeaza, Arthur Rubin, AzaToth, BD2412, Canderson7, Daverocks, Dijxtra, Faus, Francvs, Gamewizard71, Gregbard, Hans Adler, Infovarius, Ixfd64, Jerome Charles Potts, Jbeard, Jon Awbrey, Julian Mendez, Kuzmaka, Lethe, Letranova, Melchoir, Mindmatrix, Oleg Alexandrov, Patrick, Paul August, Peruvianillama, R'n'B, Radagast83, Rumping, Santiago Saint James, Sitush, Toby Bartels, Trinitresque, Trovatore, 30 anonymous edits

Logical implication Source: <https://en.wikipedia.org/w/index.php?oldid=522470966> Contributors: Adam.a.a.golding, Algebraist, Alynnna Kasmira, Ancheta Wis, Aplex, Arthur Rubin, BD2412, BDD, Borgx, CBM, Cnilep, DMacks, Dbfz, Dionyziz, Eric Kvaalen, Giftlite, Gimmetrow, Good Olfactory, Graymornings, Gregbard, Grumpyyoungman01, Hanlon1755, Hans Adler, Igoldste, Incnis Mrsi, Inquisitus, Iranway, Jamelan, Javalenok, Kbdank71, Kintetsubuffalo, Koavf, KyleP, Luna Santin, Macaddct1984, Machine Elf 1735, Magioladitis, Manii1, Minister Alkabaz, Panyd, Philogo, Slakr, Sps00789, Tijfo098, Tomas e, Tziemer991, Velho, Wbm1058, Wemlands, گیلری, 28 anonymous edits

Logical negation Source: <https://en.wikipedia.org/w/index.php?oldid=55596003> Contributors: AHRtbA==, Adam78, Adambiswanger1, Andkore, Andres, Anonymous Dissident, Arvindn, AxelBoldt, BD2412, Benc, Bluemoose, BrydoF1989, Byelf2007, Catgut, Cesarschirmer, Chalst, Chameleon, ChrisGualtieri, Christoffel K, Cookman, Cs-Reaperman, Cybercobra, DTOx, Daf, David Shay, Decltype, Deryck Chan, Dhollm, Djihed, DocWatson42, Edward, EmilJ, Enix150, Eric Qel-Droma, Ejw, FilipeS, Forderud, Francvs, Furby100, Gail, Gaius Cornelius, Giftlite, Gparker, Gregbard, Gwib, Hadal, Hadi Payami, Hans Bauer, HeirloomGardener, Holothurion, HumphreyW, Iain.dalton, Ihcoyc, Ivan Štambuk, Jarble, John Vandenberg, Jon Awbrey, Jonathan Grynspan, Kavadi carrier, Kbdank71, KnowledgeOfSelf, Knucmo2, Lambiam, Loadmaster, Lowellian, Mayoife, Mditto, Meisam, Mifter, Mindmatrix, Mr.Z-man, Musiphil, Nasnema, Nayuki, Neilc, Nortexoid, Obradovic Goran, Oleg Alexandrov, Omegatron, Oursipan, PGSONIC, Pasixxx, Patrick, Paul August, Pazouzou, Pearle, Poccil, Policron, Quatloo, Qwertymith, R'n'B, RJC, Rajah, Revengeful Lobster, Reywas92, Rrburke, Ryguasu, Santiago Saint James, Semmelweiss, Shadex9999, Simeon, Slacka123, Soler97, Spoon!, Stevertigo, Strcat, TAKASUGI Shinji, TheTito, Thumperward, Tobias Bergemann, Toby Bartels, Troels.jensen, Trovatore, UU, Vanished user 1029384756, Victor Yus, Visor, Wdchk, Whatamldoing, William Avery, Xihr, Xnn, Youandme, Zhentmdfan, Zron, Zundark, 116 anonymous edits

Lupanov representation Source: <https://en.wikipedia.org/w/index.php?oldid=335122422> Contributors: Alvin Seville, Maalosh, Michael Hardy, Oleg Alexandrov, RobinK, Welsh

Majority function Source: <https://en.wikipedia.org/w/index.php?oldid=57215801> Contributors: 0, 198.103.96.xxx, 202.141.151.xxx, ABCD, Alexei Kopylov, Balabiot, Ckape, David Eppstein, Gregbard, Ilyaraz, Jesse V., Jzcool, Lambiam, Magioladitis, Michael Hardy, Pascal.Tesson, Radagast83, TFCforever, Tobias Hoevekamp, Vanish2, Андрей Кулников, 4 anonymous edits

Material conditional Source: <https://en.wikipedia.org/w/index.php?oldid=574372718> Contributors: 2601:D:6380:49:216:CBFF:FEB8:9F3B, Alastair Haines, Amcbride, Anonymous Dissident, AnotherPseudonym, Applemister, Arno Matthias, Arthur Rubin, AugPi, Avraham, BD2412, Bearnfæder, Beefyt, Beyond My Ken, BiT, Byelf2007, CarrieVS, Charles Matthews, Cholling, Chzz, Clark Mobby, Classicalcon, Closedmouth, Cnilep, Cybercobra, Dcljr, Dcootoo, Doradus, Dreftymac, Eassin, Eggriffin, Elwikipedista, Eric Kvaalen, Fetk, FilipeS, Francvs, Fresheneesz, Fyrael, Giftlite, Gregbard, Greyfriars, Grumpyyoungman01, Hanlon1755, Hans Adler, Hgetnet, Hibou57, Iamtheudeus, Incnis Mrsi, Indomitavis, JRSpriggs, Jason Quinn, Jaymay, Jiri 1984, Jochen Burghardt, Joel D. Reid, John of Reading, Jon Awbrey, Josang, Julian Mendez, KSchutte, Kbdank71, Leif Czerny, Lukefreeman, Machine Elf 1735, Marc van Leeuwen, Martin von Gagern, Mate2code, Meisam, Melab-1, Mhss, Morriswa, Najoj, NawlinWiki, Nayuki, NevilleDNZ, Newbyguesses, NickDragonRyder, Noobnubcakes, Nortexoid, Oceanofperceptions, Olaf, Png, Pokipsy76, Pyrospirit, Radagast3, Rathkirani, Robma, Rory O'Kane, Ruy thompson, SFinside, Salgueiro, Santiago Saint James, ServiceableVillain, Sholto Maud, Soler97, Sonia, SyntaxPC, Tbsdy lives, TedPavlic, The Tetrast, Tistammer, Trovatore, Vesal, Vonkje, William Avery, Xerula, 84 anonymous edits

Material equivalence Source: <https://en.wikipedia.org/w/index.php?oldid=522632647> Contributors: ABCD, Abolen, Abyal, Acdx, Adam Conover, Adam78, Adjam, AgentPeppermint, Andres, Anonymous Dissident, Ark, Arthur Rubin, AstroHurricane001, BANZ111, BMF81, Brianjd, Bryan.burgers, CBM, CRGreathouse, Camembert, CarrieVS, Cathfolant, Chalst, Chas zzz brown, Chinasaur, Chmarkine, Chricho, Ciphers, Conversion script, Damian Yerrick, Danielpi, Dante Alighieri, DavidCary, Davkal, Delirium, Diberri, Dicklyon, Dominus, DopefishJustin, Dysprosia, Edward Z. Yang, Eggriffin, Ekevu, El Pollo Diablo, Elwikipedista, Enochlau, Eu.stefan, Evercat, Fvidlificator, Ex13, Eyu100, Forderud, Francvs, Geoffrey, Giftlite, Glenn L, Gloin, GoOdCoNEnT, Gregbard, Helder.wiki, Henrygb, Holyknight33, Iamtheudeus, Ichata, Igor Yalovecky, IgorekSF, Incnis Mrsi, Interiot, InverseHypercube, Itai, Jacquerie27, Jarble, Jason Quinn, Jewbacca, Jkelly, Jonatan Swift, Joriki, Joshwa, Karl Dickman, Kenneth M Burke, Kingturle, KjellG, Kmddmk, Larry Sanger, Ledge, Letranova, LiDaobing, Lillingen, Lotje, Master11218, Matthew Woodcraft, McKay, Melchoir, Mellum, Mets501, Michael Hardy, Mikespedia, Mindmatrix, Minehava, Mouse is back, Msh210, MsKnathan, Mvc, Najoj, Nejatarn, Netrap, Neutrality, Nzzi, Oleg Alexandrov, Patrick, Paul August, Pearle, Peterwh, Pforrest, PhotoBox, Picaroon, Pop on a Rope, Postglow, Psychonaut, Quintus314, R.e.b., Rainwarrior, Rapsar, Rbonvall, Revolver, Rhiatal, Ronhjones, Ropers, Ruakh, Ruud Koot, Ryan Reich, SGBailey, Schneau, Serpent's Choice, Shirifan, Shoeofdeath, Singularity, Skippyodo, Smaug123, Stevenj, Stevertigo, Stillnotel, Sugarfoot1001, Sunborn, Superfrank, Surfeited, Suruuna, Taak, Tarquin, TheMadBaron, Thumperward, Tijfo098, Timlevin, Tiny plastic Grey Knight, Tobias Bergemann, Toby Bartels, TrippingTroubadour, Ttzz, Urdutex, Urhixidur, UserGoogol, Vanished user 47736712, Velho, Vicki Rosenzweig, Voltagedrop, Warman06, WestwoodMatt, Widr, Wikiborg, Wikimichael22, WissensDürster, Wmahan, Woohookitty, Wshun, Wwwolf, Yuide, Zundark, Ævar Arnfjörð Bjarmason, Лев Дубовой, 165 anonymous edits

Material nonimplication Source: <https://en.wikipedia.org/w/index.php?oldid=541641446> Contributors: Alex836, BANZ111, BD2412, Bjankuloski06en, Chris Capoccia, Classicalecon, Cybercobra, David Eppstein, Francvs, Gregbard, Jesse V., Kaldari, Kbdank71, Mate2code, Maurice Carbonaro, Meisam, 4 anonymous edits

Modal operator Source: <https://en.wikipedia.org/w/index.php?oldid=544274687> Contributors: Altenmann, BD2412, BenetD, CBM, EmbraceParadox, Erhasalz, Filemon, Gregbard, JRSpriggs, Jim1138, Markhurd, Mild Bill Hiccup, PhnomPencil, R'n'B, Sydactive, Tanfil13, TheOldJacobite, Velho, 2 anonymous edits

Negation Source: <https://en.wikipedia.org/w/index.php?oldid=562405660> Contributors: AHRtbA==, Adam78, Adambiswanger1, Andkore, Andres, Anonymous Dissident, Arvindn, AxelBoldt, BD2412, Benc, Bluemoose, BrydoF1989, Byelf2007, Catgut, Cesarschirmer, Chalst, Chameleon, ChrisGualtieri, Christoffel K, Cookman, Cs-Reaperman, Cybercobra, DTOx, Daf, David Shay, Decltype, Deryck Chan, Dhollm, Djihed, DocWatson42, Edward, EmilJ, Enix150, Eric Qel-Droma, Ejw, FilipeS, Forderud, Francvs, Furby100, Gail, Gaius Cornelius, Giftlite, Gparker, Gregbard, Gwib, Hadal, Hadi Payami, Hans Bauer, HeirloomGardener, Holothurion, HumphreyW, Iain.dalton, Ihcoyc, Ivan Štambuk, Jarble, John Vandenberg, Jon Awbrey, Jonathan Grynspan, Kavadi carrier, Kbdank71, KnowledgeOfSelf, Knucmo2, Lambiam, Loadmaster, Lowellian, Mayoife, Mditto, Meisam, Mifter, Mindmatrix, Mr.Z-man, Musiphil, Nasnema, Nayuki, Neilc, Nortexoid, Obradovic Goran, Oleg Alexandrov, Omegatron, Oursipan, PGSONIC, Pasixxx, Patrick, Paul August, Pazouzou, Pearle, Poccil, Policron, Quatloo, Qwertymith, R'n'B, RJC, Rajah, Revengeful Lobster, Reywas92, Rrburke, Ryguasu, Santiago Saint James, Semmelweiss, Shadex9999, Simeon, Slacka123, Soler97, Spoon!, Stevertigo, Strcat, TAKASUGI Shinji, TheTito, Thumperward, Tobias Bergemann, Toby Bartels, Troels.jensen, Trovatore, UU, Vanished user 1029384756, Victor Yus, Visor, Wdchk, Whatamldoing, William Avery, Xihr, Xnn, Youandme, Zhentmdfan, Zron, Zundark, 116 anonymous edits

Parity function Source: <https://en.wikipedia.org/w/index.php?oldid=543418934> Contributors: Amilevy, Giftlite, M gol, Michael Hardy, Ott2, R'n'B, The enemies of god, Twri, Ylloh, 1 anonymous edits

Peirce arrow Source: <https://en.wikipedia.org/w/index.php?oldid=490791320> Contributors: A-)Brunuš, Adrianwn, ArnoldReinhold, BD2412, Bjankuloski06en, Bookandcoffee, Branzillo, Bryan Derksen, CBM, Cameronc, Chris the speller, Colin Marquardt, Cícero, Dega180, Dijxtra, Dogah, Dreamyshade, Edward, EmilJ, Emvee, Francvs, Furykef, Gamewizard71, GeorgBauhaus, Gregbard, Grendelkhan, Gubbubu, Hans Adler, Hermeneus, Hut 8.5, Igor Yalovecky, Igotta Lemma, Iolar, IronMadenRocks, Iservo, Ixfd64, JMRyan, Jallan, Jean-Frédéric, Jerzy, Jon Awbrey, Kbdank71, Krymzon, Lantern Leatherhead, Loadmaster, Mate2code, Maxim Razin, Meisam, Melchoir, MementoVivere, Metadjinn, Mhss, Mindmatrix, MrOllie, Nbarth, Nesarose, Nortexoid, Ohnoitsjamie, Pangur Ban My Cat, Paul August, Policron, Prestonmag, Pukkie, Queen of the Dishpan, Qwertys, RokerHRO, Santiago Saint James, Sietse Snel, SirPeebles, SocratesJedi, Stevertigo, Stimp, The Tetrast, Tijfo098, Trovatore, Tyomitch, UU, Urhixidur, Vujke, WarrenPlatts, Webber Jocky, 59 anonymous edits

Sheffer stroke Source: <https://en.wikipedia.org/w/index.php?oldid=553081853> Contributors: Achlaug, Ademkader, Algebraist, Archelon, Asenine, AugPi, AxelBoldt, BD2412, Bookandcoffee, Brianln, Brouhaha, Bunnyhop11, CBM, Cameronc, Chalst, Chris the speller, Conversion script, CultureDrone, Cybercobra, Dante Cardoso Pinto de Almeida, David spector, Dcoetze, Dega180, Dhanyaavaada, Dijxtra, Dogah, Dominic, Drakferion, Dspark76, Dysprosia, EmilJ, Emvee, Erud, Ezrakilty, Francvs, Fubar Obfuscous, Gamewizard71, George Leung, Giftlite, Gregbard, Gubbubu, Halo, Hans Adler, Hpypv, Imagist, InductiveLoad, JMRyan, Jamelan, JoanneB, Johnleemk, Jon Awbrey, Jones11235813, Josepholhen, Jouster, Julian Mendez, Kbdank71, KeepOpera, Ksyrie, Lambiam, Leibniz, LittleWink, Loadmaster, Luethi, M&M987, Magioladitis, Manusharma, Mate2code, Meisam, Melchoir, Mhss, Michael Hardy, Mindmatrix, Nad, Nilfanion, NormalAsylum, Nortexoid, Olmsfan, Paul August, Paul Murray, Philogo, Pie4all88, Plastikspord, ProlineServer, Qwertys, Ratiocinate, Redfarmer, Rich Farmbrough, Rohanmittal, Rorro, Rotiro, Royas, Ruud Koot, SOFOOBah, Saaska, Sam, Santiago Saint James, Sceptre, Seba5618, Set theorist, SocratesJedi, Somejan, SpK, Steven Luo, The Tetrast, TheJames, Thumperward, Trovatore, UU, Urhixidur, Vik-Thor, Vujke, Wayward, Woohookitty, Yahya Abdal-Aziz, Yoderj, Zahnradzacken, Zigger, 75 anonymous edits

Sole sufficient operator Source: <https://en.wikipedia.org/w/index.php?oldid=290417881> Contributors: Abazgin, Ancheta Wis, CBM, CBright, Cnoguera, Dixtosa, Domster, EmilJ, FMasic, Francvs, Gregbard, Guppyfinsoup, Hans Adler, Infvw1, InverseHypercube, Jamesfisher, Joshua Issac, Kaldari, Kbdank71, Krauss, LOL, MarSch, Michael Hardy, Nortexoid, Paul Murray, Paxsimius, Qwertys, R.e.s., RichF, Saralee Arrowood Viognier, Sergey Marchenko, Slrubenstein, Swpb, Tijfo098, 41 anonymous edits

Statement (logic) Source: <https://en.wikipedia.org/w/index.php?oldid=576582117> Contributors: 2001:470:1F0B:448:224:8CCF:FED4:BB70, 51kwad, Akerans, Al Lemos, BD2412, C xong, Delfeye, Discospinster, Francvs, Fresheneesz, GB fan, Giftlite, Gregbard, Hans Adler, Hribter, Jason Quinn, Khazar2, Neo-Jay, Philogo, Physis, R'n'B, Rick Norwood, Saehry, Shaun, Tgeairn, Woohookitty, Yamaha5, Zach Lipsitz, 40 ,םנ'נ' anonymous edits

Strict conditional *Source:* <https://en.wikipedia.org/w/index.php?oldid=553015932> *Contributors:* Alansohn, Aquafleur, Arthur Rubin, Beefyt, Cdeough, Chalst, Charles Matthews, Citizensunshine, D.Lazard, Gemtpm, Graymornings, Gregbard, Guanaco, Hanlon1755, Hobbes Goodyear, Incnis Mrsi, J heisenberg, KSchutte, Kbdank71, Machine Elf 1735, Melbournian, Mhss, Michael Hardy, Mu, Ocanter, Philogo, Tide rolls, Tizio, 63 anonymous edits

Symmetric Boolean function *Source:* <https://en.wikipedia.org/w/index.php?oldid=572250071> *Contributors:* HamburgerRadio, Mark viking, Michael Hardy, Twri, 1 anonymous edits

Symmetric difference *Source:* <https://en.wikipedia.org/w/index.php?oldid=567841590> *Contributors:* 4C, Ahoerstemeier, Anonymous Dissident, AugPi, AxelBoldt, Bart v M, Booyabazooka, Calcyman, Charles Matthews, Dcoetzee, Detahedron, Dysprosia, Ed.C, Evan Aad, Fuzzy, Giftlite, Hans Adler, Hyacinth, Hyad, Ideyal, Isnow, J Elliot, Jacksonwalters, Jfresen, Jon Awbrey, Juan Marquez, Karl Dickman, Knhkmb, LJosil, Madmath789, Mate2code, Melchoir, Mets501, Mmernex, Moloch981, Octahedron80, Oleg Alexandrov, OwenBlacker, Patrick, Paul August, PaulTanenbaum, Porges, QuarkyPi, Running, Salix alba, Simeirical, Slaniel, Sopholatre, Splash, Tarquin, The enemies of god, Tobias Bergemann, Trovatore, WhisperToMe, Wile E. Heresiarch, Wshun, 17 anonymous edits

Tautology (logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=576296945> *Contributors:* Allefant, Amanster, Anthony Appleyard, Apokrif, ArsenalTechKB, AshLin, Auntof6, Ayda D, Benjaminevans82, Bjankuloski06en, Booyabazooka, Bsmntbomddo, CBM, Chaser, Cicero, Dbtfz, Diego, Doradus, Drake Wilson, ENeville, Eastlaw, Eumedemito, Eupedia, Francvs, Fratrep, Fredrik, Furykef, George Richard Leeming, Gerhardvalentin, Giftlite, Gregbard, Hairy Dude, Hanlon1755, Hans Adler, Hitanshu D, Homonihilis, Indianandrew, Jamelan, Jemmy Button, Jmlullo, Jon Awbrey, Julian Mendez, Kagnu, Kbdank71, Kejia, Kuszi, Langing, Laocoön11, Laurascudder, LennyCZ, Lkruijsw, Madder, Magioladitis, Markhurd, Maurice Carbonaro, Meisam, Mf140, Michael Hardy, Miss Madeline, Mohsenkazempur, Neelix, Neurosjourn, Nick Number, Nortexoid, Omphaloscope, Pchov, Philogo, PizzaMargherita, Policron, Rifleman 82, Rivertorch, Rnb, Robertbowerman, Robina Fox, RoddyYoung, Royalasa, Seliopou, Simeon, SimonATL, Sligocki, Squids and Chips, Strait, Tanstafl28, Tbhotch, Tpbradbury, Trovatore, Victor Blacus, Vramasub, WikHead, Wvbailey, XDaniex, Zntrip, Æzen, 107 anonymous edits

Zhegalkin polynomial *Source:* <https://en.wikipedia.org/w/index.php?oldid=565894041> *Contributors:* Bkkbrad, CRGreathouse, Dougher, GBL, Gregbard, Hans Adler, Jeepday, Macrakis, Michael Hardy, Rjwilmsi, Vaughan Pratt, 1 anonymous edits

Algebraic normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=564927175> *Contributors:* CBM, Charles Matthews, CyborgTosser, GBL, Hans Adler, JackSchmidt, Jiri 1984, Jon Awbrey, Linas, Macrakis, Mairi, Michael Hardy, Ner102, Olathe, Oleg Alexandrov, Salgueiro, 4 anonymous edits

Boolean conjunctive query *Source:* <https://en.wikipedia.org/w/index.php?oldid=300523273> *Contributors:* Cdldata, Gregbard, Tizio, 4 anonymous edits

Canonical form (Boolean algebra) *Source:* <https://en.wikipedia.org/w/index.php?oldid=574759635> *Contributors:* Bgwhite, Bkkbrad, Bluefire 000, Bookandcoffee, Charles Matthews, ChrisGualtieri, Cybercobra, Discospinter, Douglaslyon, Dysprosia, Eric Le Bigot, Freevol, Fresheneesz, Giftlite, Gil Dawson, Gmoose1, Gregie156, Gurch, Hans Adler, Henriknordmark, Hughbs, Iamnitin, Ixfd64, Joeythehobo, Jon Awbrey, Jwh335, Linket, MER-C, MK8, Macrakis, Mak098765, Mess, Mhayeshk, Mhss, Michael Hardy, Modify, MoraSique, Myasuda, Odwl, Pasc06, Phosphorix, Reach Out to the Truth, SDS, Sanders muc, Sarbruis, Tijfo098, Trovatore, Wavelength, WereSpielChequers, Wrelwser43, Wvbailey, ZeroOne, Zvn, 62 anonymous edits

Conjunctive normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=578004064> *Contributors:* A3 nm, Andkore, Aquatopia, Arvindn, Ary29, Aydos, B4hand, BenBaker, Blaisorblade, Bobogoobo, Bryan Derksen, CBM, Cherlin, Cognominally, CyborgTosser, Danlev, Dardasavta, Dresdnhope, ESkog, Erik Garrison, Fresheneesz, Frostyandy2k, Giftlite, Graham87, Gregbard, Gubbubu, Hairy Dude, Hermel, Hobsonlane, IM Serious, IsleLaMotte, Jacobolus, Jamelan, Jamesx12345, Jleedev, Jochen Burghardt, Jon Awbrey, Jpvimall, Jrtayloriv, Ldo, Linas, Macrakis, Masnevets, MementoVivere, Mhss, Michael Hardy, Mike Segal, Mikhail Dvorkin, Mikolasj, MX2323, Myasuda, Niteowhels, OSDevLabs, Obradovic Goran, Oleg Alexandrov, Policron, Poromenos, PrimeHunter, Robert Merkel, Ross Fraser, Rotemliess, Simeon, Snikeris, Tesi1700, Thesilverbail, Tijfo098, Tizio, Toby Bartels, Tvdm, Wzwz, ZeroOne, 78 anonymous edits

Disjunctive normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=572167406> *Contributors:* A3r0, Ajm81, Altenmann, B4hand, Batena, Ben Spinozaan, BenBaker, Brona, Bryan Derksen, CBM, CyborgTosser, DavidCary, Doulous Christos, EmiJ, Fresheneesz, Graham87, Gregbard, Groovenstein, Gryllida, Haham hanuka, Hans Adler, Igor Yalovecky, Intervalllic, 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

Formal system *Source:* <https://en.wikipedia.org/w/index.php?oldid=569784086> *Contributors:* 16@r, Acattell, Al Lemos, Aleksd, Am Fiosagear, Ancheta Wis, Anthony.h.burton, Architectchao, Arthur Rubin, Ascänder, AugPi, Benc, Bjankuloski06en, BrainMagMo, Bsod2, Buenasdiaz, Byell2007, CBM, Charles Matthews, DesolateReality, Dmceq, Dysprosia, Felix Wiemann, Giftlite, Gregbard, Hans Adler, Hmans, JMK, JRBugembe, John of Reading, Jon Awbrey, Jonathanzung, Jonkerz, Jpbown, Kas-nik, Lambiam, Libcub, Maurice Carbonaro, Mdd, Mets501, Michael Hardy, Modster, Mpaganu, Nikiry, Obradovic Goran, Olaf, Paylett, Pcap, Philogo, Pmanderson, Popopp, Pro8, Qwertys, R'n'B, R. S. Shaw, Rekleev, Rex the first, Rmashadi, Ruud Koot, Saehry, Salt Yeung, Sleepinj, The Wiki ghost, Tijfo098, Tillmo, Timrollpickering, TomorrowsDream, Udirock, Undsowetter, Vanished user kijsdion3i4jf, Waldir, WestwoodMatt, 46 anonymous edits

Blake canonical form *Source:* <https://en.wikipedia.org/w/index.php?oldid=575862344> *Contributors:* David Eppstein, Kmzayeeem, Macrakis

Canonical normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=578004107> *Contributors:* Bgwhite, Bkkbrad, Bluefire 000, Bookandcoffee, Charles Matthews, ChrisGualtieri, Cybercobra, Discospinter, Douglaslyon, Dysprosia, Eric Le Bigot, Freevol, Fresheneesz, Giftlite, Gil Dawson, Gmoose1, Gregie156, Gurch, Hans Adler, Henriknordmark, Hughbs, Iamnitin, Ixfd64, Joeythehobo, Jon Awbrey, Jwh335, Linket, MER-C, MK8, Macrakis, Mak098765, Mess, Mhayeshk, Mhss, Michael Hardy, Modify, MoraSique, Myasuda, Odwl, Pasc06, Phosphorix, Reach Out to the Truth, SDS, Sanders muc, Sarbruis, Tijfo098, Trovatore, Wavelength, WereSpielChequers, Wrelwser43, Wvbailey, ZeroOne, Zvn, 62 anonymous edits

Herbrand normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=237163517> *Contributors:* AshtonBenson, Bjones, Linas, Mere Interlocutor, MorganGreen, 3 anonymous edits

Herbrandization *Source:* <https://en.wikipedia.org/w/index.php?oldid=439798612> *Contributors:* AshtonBenson, Bjones, Linas, Mere Interlocutor, MorganGreen, 3 anonymous edits

Horn clause *Source:* <https://en.wikipedia.org/w/index.php?oldid=576840077> *Contributors:* A Softer Answer, Altenmann, Angela, BD2412, BLNarayanan, CRGreathouse, Cek, Chalst, Charles Matthews, ChrisGualtieri, Compulogger, Dcoetzee, Doradus, Edward, Elwikipedista, EmiJ, Fieldmethods, Ft1, Gareth Griffith-Jones, Gregbard, Gubbubu, Hannes Eder, Hans Adler, Inquam, Jackee1234, Jacobolus, Jamelan, Jarble, Jochen Burghardt, Karada, Karlheg, Kevin.cohen, Kvnerma, Kw1, Ldo, Linas, Logperson, Luqui, MIRROR, MattGiua, Mhss, Michael Hardy, Michael Zeising, NYKevin, Nunoplopes, Paullaks, RatnimSnave, Rintdusts, Ritchy, Rsimmonds01, Tajko, Tawker, Template namespace initialisation script, Thadk, Theone256, Tijfo098, Tizio, Tom harrison, Woohooikit, Xmlizer, 34 anonymous edits

Negation normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=568822174> *Contributors:* Amikake3, Brian Geppert, CBM, CRGreathouse, Gregbard, Kbdank71, Kejia, Linas, Mets501, Mogism, Obradovic Goran, Olathe, Oleg Alexandrov, Pearle, Silverfish, Starblue, Vkuncak, 7 anonymous edits

Prenex normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=559091054> *Contributors:* AugPi, CBM, CRGreathouse, Charles Matthews, Coreyocomnor, Dysprosia, Epiglottis, Esoth, Gandalf61, Greenrd, IsleLaMotte, Jakob.scholbach, Jayme, Joriki, Linas, Lockeownzj00, MattGiua, Mhss, Michael Hardy, Omnipaedista, Pfortuny, Reetep, The Anome, Toobaz, 26 anonymous edits

Skolem normal form *Source:* <https://en.wikipedia.org/w/index.php?oldid=569962792> *Contributors:* 4C, Aleph4, Ashley Y, Charles Matthews, Chris the speller, David Eppstein, Dysprosia, Fresheneesz, Gandalf61, Gubbubu, Hans Adler, Heyitspeter, ILikeThings, IsleLaMotte, Jason Quinn, Jayme, Jochen Burghardt, John of Reading, Jrtayloriv, Jsnx, KYLEMONGER, Kal71, Kayobee, Klangenfurt, Linas, Mark viking, Maxme, Mhss, Nortexoid, Oleg Alexandrov, Pleasantville, Policron, Thesilverbail, Tijfo098, Tizio, Tompsci, Winterstein, Xyzzy n, 32 anonymous edits

Absorption law *Source:* <https://en.wikipedia.org/w/index.php?oldid=563012630> *Contributors:* 16@r, Awis, Bob.v.R, Bookandcoffee, CRGreathouse, Chalst, Charles Matthews, Constructive editor, Cyp, David Eppstein, David Newton, Dcoetzee, Denisarona, Dysprosia, Gregbard, Hans Adler, JackSchmidt, Jamelan, Kbdank71, Lethe, Macrakis, Mhss, Octahedron80, PamD, Policron, Pouipy, RDBury, Schneelocke, Skarebo, Szsquirrel, Tobias Bergemann, Trovatore, 14 anonymous edits

Boole's expansion theorem *Source:* <https://en.wikipedia.org/w/index.php?oldid=570321198> *Contributors:* AManWithNoPlan, AndrewHowse, BabbaQ, Bernatis123, Bwgames, Charles Matthews, DAGwyn, Denisarona, Dwija Prasad De, Freevol, Giftlite, Hamaryns, Harrigan, LOTRrules, Loz777, Macrakis, McCart42, Michael Hardy, Muammar Gaddafi, Omnipaedista, Qwertys, SamB, SebastianHelm, Trovatore, 21 anonymous edits

Boolean prime ideal theorem *Source:* <https://en.wikipedia.org/w/index.php?oldid=571187973> *Contributors:* Aleph4, AugPi, Avsmal, AxelBoldt, Bomazi, CBM, CRGreathouse, Charles Matthews, Chinju, David Eppstein, Dfeuer, Dysprosia, Eric119, Geometry guy, Giftlite, Gonzalcg, Headbomb, Hennobrandsma, Hugo Herbelin, Jon Awbrey, Kope, MarkSweep, Markus Krötzsch, Mhss, Michael Hardy, Ott2, PhnomPencil, RobHar, RoodyAlien, TexD, Tkuvho, Tobias Bergemann, Trioculite, Trovatore, Vivacissamamente, 20 anonymous edits

Compactness theorem *Source:* <https://en.wikipedia.org/w/index.php?oldid=540352099> *Contributors:* Aleph4, Algebraist, AxelBoldt, Baccyak4H, BeteNoir, C56C, CBM, Chas zzz brown, ChrisGualtieri, Christian.kissig, Crashshopper, Delirium, Dominus, Dysprosia, El C, EmiJ, Enoksrd, Gandalf61, Geh, Giftlite, Gregbard, Hans Adler, Hermel, IsleLaMotte, Jason22, Jewbacca, Jossi, Julian Mendez, Lethe, Linas, Luca Antonelli, Lupin, Mhss, Michael Hardy, MichalKotowski, Msh210, Nbarth, Nortexoid, Numbo3, Obradovic Goran, Passingtramp, Paul August, RDBury, Rotem Dan, SDC, Salgueiro, Schneelocke, Tesquivello, Tkuvho, Turms, Turtleboy0, Zero sharp, Zundark, 38 anonymous edits

Consensus theorem *Source:* <https://en.wikipedia.org/w/index.php?oldid=574888207> *Contributors:* AugPi, Firetrap9254, Gregbard, Kenathte, Kruckenberg1, Macrakis, Magioladitis, Merlin444, Pcap, Rich Farmbrough, Sabar, Success dreamer89, Trovatore, 5 anonymous edits

De Morgan's laws *Source:* <https://en.wikipedia.org/w/index.php?oldid=572738486> *Contributors:* 16@r, Action ben, Adambiswanger1, Alexius08, Alphax, Art LaPella, AugPi, B1atv, Benjigil, Bkkbrad, Bluemathman, Bongomatic, Boongie, Boredzo, Byner, Capricorn42, Cdiggins, Chalst, Charles Matthews, Chewings72, Choster, ChromaNebula, Cldoyle, Coolv, Cori.schlegel, Cybercobra, DVdm, DanielZM, DannyAsher, Darktemplar, David Shay, Dcoetze, DesertSteve, Dorfl, Drae, Drake Redcrest, Dysprosia, ESkog, Ebertek, EmilJ, Epr123, Eric-Western, Fratrep, Fredrik, Futurebird, General Jazza, Giftlite, Gobonobo, Graham87, Gregbard, Guppyfinsoup, Hadal, Hairy Dude, Hannes Eder, Hans Adler, Helgus, 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 Sloane, MikeLynch, Mindmatrix, Misercou, Mitch Feaster, Mudlock, Najoj, Nitku, Nutster, Oleg Alexandrov, Petrejo, Policer, R'n'B, RBarry Young, RDBarry, Rapsar, Rodriguez, Rrnr, Saric, Scrutchiefield, SirPeebles, Smimram, Smoseson, Smylers, Squelle, Starblue, Stdazi, Stpasha, Subtractive, Sylvier11, TWiStErRob, TakuuyaMurata, Tarquin, The Anome, The wub, Tide rolls, Ttemnebkrum, Ttvo, Waleed.598, Wavelength, Widr, Xiaodai, 166 anonymous edits

Duality (order theory) *Source:* <https://en.wikipedia.org/w/index.php?oldid=546059869> *Contributors:* Charles Matthews, David Eppstein, Fredrik, Giftlite, Markus Krötzsch, Michael Hardy, PaulTanenbaum, Rschwieb, Silly rabbit, Sterrett, Tobias Bergemann, Trovatore, Vadmium, Zundark, 3 anonymous edits

Laws of classical logic *Source:* <https://en.wikipedia.org/w/index.php?oldid=421259881> *Contributors:* Abecedarius, AndrewA, Arthur Rubin, Autof6, BaseballDetective, Brad7777, CBM, CMBJ, CRGreathouse, Ched, Cheerios69, Classicaelecon, DLazard, Daniel1998, David Eppstein, Delusion23, Dianna, ESkog, EmilJ, Francvs, Gamewizard71, Gary King, Ghaly, Gregbard, Hairy Dude, Hans Adler, Hugo Herbelin, Incnis Mrsi, ItsZippy, Ivannoriel, Izno, Jiri 1984, Jmjeremy, Jmencisom, JohnBlackburne, Jrtayloriv, Khazar, LZ6387, Lambiam, LannaEditArticles, LuluQ, Magioladitis, Matthew Kastor, MetaNest, Michael Hardy, Michiel Helvensteijn, Mindmatrix, Minusia, Muammar Gaddafi, Neelix, Northamerica1000, Oleg Alexandrov, Oxonienses, Paul August, Pgallert, Pome, Proxyma, Rlwood, Robert Thyder, Rotson, Shevek1981, Sofia karampataki, Soler97, StuRat, Telfordbuck, The Rahul Jain, Thorwald, Tijfo098, TonyBrooke, Trovatore, Vaughan Pratt, Waldhorn, Wavelength, William Avery, Wolfmanx122, Wvbaily, 102 anonymous edits

Peirce's law *Source:* <https://en.wikipedia.org/w/index.php?oldid=556940128> *Contributors:* 16@r, Aratus Soli, Ariel Black, Ashley Y, Banno, Branzillo, Chalst, Charles Matthews, Deep Atlantis, Doctor Dillamond, Doradus, Four Dog Night, FunnyMan3595, Furby100, Gamewizard71, Giftlite, GreatWhiteNortherner, Gregbard, Hut 8.5, Iserdo, JRSPriggs, Janburse, Jesper Carlstrom, Jon Awbrey, Julian Mendez, La Mejor Ratonera, Lanter Leatherhead, Leibniz, Metz501, Mhss, Michael Hardy, Mkoconnor, MrOllie, Nortexoid, Obsidian-fox, Ossimanners, Pabix, Pangur Ban My Cat, Pinethicket, Queen of the Dishpan, Root72, Savourneen Deelish, Squids and Chips, Tassedethe, The Tetrast, Tkuvho, Webber Jocky, 29 anonymous edits

Poretsky's law of forms *Source:* <https://en.wikipedia.org/w/index.php?oldid=571622801> *Contributors:* Bearcat, Macrakis, Malcolma

Stone's representation theorem for Boolean algebras *Source:* <https://en.wikipedia.org/w/index.php?oldid=567782900> *Contributors:* Aleph0, AugPi, Beroal, BeteNoir, Blotwell, CBM, Chalst, Chinju, David Eppstein, Falcor84, Fropuff, Giftlite, JanCK, Kuratowski's Ghost, Linas, Markus Krötzsch, Mhss, Michael Hardy, Naddy, Nosuchforever, Pjacobi, Porton, R'n'B, R.e.b., Rschwieb, Sharpcomputing, Smack, StevenJohnston, Tkuvho, Tobias Bergemann, Trovatore, Tsirel, Vivacissamente, Zundark, 17 anonymous edits

Boole's syllogistic *Source:* <https://en.wikipedia.org/w/index.php?oldid=541308032> *Contributors:* Atiwok, Brad7777, Chalst, Gregbard, Mate2code, Mhss, Michael Hardy, Mudlock, Sam Hocevar, StuRat, Trovatore, 5 anonymous edits

Entitative graph *Source:* <https://en.wikipedia.org/w/index.php?oldid=544261210> *Contributors:* David Eppstein, Gamewizard71, Jon Awbrey, JonDePlume, Mhss, Michael Hardy, N4nojohn, R'n'B, Rjwilmsi, The Tetrast, Zundark, 10 anonymous edits

Existential graph *Source:* <https://en.wikipedia.org/w/index.php?oldid=572275088> *Contributors:* 2602:306:C540:2A40:F4C4:1825:B35F:9B, Ael 2, Ancheta Wis, AndrewHowse, Argableg1IV, AvgPi, Avaya1, CBM, Charles Matthews, Djacov, Eleuther, Elwikpedista, Gecko, Giftlite, IanManka, Jon Awbrey, Jujutacular, Julian Mendez, Kl4m, Ktr101, Mdd, Metz501, Mhss, Michael Hardy, Moorlock, My Cat inn, N4nojohn, Oleg Alexandrov, Palnot, Poccil, R'n'B, Sanfranman59, Sdorrance, Silverfish, Synergy, The Tetrast, Tizio, Toby Bartels, TraxPlayer, Zundark, 81 anonymous edits

Implicant *Source:* <https://en.wikipedia.org/w/index.php?oldid=567129756> *Contributors:* Ceklock, Charles Matthews, Chendy, Fcdesign, Fresheneesz, Genuineleather, HopeSeekr of xMule, Jmabel, Jon Awbrey, Macrakis, Mailer diablo, Materialscientist, McCart42, Mhss, Michael Hardy, MrOllie, Nviladkar, Odwl, Pako, Portisere, Pwoestyn, Ra2007, Squids and Chips, Sri go, Svdb, 28 anonymous edits

Laws of Form *Source:* <https://en.wikipedia.org/w/index.php?oldid=576358921> *Contributors:* Abracadab, Adavidb, AndrewHowse, Arthena, AustinKnight, Autarch, Blainster, Bluemoose, CALR, CBM, CXCV, Crummer, Charles Matthews, Chris the speller, Chris83, Concerned cynic, Creidieki, Cyferx, Danielgschwartz, David Eppstein, Dutton Peabody, EagleFan, Ebear422, Erneststrom, FayssalF, Gaius Cornelius, Gerold Broser, Giftlite, Grafen, Gregbard, Hairy Dude, Hans Adler, IanManka, Inc, J04n, JaGa, John Vandenberg, Jon Awbrey, Jpvinal, Kai-Hendrik, Lavintzin, Leolaursen, Lupin, M a s, Metz501, Michael Hardy, Mike Dillon, N4nojohn, NULL, Nehrams2020, Newbyguesses, Nick Number, Omnipaedista, Ospix, Palnot, PamD, Paul Foxworthy, Pdtturney, Philip ea, PhnomPencil, Qaz, R'n'B, RANesbit, Reyk, Rich Farmbrough, Rjwilmsi, Robofish, Rschwieb, Salix alba, Sam, Sapphic, Scdevine, SchreiberBike, Sigfpe, Station1, Supergee, Suruena, The Nut, The Tetrast, Tijfo098, Timwi, Tompsc, Trovatore, Waldir, Yworo, Zundark, 380 anonymous edits

Logical graph *Source:* <https://en.wikipedia.org/w/index.php?oldid=544257079> *Contributors:* A Thousand Dancing Hamsters, Antonielly, Branzillo, Brigit Zilwaukee, Buchanan's Navy Sec, C.Fred, C.W. Murry, CBM, CRGreathouse, CardinalDan, Chester County Dude, Closedmouth, Coffee, Dave Chaparral, David Eppstein, Dbtfz, Delaware Valley Girl, DoubleBlue, E946, El C, Flower Mound Belle, FlyHigh, Gamewizard71, Giftlite, Goethean, Gogo Dodo, Hans Adler, Hut 8.5, Ichiboku Natabori, Igotta Lemma, Islaammaged126, Jeffrey O. Gustafson, Jon Awbrey, JzG, Lantern Leatherhead, Linas, Marsboat, Metz501, Michael Hardy, Mostlyharmless, MrOllie, Mrs. Lovett's Meat Puppets, Navy Pierre, OS2Warp, On This Continent, Overstay, Pangur Ban My Cat, Paul August, Pmanderson, Poke Salat Annie, Queen of the Dishpan, Seb26, Slakr, Southeast Penna Poppa, TFCforever, The Tetrast, Throw it in the Fire, Trebor, Trovatore, Twri, Unco Guid, Viva La Information Revolution!, Wolf of the Steppes, Xuanji, Yolanda Zilwaukee, 9 anonymous edits

Boolean domain *Source:* <https://en.wikipedia.org/w/index.php?oldid=550845409> *Contributors:* AndrewHowse, Asparagus, Autocratic Uzbek, Bibliomaniac15, Boute, Brigit Zilwaukee, Buchanan's Navy Sec, C.Fred, CBM, CRGreathouse, Chester County Dude, Cje, Cliff, Closedmouth, Coredesat, Círcero, Delaware Valley Girl, DoubleBlue, Doubtentry, El C, Flower Mound Belle, Francvs, Gogo Dodo, Hans Adler, Hans Dunkelberg, Hut 8.5, Icharus Ixion, Incnis Mrsi, Jamelan, Jeffrey O. Gustafson, Jon Awbrey, JzG, KJS77, Marsboat, Matthi3010, Mhss, Mrs. Lovett's Meat Puppets, Navy Pierre, Nbarth, Overstay, Pascal.Tesson, Poke Salat Annie, Salix alba, Slakr, Southeast Penna Poppa, Theonlydavewilliams, Toby Bartels, Versageek, Viva La Information Revolution!, Wolf of the Steppes, Xuanji, Yolanda Zilwaukee, 5 anonymous edits

Boolean ring *Source:* <https://en.wikipedia.org/w/index.php?oldid=577325710> *Contributors:* Albmont, Amazingbrock, Anita5192, AshtonBenson, AugPi, AxelBoldt, Charles Matthews, Cokaban, Dcoetze, Dysprosia, Fredrik, Giftlite, Hans Adler, Hwasungmars, JackSchmidt, Jakito, Jeepday, Jochen Burghardt, Mate2code, Mhss, Michael Hardy, Michael Sloane, NSLE, Nbarth, Oleg Alexandrov, R.e.b., Rschwieb, Salix alba, TakuuyaMurata, Trovatore, Valley2city, Vanished user 1029384756, 21 anonymous edits

Goodman–Nguyen–van Fraassen algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=551305589> *Contributors:* Brad7777, CharlotteWebb, Good Olfactory, Knorlin, Michael Hardy, Psinu, Trovatore, 1 anonymous edits

Interior algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=550249120> *Contributors:* Aspects, Bejnari, Charles Matthews, Giftlite, Gogo Dodo, Gregbard, Hans Adler, Hyacinth, Kuratowski's Ghost, Linas, Metz501, Mhss, Michael Hardy, Omnipaedista, R'n'B, Trovatore, Zundark, 126 anonymous edits

Lindenbaum–Tarski algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=568994024> *Contributors:* Bloodshedder, CBM, Charles Matthews, Charvest, ChrisGualtieri, Classicaelecon, Gauge, Gelingvistoj, Hugo Herbelin, Jochen Burghardt, Jon Awbrey, Mets501, Mhss, Nortexoid, Oerjan, Piotrus, Sneakfast, Tijfo098, Tobias Bergemann, Trovatore, 4 anonymous edits

Relation algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=560903974> *Contributors:* AshtonBenson, AugPi, Balder ten Cate, Brad7777, CBM, Charles Matthews, Charvest, Concerned cynic, D6, David Eppstein, Elwikpedista, Giraffedata, Gregbard, Hans Adler, Irmy, JohnBlackburne, Jon Awbrey, King Bee, Koavf, Lambiam, Leruit, Lethe, LilHelpa, Linelor, Mboverload, Metz501, Mhss, Michael Hardy, Nastor, Nbarth, Ott2, Paul Carpenter, Physis, Plasticup, QuadrivialMind, R'n'B, Ramsey2006, Sam Staton, Samppi111, SchreyP, Sjcoosten, The Tetrast, Tillmo, Tobias Bergemann, Vaughan Pratt, Woohookitty, Zundark, 129 anonymous edits

Residuated Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=544881178> *Contributors:* Charvest, Gracefool, Mhss, PWilkinson, Tobias Bergemann, Vaughan Pratt, 2 anonymous edits

Robbins algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=567970863> *Contributors:* Afteread, Giftlite, Irbisgreif, Jdvelasc, Peap, Qwertyus, Spiros Bousbouras, Thehotelambush, Tobias Bergemann, Trovatore, Zaslav, 21 anonymous edits

Sigma-algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=575283331> *Contributors:* IfForTheMoney, 202.141.81.xxx, 24.200.219.xxx, A koyee314, Alfredo J. Herrera Lago, Amirmath, AndrewKepert, Archelon, ArnoldReinhold, AxelBoldt, Barnaby dawson, BearMachine, Brad7777, BrideOfKripkestein, CBM, CRGreathouse, Cerberus0, Charles Matthews,

Charvest, Chinju, Cmapm, Conversion script, Crasshopper, Crawfoal, Dan Hoey, David Cooke, Dbtfz, Digby Tantrum, Dinno, Dysprosia, Elwikipedista, EmilJ, Fibonacci, Forgetfulfunctor, Gala.martin, Gauss, Giftlite, Godvjr, Graham87, Gubbubu, Henning Makholm, Hippasus, Iwnbp, JanusDC, Jayme, Jheald, Jim.belk, Jmath666, Joeabauer, Jrtayloriv, Jung dalgish, KHamsun, Karada, Keith111, Lambiam, Lethe, Li3939108, Limit-theorem, Linas, Lucinos, Madmath789, Mark viking, MathKnight, Max139, Mboverload, Mct mht, Melchoir, Melcombe, Mets501, Michael Hardy, MicoFilos, Miguel, MisterSheik, Moyacercerchi, Msh210, Nbarth, Nielses, NumSPDE, Object01, Ocsenave, Oleg Alexandrov, Paartha, Passw0rd, Paul August, Polcrion, QuarkyPi, Quentat, Rafi5749, RayAYang, RogierBrussee, Romanm, Ruakh, Salix alba, SgtThroat, Solstag, Soumyakundu, Stevan White, StevenJohnston, Stj6, Stott, Stpasha, Tarquin, Teamps42, Thegamer 298, Trovatore, Tsirel, Ultramarine, Vivacissamamente, Vrable, William Elliot, Xantharius, Y256, Zaslav, Zundark,刻意, 93 anonymous edits

Topological Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=346088468> *Contributors:* CBM, Jon Awbrey, Oleg Alexandrov, The Great Redirector, Trovatore, 4 anonymous edits

Two-element Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=569491236> *Contributors:* Assiliza, Avaya1, CBM, Classicaelecon, GTBacchus, Gernot.salzer, Giftlite, Gregbard, Hans Adler, Igny, Incnis Mrsi, Jordet, Lambiam, Linas, MCAllen91, Mhss, Michael Hardy, Nakon, Nbarth, Ngabendi, Nick Number, Nurg, Oleg Alexandrov, Palnot, Pjpvj, Plugwash, R'n'B, Salix alba, Tagremover, Tijfo098, Trovatore, WimdeValk, Zundark, 49 anonymous edits

Complete Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=544005066> *Contributors:* Angelobear, AshtonBenson, CBM, Charles Matthews, Closedmouth, Giftlite, Hans Adler, Headbomb, Jemiller226, Melchoir, Mets501, Mhss, Michael Hardy, Noobeditor, Qm2008q, R.e.b., Scythe33, Shanel, Silverfish, Tim Ayeles, Trovatore, Vaughan Pratt, VictorPorton, Zero sharp, 9 anonymous edits

Derivative algebra (abstract algebra) *Source:* <https://en.wikipedia.org/w/index.php?oldid=543854064> *Contributors:* Brad7777, CBM, Davyzhu, EmilJ, Giftlite, Mets501, Mhss, Trovatore, Unara, 11 anonymous edits

First-order logic *Source:* <https://en.wikipedia.org/w/index.php?oldid=577022563> *Contributors:* .digamma, 34jjkkj, A3 nm, Ahmed saeed, Ahouseholder, Alastair Haines, Almit39, AnakngAraw, Apolkhanov, Arthur Rubin, AshtonBenson, AugPi, Avakar, AxelBoldt, Axl, Banazir, Be hajian, BenBaker, Bender235, Bgwhite, Blaisorblade, Bookandcoffee, Borgx, Brick Thrower, ByElf2007, CBM, CMB2, CRGreathouse, Cacadril, Caesura, Camn86, Carbo1200, Cdills, Chalst, Charles Matthews, Chidoftv, Chinju, Classicaelecon, Cloudyed, ConcernedScientist, Conversion script, Creidieki, Cronholm144, Crowne, Dan Gluck, Danman3459, David Eppstein, David.Monnaux, Dbtz, Detozee, DesolateReality, Dhruvbaldawa, Dhruvee, Diannaa, Djk3, Dkf11, Dpol, Dwheeler, Dysprosia, ELDRAS, Edward, Ejars, Eksplio, Eleuther, Elwikipedista, EmilJ, English Subtitle, Eubulide, Exostor, Expensivehat, Filemon, Fl, Foxjwill, Francvs, Frecklefoot, Fredrik, Gabefair, Gf uip, Gherson2, Giftlite, Greenrd, Gregbard, Grim23, Gubbubu, Hakeem.gadi, Hans Adler, Harburg, Henning Makholm, Heyitspeter, Hilverd, Histre, Hobsonlane, Holyseven007, Hq3473, Ht686rg90, Iannigb, Igodard, Inquam, InverseHypercube, Iridescent, Ifxfd64, JECompton, JRSpriggs, Jan1ad, Jaseemabid, Jay Gatsby, Jayme, Jirislaby, Jleedev, Jod, Jon Awbrey, Jorend, Joriki, Jsnx, Juansempere, Julian Mendez, JulianMendez, Karl-Henner, Katieh5584, Kelly Martin, Kendrick Hang, Kgoarany, Kim Bruning, Kku, Klausness, Kumiko (renamed), Kusunose, Kwertii, LBouehoun, Lambiam, Lauri.pirttaho, Lethe, Lifeformmho, Linas, Ljf255, Loadmaster, Looxix, Lord British, Lucidish, Malleus Fatuorum, MarSch, Mark Renier, Markhurd, Marner, Mate2code, Maurice Carbonaro, Meloman, Mets501, Mhss, Michael Hardy, Michael Stone, Mike Fikes, Mike Segal, Mild Bill Hiccup, Mindfruit, Minesweeper, Mistercupcake, Mmm, Mojo Hand, Mormalge, Mpiesenbr, Msh210, Nahaj, Nanmus, Nanobear, NavarroJ, Netraprt, NicDumZ, Nick Number, NickGarvey, NoBu11, Norman Ramsey, Nortexoid, Nzrs0006, Obradovic Goran, Oleg Alexandrov, Omphaloscope, Onceler, Otto ter Haar, Palnot, Patrick, Paul August, Peap, Pdibner, Penumbra2000, Per Olofsson, Philogo, Phuzion, Physis, Phyte, Pkreeker, Pmetzger, Polcrion, Pomte, R.e.b., RG2, Rjaguar3, Randall Holmes, Randomblue, Reach Out to the Truth, Rich Farmbrough, Richard L. Peterson, Rjs.swarnkar, Rjwilmsi, Rlcolasanti, RobHar, RubyQ, Sameer0s, Sampatlek, Sanpra1989, Saric, Siroxo, SixWingedSerpent, Slaniy, SouthLake, Spyramid, SpigotMap, Spug, Starlord, Stevertigo, Subversive.sound, Tasseddie, TechnoFaye, Templatypedef, Tesseran, The Anome, The Tetrast, Tigranes Damaskinos, Tijfo098, Timwi, TitusCarus, Tizio, Tkuvhlo, ToastiII, Tobias Bergemann, Trovatore, Turms, Urhixidur, Utcursh, Vandens, VanishedUserABC, Velho, VictorAnyakin, Virago250, Voorlandt, Whitepaw, WikHead, Willhig, Wireless99, Xplat, Youandme, Zeno Gantner, Zero sharp, 293 , זנ'נ' anonymous edits

Free Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=544182726> *Contributors:* Arthur Rubin, BD2412, CBM, CSTAR, Chalst, Charles Matthews, Daniel Brown, Gregbard, Jiri 1984, Mate2code, Mhss, Oleg Alexandrov, Output, R'n'B, Trovatore, Zundark, 8 anonymous edits

Heyting algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=559504179> *Contributors:* 925faahsflkh917fas, Accidus, Altenmann, CBM, CBM2, Ceroklis, Charles Matthews, Charvest, David Eppstein, DefLog, Elwikipedista, EmilJ, Gauge, Giftlite, Giorgiomugnaini, Gregbard, Gzhanstong, Husond, Ioqzc, Kompik, Kuratowski's Ghost, Lethe, Linas, Marcosaedro, Markus Krötzsch, Mets501, Mhss, Michael Hardy, Nortexoid, Omnipaedista, Phys, Rjwilmsi, Rogério Brito, STyx, Sam Staton, Sjejoosten, Sopholatre, Steve2011, TheNewLayoutSucks, Tijfo098, Tillmo, Trovatore, UKoch, Urhixidur, Vaughan Pratt, Zundark, 74 anonymous edits

Monadic Boolean algebra *Source:* <https://en.wikipedia.org/w/index.php?oldid=543612705> *Contributors:* Charles Matthews, Gregbard, Hans Adler, Kuratowski's Ghost, Mhss, Michael Hardy, R'n'B, Safek, Tijfo098, Trovatore, 16 anonymous edits

Skew lattice *Source:* <https://en.wikipedia.org/w/index.php?oldid=542845734> *Contributors:* Joaopitacosta, Kevetko, Mhsiggers, R'n'B, Reyk, Rich Farmbrough, Salix alba, Stemonitis, 3 anonymous edits

And-inverter graph *Source:* <https://en.wikipedia.org/w/index.php?oldid=527502788> *Contributors:* Aaron Hurst, Adrianwn, Alan Mishchenko, Andreas Kaufmann, Andy Crews, Appraiser, Ettrig, Gregbard, Igor Markov, Jlang, Jon Awbrey, Jonathan de Boyne Pollard, Ketiltrot, Linas, Michael Hardy, Mikeblas, Nobletripe, Pigorsch, Pkkao2, Rjwilmsi, Trovatore, 5 anonymous edits

Boolean analysis *Source:* <https://en.wikipedia.org/w/index.php?oldid=420195144> *Contributors:* AROMME, David Eppstein, Derek Andrews, Mdd, Melcombe, Michael Hardy, Otolemur crassicaudatus, PaulKeizer, Rjwilmsi, Ronz, Schrepp, Vangelis12

Boolean operations in computer-aided design *Source:* <https://en.wikipedia.org/w/index.php?oldid=539172105> *Contributors:* Eastlaw, Michael Hardy, Nickdi2012, Ritchie333, 1 anonymous edits

Circuit minimization *Source:* <https://en.wikipedia.org/w/index.php?oldid=577863793> *Contributors:* Andreas Kaufmann, ChyranandChloe, David Eppstein, Delasz, Dfass, FUZxxl, Jackmcbar, Michael Hardy, RupertMillard, Steaphan Greene, Tijfo098, Trovatore, Uoft ftw, WillowW, Wtshymanski, 4 anonymous edits

Espresso heuristic logic minimizer *Source:* <https://en.wikipedia.org/w/index.php?oldid=576616860> *Contributors:* Altenmann, Delasz, George A. M., GreanVII, Hermel, LordJeff, LouScheffer, Michael Hardy, NikhilKrgvr, Oleg Alexandrov, Pgr94, SigmaEpsilon, TeamX, WimdeValk, Woortery, Yakushima, 23 anonymous edits

Logic gate *Source:* <https://en.wikipedia.org/w/index.php?oldid=573847656> *Contributors:* 0x8I9J*, 129.26.12.xxx, A0602336, AJim, AMbroodEY, Abhignarigala, Abhinav dw6, Abhishek Jacob, Adam outlier, Adam1213, Ademkader, Aecis, Aeons, Akamad, AJ Lemos, Alai, Alex:D, Alienskull, Ancheta Wis, Anclation, Andris, Andy, Andy M. Wang, Arnavion, Arteile, Arthena, Ashton1983, Athernar, Atifme, Audriusa, AxelBoldt, BD2412, Bantman, Barber32, Bean49, BenBaker, Berserkers, Big angry doggy, Bigdumbdinosaur, Bkell, Bkkbrad, Blackenblue, Blues-harp, Boing! said Zebedee, Bongwarrior, Bonzo, Bookandcoffee, Booty443, Booyabazooka, CSTAR, Can't sleep, clown will eat me, CanisRufus, Capricorn42, Carmichael, Cburnett, Centrx, Chef Super-Hot, Cikicdragan, Cipherswarm, Circuit dreamer, Clarkcj12, Colin Marquardt, Cometystyles, Conversion script, Creidieki, Crystallina, DVdm, Dacium, Dan100, Dancter, DavidCary, Deadworm222, Derek Ross, Dicklyon, Diomidis Spinellis, Dirkbb, Discopinter, Djkrainik, DmitiTrix, Dmitry123456, Dominus, DonkeyKong64, DoubleBlue, Draconicfire, Dspark76, Dysprosia, ESKog, Eassin, Easter Monkey, EddyJ07, Egmontaz, Eibx, Elep2009, Eleuther, Epachamo, Eprb123, Espetkov, Evaluist, Eveningmist, Everyking, Ewlyahoocom, Feis-Kontrol, Firebug, Folajimi, Forward Until Dawn, Frecklefoot, FreddieRic, FreelanceWizard, Fresheneesz, Frymaster, Fibhrygv, GLPeterson, GOD, GOVIND SANGLI, GRAHAMUK, Galloping Moses, Giftlite, Glenn, Glrx, GrayFullbuster, Gregbard, Guy Harris, Hanacy, Hans Adler, HereToHelp, Heron, Hersfold, Hgilbert, Hmrorris94, Hooperbloob, Horselover Frost, Husigeza, I Like Cheeseburgers, Ian Burnet, Igor Markov, Inductiveload, Iranway, Itsmeshiva, Ifxd64, J Di, J.delanoy, JDspeeder1, JHunter1, JNW, JYi, Jackfork, James086, Jamesmcmahon0, Jcbarr, Jeaday, Jim1138, Jbeard, Jk2q3jrkls, Jkominek, Jlang, Jni, Joaquin008, JoeKearney, Jon Awbrey, Jonathan de Boyne Pollard, Jonpro, Jschnur, Kaldari, Karada, Karmosin, Kglavin, Kineox, Knownot, Kuru, La goutte de pluie, Lambiam, Latka, Lear's Fool, Lectonar, Lightspeedchick, Lilbonanza, Lindosland, Logan, Lord Kelvin, Lordhatus, LucasVB, LunaticFringe, Luzian, M 3bdelqader, MER-C, Mac, Machine Elf1735, MagnaMopus, Magnus Manske, Mahjongg, Mamidanna, MarXidad, Marchal Ney, Marek69, Marylee23, Marysunshine, Massimiliano Lincetto, Materialscientist, Maximus Rex, Mblumer, Meistro god, Melab-1, Mello newf, Mets501, Mgiganteus1, Michael Hardy, Mindmatrix, Mirror Vax, Mormegil, MrX, Mrand, Mrt3366, Mudlock, Murugango, Mushroom, Mz bankie, N313ts, N5lh, Nabla, Namazu-tron, Neparis, Netsnipe, Niv.sarig, OLEnglish, Ol Fifth, Omegatrash, Onix, Oxyromor083, Paul Murray, Peruvianillama, Peter Winnberg, Philip Trueman, Piano non troppo, Pinethicket, Pingveno, Pion, Plugwash, RMSfromFiC, RTC, Rajiv Beharie, Rangi42, Rchard2scout, Reddi, Rettetast, Rhdv, Rich Farmbrough, Rick Sidwell, Rickterp, Rilak, Robchurc, Robert Bond, Rohamittar, Roman12345, Roo72, Rsmry, SCBailey, STEDMUND07, Safeskiboydunkno, Salvar, Shadow148, Sjorford, Sketchmoose, Skoch3, Smark33021, Snow Blizzard, SocratesJedi, SorryGuy, SpaceFlight89, Spirit469, Stannered, Stephnb, Stevenglowa, Stevertigo, StuRat, Sunderland06, Swtpc6800, Syndicate, SynergyBlades, THEN WHO WAS PHONE?, Teammm, Teh Naab, Tentinator, The Anome, The Nameless, The Rambling Man, The Red, The Tetrast, The Thing That Should Not Be, Tide rolls, Tijfo098, Towopedia, Trinibones, Tygrr, Twy7, USPatent, Updatebjarni, Urocyon, V8rik, Vadmium, VanFowler, Vdsharma12, Versus22, Vibhijain, Vishnava, Vonkje, Vrenator, WarlordFrederick, Wasell, Wbm1058, Whitepaw, Wikitikitaka, Wilking1979, WimdeValk, Wtshymanski, XPEHOPE3, Yyy, ZanderSchubert, Zen-in, ZeroOne, Zvn, 866 anonymous edits

Augustus De Morgan *Source:* <https://en.wikipedia.org/w/index.php?oldid=575259749> *Contributors:* 216.60.221.xxx, Abductive, Alastairmciver, Almit39, Anomalocaris, AnonMoos, Anthony.moore, Astrochemist, AugPi, Autodidaktos, Bender235, Bill Thayer, Billlion, Blanchardb, BrainyBabe, Burn, Cartman0052007, Chenopodiaceous, Cherlin, Chinju, Choess,

ChrisGualtieri, Conversion script, Crazyeddie, Crosbiesmith, Cybercobra, D. Webb, D6, Da Joe, DanielZM, Deleting Unnecessary Words, Deor, Dimadick, Download, Drostie, Dsp13, Ekaterin-nl, Eliyak, Eloquence, Emperorbma, Ericamick, Fadesga, Fredrik, Gadfium, Gaius Cornelius, Gandalfxiviv, Gene Ward Smith, Giftlite, Gobonobo, Graham87, GreenUniverse, Gregbard, Ground Zero, Helgus, Hemmingsen, Henrygb, Heron, INKbusse, Icairns, Jaredwf, Jasperdoomen, Jeepday, Jinian, John, Johnbibby, Jon Awbrey, Jpbowen, Jumbuck, JustAGal, Kestrel man hoovers in the dark, Ketiltrotur, Kingturtle, Kjaergaard, KoenDelaere, Lang rabbie, Livededge12, Lucidish, Maotsetung5, Materialscientist, Mav, MessinaRagazza, Michael Hardy, Mirer, Mk5384, Monegasque, Mrguyguy226, Nimetapoeq, Nimmacer20, Not-just-yet, Ocanter, Omnipaedista, Ontoraul, Palnot, Paradocotor, Paul August, Peruvianllama, Polylerus, Poor Yorick, Porcher, Pred, Proslfaes, QueenAdelaide, RJM81, RSM, Reminiscenza, RexNL, Rgdboer, Rich Farmbrough, Saturn star, Savantas83, Shlrare, SimonP, Skeptic2, Smb1001, Soroko, Sosayso, Spellmaster, Srin181, Sun Creator, Suslindisambiguator, Tarquin, Terper, The Tetrast, Timrollpicking, TonyW, TorontoFever, Toytoy, Tsemii, Utcurtsch, Vanish2, Vaughan Pratt, Waterwheel, Wholotone, Wilt, XJaM, Xiaopao, Aethelwold, 124 anonymous edits

Charles Sanders Peirce Source: <https://en.wikipedia.org/w/index.php?oldid=574753413> Contributors: (aeropagitica), 172, 216.60.221.xxx, 2602:306:CD15:1A40:6968:5439:8C21:B695, A New Nation, Aaron charles, Adam04, AdinaBob, Alan Liefting, Albert Wilford Monroe, Alcmaeonid, All Men Are, Allecher, Alt:pasta, Ancheta Wis, And Dedicated To, And Seven, Andres, AnnMBake, Antandrus, Anthrophilos, AntisocialSubversive, Archelon, ArglebargleIV, Ariel Black, ArkinAardvark, Arx Fortis, Askari Mark, Atfyfe, AtomikWeasel, Avraham, AxelBoldt, B9 hummingbird hovering, BD2412, Badlytwice, Balcer, Banuo, Bbatsbe, Beetster, Ben BenBaker, Bender235, Beyond My Ken, Billy Hathorn, BirgitteSB, Bkell, Bkonrad, Blainster, Bob Burkhardt, BranJ, Brent williams, Brockorgan, Brought Forth, Bunnyhop11, Butseriouslyfolks, Chalst, Charles Matthews, CharlesMartel, Chase me ladies, I'm the Cavalry, Chekaz, Cherlin, Chira, Christian Roess, Christofurio, Chromancer, Cnilep, Cobalbluetony, Colonies Chris, Conceived In Liberty, Conversion script, Coppertwig, Created Equal, Cristian.albanese, Cummings01, Curps, Cyberstrike2000x, CyrusC, D6, DASonnenfeld, DCDuring, Damrvrhunter, Dancesince, Dani0rad, Danny lost, Darrell Wheeler, Davehi1, David Eppstein, David spector, David91, Dearborn Wacker, Deflective, Delirium, DeserfSteve, Deville, Dhart, Discospinster, Diverting Internet Gossip Dress Up Game, Dmr2, Doc Glasgow, Docu, Don4of4, Dreadstar, Ebe123, Ed Nauseum, Edvard Munchkin, Edward E. Nigma, Eemil Kankaanpää, Eldredo, ElinorD, Enchanter, Epistemologist, Eric.dane, Eurleif, Ezrakility, Fang Aili, Farmer Kiss, Favonian, Four Score And Seven Years Ago, Francys, FranksValli, FunnyMan3595, G7a, Gaius Cornelius, GcsWRhlc, Gecko, Geminata, Genesiwinter, Geremia, Gerry Ashton, GhostofSuperslum, Giftlite, Gilliam, Gimmetrow, GirasoleDE, Go for it!, Goethean, Goofé Dean, Graysnots, Gregbard, Gruebleen, Haemo, Hans Adler, Headbomb, Heath, Hede2000, Heron, Hockey Knight In Canada, Idiothek, Igiffin, J. Van Meter, JEN9841, JLJ, JLATondre, JYOuyang, Jac16888, Jackhynes, Jagness, Jay Gatsby, Jayig, Jbdavez, Jboy, Jean Howbree, Jean Santeuil, Jeandré du Toit, Jeff G., Jinx The Phinque, Jivecat, Jizzbug, Jishapiro, Jlwelsh, Jmcne, Joe House, John Deas, Johnpacklambert, Jon Awbrey, Jonny WÅDD, JorgeGG, Jossi, Josteinn, Jpgordon, JustZeFaxMaam, JzG, K, KHamsun, KYPark, Kaldari, Kato9Tales, Kazvorpal, Khb3rd, Kelly Martin, Kevin B12, Kiefer, Wolfowitz, Kirshank, KI833x9, Knight Of The Woeful Countenance, Knucmo2, Koavf, Kohser 3.0, Ktr101, Kurykh, Kwamikagami, Kzollman, Larstebil, Laurapr, Lawandeconomics1, Leibniz, Leon..., Leutha, Lichtconlon, Lightmouse, Lisatwo, Llywrch, LogicMan, Love And Fellowship, Luk, Ludos3, Luna Santin, M4701, MECU, MZMcBride, Magioladitis, Magister Mathematicae, Marc Girod, Master son, Materialscientist, Maunus, Max Plodwonka, Mdd, Meco, MegaSloth, MengTheMagnificent, Mhss, Mhyim, Michael Hardy, Michael Rogers, Mike Rosoft, Mike hayes, Mild Bill Hiccup, Moe Epsilon, Monegasque, Moreschi, Moulbrey, MrOllie, Name Slightly Anonymized, NawlinWiki, Neilc, Nigholith, Nikolaos Bakalis, Nimetapoeq, Ninly, Nimmacer20, Nk, Odd nature, Olav Smith, Olivier, Omnipaedista, On This Continent, Only, Ori.llvneh, Ortolan88, Our Fathers, Palnot, Pasixxx, Pastordavid, Paul A, Peruvianllama, Peter Graif, Phuzion, Plasticup, Plastikspork, Plindenbaum, Pmanderson, Queen of the Dishpan, R'n'B, RA0808, Ragesosa, Razimantv, Rdsmith4, Recognition, Reedy, Reflex Reaction, Renamed user 4, RexNL, Reywas92, Rich Farmbrough, Ripberger, Rjwilmsi, Robson correia de camargo, Rodhullandem, Ruud Koot, Salix alba, SamCardioNgo, Samboner, Samsara, Santiago Saint James, Sardanaphalus, Sorrance, Search4Lancer, Semi Virgil, Sethmahaney, Shadowjams, Sheez Louise, Sir Humphrey Appleby, Slim Margin, SlimVirgin, Slrubsten, Sly of he Green, Snowolf, So1, SomeStranger, Spaghettisburg Address, SqueakBox, Srich32977, Stan Shebs, SteinbDJ, Stephen Gilbert, Stevenzenith, SummerWithMorons, Sun Creator, Sureupon, Swampyank, Tabletop, Tail, Tarquin, Tbhotch, Tdaim, Tequila Mockingbird, Terrek, Tevildo, The Alchemist, The Anome, The Man in Question, The Proposition That, The Tetrast, The Transhumanist, TheGrappler, Thekohser, Thomasmeeks, Thumperbar, Tjmayerins, Tomisti, Tony1, Treutrs, Trikaduliana, Tsk351, Typewritten, Urhixidur, Uri, VI-LIII, Vanished User 0001, Velho, Venus Verdigris, Voice Of Xperience, Vojvodaen, WAS 4.250, WOSLinker, Wahrmund, Wareh, Way of Inquiry, West Brom 4ever, Who's Wife, Widr, Wiki alf, WikiPedant, Would Goods, Wyli Ali, Xact, Yamara, Ypetrachenko, Yuval madar, Zbxgscqf, Zelda Zilwaukee, Ziemia Cieszynska, Zoicon5, Ødipus sic, 543 anonymous edits

George Boole Source: <https://en.wikipedia.org/w/index.php?oldid=576488492> Contributors: 19cass20, 2008CM, 444x1, 4twenty42o, A.amitkumar, ABCD, ABF, Abcpathros, Acroterion, Adam Cuerten, Addihockey10, Addshore, Adw2000, Ahoerstemeier, Alai, Alan Liefting, Alexius08, Almit39, Altenmann, AltiusBimm, Ancheta Wis, Andrew Gray, Anglicanus, Animuum, Anna Lincoln, Antandrus, Any820, AppaBalwant, Arakunem, ArcherOne, ArielGold, Arthena, At-par, Atlanta, Australopithecus2, BD2412, BRG, BSTemple, Babelfisch, Banes, Bayle Shanks, Bejnar, Blainster, Blankslate8, Bob Burkhardt, Bobo192, Boldstep, Bollar, Bongwarrior, Brianga, Brother Francis, C.Fred, CA387, CWenger, Can't sleep, clown will eat me, Capricorn42, Cgilmer, Chalst, Charles Matthews, Chiswick Chap, Chris857, Chzz, Ckatz, Crazynas, Cwkmail, D6, DARTH SIDIOUS 2, DGG, Da nuke, Db099221, Deb, Deflective, Denisarona, Dimadick, Discospinster, Djegan, Djin, Djimes12, Djordjevic, Doctormatt, Donner60, Doradus, Dr.h.friedman, Dwight666, EDUCASE3E, Edward321, El C. Ellywa, Elwikpedista, Eric Corbett, Favonian, FeanorStar7, Fordmadoxfraud, Fox Wilson, Fredrik, Fuhghettaboutit, Gadfium, GenghisKhanviet, Gerfinch, Giftlite, Gilliam, Goegales4321, Graham87, Grahamec, Gregbard, Grembleben, Grendelkhan, Grumpyyoungman01, Guiolopez, Gurch, Gylix, Gzanstrong, Hairy Duke, HalfShadow, Hephaestos, Heron, Hirzel, Hqb, Hydrogen Iodide, Icarins, IncognitoErgoSum, Inter, Introvert, Inwind, Irchristian, Ironholds, Isidore, J.delanoy, JDspeeder1, Jfreeman, Jackiespeel, Jagged 85, JamAKiska, Jan Blanicky, Jaraalbe, Jaredwf, Jason Quinn, Jassonpet, Jim Douglas, Jmundo, Jon Awbrey, Karen Johnson, Kaszeta, Keegan, Keith D, Ketchup1147, Ketiltrout, Kingpin13, Kkm010, Knight1993, Kolja21, Kukini, Kumiokos (renamed), Kwamikagami, Kyz, LPChang, Lastorset, Lightmouse, Lights, Lilhinx, LincsPaul, Logan, Logicist, Logicus, Lucidish, Luisrock2008, Lumos3, MER-C, Magioladitis, Malles Fatuorum, Mallspeeps, Mandolinface, Martinyl, Mateck, MattGiua, Matthew Fennel, Mav, Meigan Way, Mhyim, Michael Hardy, MickyDripping, Miranda, Miym, Monegasque, Msrasnw, Mwanner, Naaman Brown, Nabokov, Nephenites, Newbyguesse, Nimetapoeq, Nimmacer20, Notjim, Nuker, Obmjintokcus, Ohconfuser, Ojan, Oleg Alexandrov, Omnipaedista, Ontoraul, Orelstrigo, OrgasGirl, PBS-AWB, PMLawrence, Peruvianllama, Peter Karlson, PeterMKehoe, Peter, Philip Trueman, Pmanderson, Poor Yorick, Prashantongarker, Professorial, Proteus, R'n'B, Reddi, Renesis, Rettetast, Rgdboer, RickK, Rjparsons, RogDel, Romanm, Roy da Vinci, Ruszewski, Ryan032, ST47, Sageofwisdom, Samsara, SandStone, Scjessey, Sfan00 IMG, Shubinator, Ske2, Skeptic2, Skizzik, Slaniel, Smalljim, Smallweed, Snow Blizzard, Sodin, Solomon7968, Spellmaster, SpuriousQ, SqueakBox, Squids and Chips, Stan Shebs, StudentmrB, Supertouch, Sylent, TYelliott, Tamfang, Tanthalans39, Tapir Terrific, Tarquin, TedColes, Template namespace initialisation script, The Grumpy Hacker, The Tetrast, The wub, TheGrappler, Thingy, Timmy1, Tinlv7, Tomisti, TonyW, Trovatore, Twtvhat, Umapathy, ValBaz, Vanished user ikijerw34iaeolaserific, WJBscribe, Webclient101, Wereon, Widr, Wiki alf, Woohookitty, Wsvlqc, Wtmitchell, Ww2censor, Xcentaur, Xenophon777, Xezbeth, Yamamoto Ichiro, Yashowardhani, YeIrishJig, Zaheen, Zfr, Zone, 574 anonymous edits

Ivan Ivanovich Zhegalkin Source: <https://en.wikipedia.org/w/index.php?oldid=541054644> Contributors: FeanorStar7, Hans Adler, IceCreamAntisocial, Michael Hardy, Monegasque, Omnipaedista, Oracleofottawa, Vaughan Pratt, Waacstats, Woohookitty, XJaM, 2 anonymous edits

John Venn Source: <https://en.wikipedia.org/w/index.php?oldid=577364265> Contributors: A bit iffy, A3RO, AHMartin, Aa42john, Aaron Brenneman, Alansohn, Alex91119111, Almit39, Amplitude101, Andre Engels, AndyCjp, Antonio Lopez, Astrochemist, Athelstan53, Avs5221, Basilo12, Bongwarrior, CALR, Capricorn42, Carlossuarez46, Cjrother, Clark89, CommonsDelinker, D6, DMacks, Discospinster, Doc glasgow, Dreadstar, Dsp13, Duncan.Hull, Dungodung, Elliskev, Epbr123, Ferhat9, Fetofs, Fourth ventricle, Franhigg, Fredrik, Glane23, INic, Icosahedron, ImgariJ, J.delanoy, JDP90, JLATondre, Jaraalbe, JmCor, Jmndo, Johnbibby, Johnpseudo, Jona Mur, Jpbowen, Jredmond, Kaisershatner, Karlewis, Katieh5584, Keegan, Keith D, Kingbojk, Kingpin13, KnowledgeOfSelf, Lucidish, MC Hammer, MC10, Magioladitis, Martarius, Mdd, Misibaci, MrChupon, MusikAnimal, Mxn, Necrofear, Nick, Nick Number, Oleg Alexandrov, Ollie, Omnipaedista, Op, Deo, PBS, Paddles, Paul Barlow, Paul Clapham, Paulebrown, Philip Trueman, Poor Yorick, Puffin, Radgeek, Reedy, Res2216firestar, Rjwilmsi, Rror, Sagaciousuk, Schutz, Sean123123, Shadowjams, SimonArlott, Solomon7968, Starlemusique, Stefano510, Stereotyper, SuperHamster, Tabletop, Taraborn, The Illusive Man, The wub, Thinggg, Tide rolls, Tow, Troy 07, W.D., Wafulz, Wegeton, Widr, Wikipelli, Wknight94, Xn4, Yimby, Yintan, Zachlipton, Zazou, Zeekec, Zeno Gantner, 'O oþþpoç, 285 anonymous edits

Marshall Harvey Stone Source: <https://en.wikipedia.org/w/index.php?oldid=572669530> Contributors: Agricola44, Akriasas, Algebraist, Almit39, Anne Bauval, Archelon, Bbsrock, Charles Matthews, D6, DFRussia, Dan Gardner, Davehi1, David Eppstein, Divega, Dungodung, Edward Vielmetti, G716, JYOuyang, Johnpacklambert, Jona123, Joolz, Joseph Solis in Australia, JustAGal, Ktr101, Lockley, Lupin, MagneticFlux, Michael David, Michael Hardy, Nbarth, Nsk92, Omnipaedista, PDH, Paul Taylor, Prmacn, Pvmonthide, RDBrown, RayAYang, Rjyanco, RobertWHarper, Sean.hoyland, Sullivan.t.j, Suslindisambiguator, Trovatore, Waacstats, Woodshed, 22 anonymous edits

William Stanley Jevons Source: <https://en.wikipedia.org/w/index.php?oldid=568770510> Contributors: 16@r, 7h3 3L173, A. Carty, AdamSmithe, Ancheta Wis, Angelaa26, Anonymous Dissident, Anthon.Eff, Ave05mwg, Bearcat, Bender235, Bobblewik, Cab4p, Charles Matthews, Colonies Chris, Conti, Crosbiesmith, D6, Darolew, DavidCBryant, DavidLevinson, Dcoetzee, Defective, DelftUser, Denarius, Dino, DI2000, Doneerd, Felix Folio Secundus, Florm, Forbsey, Gian-2, Giftlite, Goochelaar, Gregbard, Gurch, Helvetius, Ineuw, Isnow, JASpencer, JakeRs6666, John Foley, Jon Awbrey, Jondk, Joseph Solis in Australia, Jpbowen, Julian Felsenburgh, Juvepa2002, Jyril, KF, Languagehat, Lawandeconomics1, LawrenceKhoo, Lestrade, Marghanita, Matan2005, Michael Hardy, Mintleaf, Nat11, Nelson, Nerfer, Nimetapoeq, OS2Warp, Omnipaedista, Oracleofottawa, Oxbox, P. S. Burton, Parafaustus, Pcpccp, Plotinus, Plucas58, Profoss, R'n'B, RandomP, Rich Farmbrough, Rinconsolero, Rjwilmsi, Rodhullandem, Roundhouse0, Seba5618, Shoeofdeath, Soobrickay, Srich32977, StAnselm, Thomasmeeks, Timrollpicking, Tomisti, Trident13, Tuesdaily, Viriditas, Volunteer Marek, Waacstats, Wikidea, Wikikrsc, Wlwhyte1, Wprlh, Zootsuits, 75 anonymous edits

Clause (logic) Source: <https://en.wikipedia.org/w/index.php?oldid=543258750> Contributors: Amalas, Animist, Calle, Charivari, Compulogger, Gregbard, Jrtaylor19, Lgavel, Mhss, PieRRoMaN, Rafael Keller Menezes, Simeon, Slakr, Stephan Schulz, Tijfo098, Tizio, Yuriz, 15 anonymous edits

Contradiction Source: <https://en.wikipedia.org/w/index.php?oldid=572673668> Contributors: 16@r, 84user, Abootmoose, Acebrock, Ajd, Aldux, Alexander1257, Algebraist, Ancheta Wis, AndriuZ, AndyCjp, Andyfugard, Andyjsmith, Anna Frodesiak, Atethmekos, BD2412, BanyanTree, Barkeep, Barras, Bongwarrior, CBM, CCS81, Can't sleep, clown will eat me, Carabinieri, CesarB, Chalst, Chooserr, Connormah, Daira Hopwood, Discospinster, Dominus, Dysprosia, EdJohnston, Ekem, Epsilon0, Feministo, Fillupp, Francvs, Gamewizard71, Gary1234, GassyGuy, GeneralManager, Giftlite, Glane23, Graymornings, Gregbard, Hair, HarryHenryGebel, Holen4, Husond, Hut 8.5, Hydrex, Iamtheman217, J.delanoy, JackLumber, Jason Quinn, Jaymczone, Jdevine, Jeffrey Mall, Jim1138, Jitse Niesen, JoeFromrandb, JohnnyB256, Johnuniq, Jokestress, Jpatokal, Jpbowen, Jumbuck, KataLaveno, Kbdank71, Keilana, Killiondunde, Kingpin13, Kku, Lazulilasher, LilHelpa, Litnin200, Lukeskywanker76, Lulu of the Lotus-Eaters, Lycurgus, Lyseh, M3taphysical, Marek69, Markhurd, Martious, Maurice Carbonaro, Mayumashu, Mets501,

Michael Hardy, Mild Bill Hiccup, Mrg3105, NY Amateur, Nathanielfirst, Nburden, NeonMerlin, Netsnipe, NickCT, OkPerson, Oleg Alexandrov, Ootachi, Orange Suede Sofa, Organous, Oursipan, Panchitaville, Pedant17, Perostoj, Peterdjones, Pharaoh of the Wizards, Pinethicket, Piotrus, Poccil, Policron, Porcher, RainbowOfLight, RandomAct, Rich Farmbrough, Rillian, Rnickel, Robo37, Ryguasu, Ryguy1994, SWAdair, Sango123, SchreiberBike, Shadowjams, Siebrand, Siroxo, Smalljim, Snigbrook, Spencerk, Springnuts, Squids and Chips, Srich32977, Sumek, Talmage, Thane, The Founders Intent, Thomasdav, Tide rolls, Trinitrix, Tziemer991, Utcursch, Vanished user psdfiwenf3niurunfuh234ruhwdb7, Velho, Victor2399, Vs3894, Wavelength, Who123, Wikielwikingo, Wvbailey, Xinyu, Xyzzy n, Yintan, Zarex, Zekechills, 205 , ٢٠١٧ anonymous edits

Deductive closure Source: <https://en.wikipedia.org/w/index.php?oldid=547342594> Contributors: Andrewaskew, Bumm13, CBM, Gregbard, Guppyfinsoup, Hans Adler, John Quiggin, KSchutte, Koavf, NeilFraser, Ninly, Pjrm, RDBrown, Tassedet, 3 anonymous edits

Formation rule Source: <https://en.wikipedia.org/w/index.php?oldid=545095728> Contributors: Arthur Rubin, Cpiral, EmilJ, Giftlite, Gregbard, Hans Adler, Kbdank71, Michael Hardy, Tahu88810, The Wiki ghost, Timrollpicking, 1 anonymous edits

Frege system Source: <https://en.wikipedia.org/w/index.php?oldid=489109402> Contributors: Bynne, EmilJ, Empty Buffer, Headbomb, Michael Hardy

Frege's propositional calculus Source: <https://en.wikipedia.org/w/index.php?oldid=542543131> Contributors: AugPi, CBM, Charles Matthews, Cmdrjameson, Gregbard, Gubbubu, Jpbowen, Julian Mendez, Kku, MathMartin, Mets501, Mhss, R'n'B, Rich Farmbrough, The Tetrast, Wclark, 6 anonymous edits

Implicational propositional calculus Source: <https://en.wikipedia.org/w/index.php?oldid=545356518> Contributors: BD2412, Balloonguy, Byelf2007, CBM, EmilJ, Grafen, Graymornings, Gregbard, Hotfeba, Hugo Herbelin, JRSpriggs, Mhss, N4nojohn, R'n'B, RDBury, 3 anonymous edits

Intermediate logic Source: <https://en.wikipedia.org/w/index.php?oldid=543489874> Contributors: Byelf2007, CBM, Charles Matthews, EmilJ, Gregbard, Grue, Hpvp, Hugo Herbelin, Kbdank71, Kenneth M Burke, Leibniz, Lokiclock, Lysdexia, Mets501, Mhss, Oleg Alexandrov, Silverfish, 5 anonymous edits

List of logic systems Source: <https://en.wikipedia.org/w/index.php?oldid=576474628> Contributors: EmilJ, Gregbard, Hugo Herbelin, JRSpriggs, Markhurd, Michael Hardy, Nick Number, Omnipaedita, R'n'B, Radagast3, Sun Creator, Slawomir Bialy, Tkuvho, UniversumExNihilo, 7 anonymous edits

Literal (mathematical logic) Source: <https://en.wikipedia.org/w/index.php?oldid=565613468> Contributors: CBM, Egriffin, Gregbard, Kbdank71, Krassotkin, Lagenar, Mhss, Mikhail Dvorkin, Mikolasj, Obradovic Goran, R'n'B, Simeon, Termininja, Tiago de Jesus Neves, Tijfo098, 7 anonymous edits

Logical consequence Source: <https://en.wikipedia.org/w/index.php?oldid=577523781> Contributors: Adam.a.golding, Algebraist, Alynna Kasmira, Ancheta Wis, Aplex, Arthur Rubin, BD2412, BDD, Borgx, CBM, Cnilep, DMacks, Dbtbfz, Dionyziz, Eric Kvaalen, Giftlite, Gimmelrow, Good Olfactory, Graymornings, Gregbard, Grumpyyoungman01, Hanlon1755, Hans Adler, Igoldste, Incnis Mrsi, Inquisitus, Iranway, Jamelan, Javalenok, Kbdank71, Kintetsubuffalo, Koavf, KyleP, Luna Santin, Macaddct1984, Machine Elf 1735, Magioladitis, Mani1, Minister Alkabaz, Panyd, Philogo, Slakr, Sp00789, Tomas e, Tziemer991, Velho, Wbm1058, Wemlands, ۲۸ anonymous edits

Nicod's axiom Source: <https://en.wikipedia.org/w/index.php?oldid=548205391> Contributors: Empty Buffer, GrafZahl, Gregbard, Laforgue, Michael Hardy, Ser Amantio di Nicolao, UniversumExNihilo, 2 anonymous edits

Open sentence Source: <https://en.wikipedia.org/w/index.php?oldid=541205918> Contributors: Alan Lifting, AndrewHowse, Atoll, AugPi, Bigbluefish, CBM, Clconway, DevastatorIC, ENeville, EmilJ, Etincelles, Gregbard, JaGa, JamesBWatson, Jason Quinn, Jshadias, Kbdank71, Klemen Kocjanec, LOL, Ledy, Linas, Magog the Ogre, Malcolm, Melchoir, Mhaitham.shammai, Mhss, Michael Hardy, Nemti, Oleg Alexandrov, Oliver Pereira, Pizza Puzzle, Randomblue, Rgdboer, Ruakh, Sharky, Toby Bartels, Unisonos, Wimt, 27 anonymous edits

Predicate (mathematical logic) Source: <https://en.wikipedia.org/w/index.php?oldid=573488095> Contributors: Abdull, Action potential, Alex S, Arthena, Barbara Shack, Bunyk, CBM, César (usurped), Ego White Tray, FrozenMan, Gamewizard71, Gregbard, Hans Adler, Jason Quinn, Jblurlinson, Joseph Solis in Australia, Kenyon, Lh389, Linas, Llorenzi, Lradrama, Marknew, Mladifilozof, Mmehdi.g, Odoncaoa, Philogo, R'n'B, Sandik, Scsibus, SimonP, Soroush83, Svick, Tijfo098, Toby Bartels, UncleFluffy, UndisputedLoser, Velho, William Avery, Zundark, 20 anonymous edits

Principle of distributivity Source: <https://en.wikipedia.org/w/index.php?oldid=445575232> Contributors: Btyner, CBM, Chalst, Charles Matthews, Erudecorp, Eskilp, Gamewizard71, Gregbard, Ghanstong, Jb-adder, LiederLover1982, Oleg Alexandrov, Rjwilmsi, Shirahadasha, Tobias Bergemann, 9 anonymous edits

Proof by contrapositive Source: <https://en.wikipedia.org/w/index.php?oldid=563927798> Contributors: Alexander.mitsos, Andyfugard, BD2412, Badgernet, BertSeghers, Brianjd, Byelf2007, Carabinieri, Centrx, Daniel5Ko, DesolateReality, Geoffrey.landis, Gregbard, InverseHypercube, Jackofhats, Jason Quinn, Jimmaths, NickPenguin, PerryTachett, UsaSatsui, حمادي حسني, ۸ anonymous edits

Proposition Source: <https://en.wikipedia.org/w/index.php?oldid=577526593> Contributors: 16@r, 2A01:E35:2426:85E0:1141:24F4:6AA2:A96A, A8UDI, ABF, Ali, AllenFerguson, Amicuspublilius, Andkore, Andrewghutchison, Antandrus, AxelBoldt, Banno, Barkeep, Barticus88, Bobo192, Bobrayner, Bookinvestor, BrideOfKripenstein, Brolin Empey, Byelf2007, CBM, CRGreathouse, Caiaffa, Chalst, Ckatz, Cnilep, Coasterlover1994, Connormah, Conti, Coppertwig, Cybercobra, Dalot, Denys, DetectiveKraken, Dthomsen8, Duesentrieb, Dwnelson, E235, Eastlaw, Eekerz, Ehuss, Envee, Evanreyes, Evercat, Eyesnore, Finell, Fresheneesz, Funandtrvl, Gamewizard71, Giftlite, Ginsengbom, Good Olfactory, Gregbard, Grumpyyoungman01, Gveret Tered, Hairy Dude, Hans Adler, Hans-Jürgen Streicher, Hapsiainen, Harryboyles, Helix84, Honestrosewater, Hugo Herbelin, Ignacioerico, Iridescent, Ish ishwan, Isnow, J.delanoy, Jason Quinn, Jaymay, Joe Wreschnig, John of Reading, Jon Awbrey, Juansempere, K.lee, Katalaveno, King Lopez, Kredal, Kzollman, LAAFan, Lacatasias, Lakitu, Larry V, Legion fi, Leolaursen, Levineps, Luis Felipe Schenone, MER-C, MPrel, Makotyo, Martpol, Mate2code, Mav, Michael Hardy, Minesweeper, Monkeymann, Motomuku, NSH001, Nieske, Ojigiri, Omnipaedita, Oxymoron83, Patl, Philogo, Philthecow, Phiwu, Pinethicket, Pomte, Purnendu Karmakar, RJFJR, RedWolf, Reddi, Rick Norwood, SanketDash, Satellizer, Sdorrance, Sebbe, Sethmahoney, Stevertigo, Stevietheman, Stwalkerster, Superborsuk, THEN WHO WAS PHONE?, TakyuMurata, Tgeairn, The Thing That Should Not Be, TheSuave, Timrollpicking, Tobias Bergemann, Toby Bartels, Tracerbullet11, V2Blast, Velho, Voyaging, WhatamIdoing, Wknight94, Woohooikit, Wortafad, Yalcram, Yesterdog, Zeno Gantner, Zoe, ۱۸۱ anonymous edits

Propositional proof system Source: <https://en.wikipedia.org/w/index.php?oldid=531791071> Contributors: Ben Standeven, CBM, David Eppstein, EmilJ, Gregbard, Jocme, LittleWink, Magioladitis, Michael Hardy, R'n'B, Rdt, 6 anonymous edits

Propositional variable Source: <https://en.wikipedia.org/w/index.php?oldid=541181878> Contributors: Aisaac, AndrewHowse, CBM, Creidieki, Foxjwill, Giftlite, Gregbard, Jon Awbrey, Julian Mendez, Kbdank71, Mets501, Mhss, Mitsukai, Pdabrowiecki, Pomte, Tijfo098, Trovatore, Woohooikit, 4 anonymous edits

Rule of inference Source: <https://en.wikipedia.org/w/index.php?oldid=576571855> Contributors: Algebraist, ArglebargleIV, Arthur Rubin, BAxelrod, BD2412, Brighterorange, Byelf2007, CBM, CRGreathouse, CSTAR, Cleared as filed, Dan Gluck, Delphinebbd, Elwikipedia, Elwood j blues, EmilJ, Epbr123, Eusebius, Fram, Gamewizard71, Giraffedata, Gregbard, Grumpyyoungman01, Hurricane Angel, JHunterJ, Jason Quinn, Jim.belk, Jori, Kbdank71, Khalid hassani, Ldo, Lokiclock, Lucidish, Markus Krötzsch, MeltBanana, Mhss, Michael Hardy, Nahaj, Neilc, Nortexoid, Physis, Poor Yorick, Quadell, Robofish, Rossami, Ruud Koot, Simeon, Tesseract2, Tijfo098, Timrollpicking, Tktkt, Undsweiter, Waldir, WillMall, 30 anonymous edits

Rule of replacement Source: <https://en.wikipedia.org/w/index.php?oldid=568557522> Contributors: Arthur Rubin, Dooooot, ENeville, Gregbard, Jochen Burghardt, Michael Hardy, SwisterTwister, 4 anonymous edits

Second-order propositional logic Source: <https://en.wikipedia.org/w/index.php?oldid=451145773> Contributors: Chalst, Epsilon0, Gregbard, Jason Quinn, Meloman, Michael Hardy

Substitution (logic) Source: <https://en.wikipedia.org/w/index.php?oldid=576633741> Contributors: Bomazi, Chalst, Gregbard, JDspeeder1, Jarble, Jochen Burghardt, Mhiji, Mild Bill Hiccup, Peap, R'n'B, Ruud Koot, Tijfo098, 3 anonymous edits

Syntaxcategorical term Source: <https://en.wikipedia.org/w/index.php?oldid=551644001> Contributors: Ael 2, AugPi, Dale Chock, DragonflySixtyseven, Fram, Grammar conquistador, Gregbard, Here today, gone tomorrow, HistorianofLogic, Hmains, Honestrosewater, John of Reading, Khazar2, Markyb23, Mike Rosoft, Omnipaedita, Paul August, Tim Q. Wells, Wbeek, 7 anonymous edits

System L Source: <https://en.wikipedia.org/w/index.php?oldid=461033328> Contributors: Aiken drum, Hugo Herbelin, Michael Hardy, RichardVeryard, 1 anonymous edits

Unsatisfiable core Source: <https://en.wikipedia.org/w/index.php?oldid=546252918> Contributors: D123488, DBeyer, Edward, EgoWumpus, Gregbard, Jok2000, LouScheffer, Michael Hardy, Salmar, 6 anonymous edits

Zeroth-order logic Source: <https://en.wikipedia.org/w/index.php?oldid=546675901> Contributors: Aleph4, Ars Tottle, Autocratic Uzbek, C.Fred, CBM, CardinalDan, Coredesat, Delaware Valley Girl, DionysiusThrax, Flower Mound Belle, Giftlite, Gogo Dodo, Gregbard, Hans Adler, Humain-commre, Hut 8.5, IP Phreely, Jamesfisher, Jon Awbrey, Julian Mendez, JzG, Kaldari,

Kbdank71, Lambiam, Lethe, Majorly, Mets501, Mhss, Michael Hardy, Michael Slone, Mrs. Lovett's Meat Puppets, Navy Pierre, Nn123645, Paul August, Poke Salat Annie, Redrocket, RxS, Santiago Saint James, Seb26, Slakr, The One I Love, The Proposition That, Trainshift, Trovatore, Unco Guid, Unknown Justin, Unschool, Versageek, Viva La Information Revolution!, West Goshen Guy, 11 anonymous edits

Affirming a disjunct *Source:* <https://en.wikipedia.org/w/index.php?oldid=54118581> *Contributors:* Andeggs, Bookandcoffee, Bryan Derksen, Chi Sigma, Dysmorphodrepanis, Frayr, Graymornings, Gregbard, Grumpyyoungman01, Jon Awbrey, KSchutte, Knucmo2, Machine Elf 1735, Mrwojo, Niggurath, Shinmawa, Silence, Taak, Vzbs34, XDanielx, 13 anonymous edits

Affirming the consequent *Source:* <https://en.wikipedia.org/w/index.php?oldid=577510796> *Contributors:* 271828182, Alba, Andeggs, Aprofel, B F Gray, Bhmunos, Bloomingdedalus, Bryan Derksen, CBM, Cadr, ChrisGualtieri, Conversion script, Corey, Corvi42, Denispis, DevAudio, Dianelos, Factorial, Forest51690, Gregbard, Grumpyyoungman01, Hillary.conway, Humanoid12, Isilanes, JRM, Jamelan, Jfromcanada, John Bentley, KSchutte, Kjoonlee, Klemen Kocjanic, Korruski, Kwadavids, Kyu-san, Ljhenshall, Loadmaster, Logicchecker, Loodog, Ludimer, MC10, Magioladitis, Manwiththemasterplan, Michael Hardy, Mindspillage, Momergil, Mrwojo, N8cantor, Neil Brown, Neilc, Nneonneo, Philogo, Rabbiz, Raistlinjones, Rdsmith4, Renamed user 4, Rich Farmbrough, Rursus, Shawnc, Short Brigade Harvester Boris, Silence, SjolanderM, Snaxalotl, SomeDudeWithAUserName, Sonia, Srd2005, Steel, Sumergocognito, Sundar, Suseno, THobern, Taak, Tarquin, The Nameless, The electron, Theresa knott, Thickslab, Voidvector, Walkiped, Wik, 62 anonymous edits

Denying the antecedent *Source:* <https://en.wikipedia.org/w/index.php?oldid=570611840> *Contributors:* 271828182, Akerans, Allformweek, Andeggs, Angr, Bookandcoffee, Bryan Derksen, CharlesGillingham, DavidHozAu, DavidSTaylor, Elembis, Factorial, Furby100, Gaga, Gemtpm, Gen. Quon, Gongshow, Gregbard, Hiddenhearts, Isilanes, Julian, Jamelan, Jeltz, Jfromcanada, Jpebo, KYPark, Kaveh, Lycte, Mark.camp, Mrwojo, NickelShoe, Obscurans, Pseudomonas, Richard001, Rursus, Sasquatch, Shawnc, Silence, Steel, Supermorf, Taak, Voidvector, Waldir, WhisperToMe, William M. Connolley, Xaquseg, Ybbor, Zocky, 42 anonymous edits

List of rules of inference *Source:* <https://en.wikipedia.org/w/index.php?oldid=577871721> *Contributors:* 2A02-8421:1336:9400:349C:81CD:2417:78BB, Anonymous Dissident, ArnoldReinhold, AugPi, Bkell, CBM, Colonies Chris, El C, Fireice, Gregbard, Infinity ive, JMRyan, JamesMazur22, Jitse Niesen, Jiy, Jonathanzung, Julian Mendez, Kumioko (renamed), LilHelpa, Magioladitis, Marc van Leeuwen, Mark Renier, Mhss, Mogism, Poor Yorick, Rezarob, Shadro, Sidmow, Stwalkerster, Toby Bartels, ZeroOne, 38 anonymous edits

Absorption (logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=577170471> *Contributors:* BukLauBrah, Dcirovic, Dooooot, Gregbard, Michael Hardy, PamD, SwisterTwister, 5 anonymous edits

Admissible rule *Source:* <https://en.wikipedia.org/w/index.php?oldid=569029588> *Contributors:* Brighterorange, CBM, Chalst, EmilJ, Gregbard, Hailey C. Shannon, I dream of horses, Jeffrey Bosboom, Lambiam, Magioladitis, Pacogo7, Rjwilmsi, Rycle731, SchreiberBike, Simeon, Tijofo098, Timo Honkasalo, Woodshed, Zvika, 2 anonymous edits

Associative property *Source:* <https://en.wikipedia.org/w/index.php?oldid=573574650> *Contributors:* 16@r, AdiJapan, Alansohn, Anaxial, Andre Engels, Andres, Auntof6, AxelBoldt, Banus, Bender2k14, Brona, Burn, CBM, Charles Matthews, Charvest, Christian List, Classicalcon, Cliff, Conversion script, Cothrun, Creidieki, DLazard, DAGwyn, Daniel5Ko, Darklich14, David Eppstein, Deadcorpse, Deleting Unnecessary Words, Dysprosia, Egriffin, Ex13, Fox Wilson, FreplySpang, Furby100, Furka ocean, Fuzzform, GaborLajos, Giftlike, GoingBatty, Graham87, Gregbard, H2g2bob, Hello71, Herbee, Ideyal, JCraw, JRSpriggs, Jason Quinn, Jeronimo, Josh Parris, Jshflynn, Kenrambergm2374, Kevin Gorman, KnightRider, Krishnachandranvn, Kuraga, Kwantus, Lethe, Linas, Lugia2453, MacMed, Marek69, Mate2code, Mattblack82, Melchoir, Michael Hardy, Mmernex, Mr Gronk, Mudshark36, Octahedron80, Oleg Alexandrov, PI314r, Patrick, Paul August, Physis, Pinethicket, Pizza Puzzle, Pyrospritz, Quondum, Rang213, Razimantv, Rbanzai, Rich Farmbrough, Robinh, Salgueiro, Salix alba, SchifityThree, SixWingedSeraph, Smig, Smooth O, Some jerk on the Internet, Thane, The Thing That Should Not Be, Thumperward, Tide rolls, Tobias Bergemann, Toby, Toby Bartels, Trefgy, Ujoimro, Wereon, Will Beback, Wine Guy, XJaM, Xiv, Yurik, Zundark, Zven, 123 anonymous edits

Biconditional elimination *Source:* <https://en.wikipedia.org/w/index.php?oldid=576239109> *Contributors:* Angela, CBM, Conversion script, Gamewizard71, Graham87, Gregbard, GregorB, Jim.belk, Justin Johnson, Lambiam, LilHelpa, Mark viking, Patrick, Rich Farmbrough, 4 anonymous edits

Biconditional introduction *Source:* <https://en.wikipedia.org/w/index.php?oldid=563093236> *Contributors:* Arthur Rubin, CBM, Conversion script, Dooooot, Gamewizard71, Graham87, Gregbard, Jim.belk, Jitse Niesen, Justin Johnson, Patrick, Sketchee, 9 anonymous edits

Commutative property *Source:* <https://en.wikipedia.org/w/index.php?oldid=576379276> *Contributors:* 100110100, 12cookk, 131.155.14.xxx, 16@r, 2602:306:370F:D40:2D46:9CBA:9558:BFED, Aaron Rotenberg, Acdx, Aceshooter, Ahoerstemeier, Am Fiosagair, Ameulen11, Andre Engels, Anonymous Dissident, Arnaugir, Arthena, Ashmoo, AxelBoldt, B.d.mills, Bando26, BenFrantzDale, Burn, Charles Matthews, Childzy, ChrisGualtieri, Christian List, Classicalcon, Complexica, Conversion script, Cybercobra, DAGwyn, DBooth, Dan Gluck, Danièle.tampieri, DavidLeighEllis, Davidlu0421, Dbachmann, Deathnomad, Derek.cashman, Dger, Dirac66, Dithridge, Dreftymac, Dskjhgd, Dysprosia, Dzer0, Ebeler, El C, Enochlau, Floridi, Fox2k11, Fr33ke, Francev, Frenchge, Fibhrygv, GTBacchus, Gail, Gcm, Gdr, Geometry guy, Giftlike, Gilderien, Graham87, Grashii, Gregbard, Ground Zero, Hairy Dude, HamburgerRadio, Haseldon, Headbomb, Herbee, IRWolfie-, Ideyal, Isopropyl, Ivan Štambuk, JamesAM, Jeff3000, Jeffq, JII, Joe8824, JoergenB, Josh Parris, Joshelepknpsa, JustUser, Khazar2, Knutux, Kozuch, Larsnostdal, Legion fi, Life, Liberty, Property, Linas, Lupin, MacMed, Malcolm rowe, Marechal Ney, Masnevets, Master of Puppets, Mattpickman, Melchoir, Michael Hardy, Michael Slone, MichaelRWolf, Mikael Brockman, Mike2vil, Mikemill, MisterSheik, Mlessard, Mlm42, Mormegil, Nacho Librarian, Nafis ru, Nev1, None but shining stars, Octahedron80, Oleg Alexandrov, OverlordQ, Patrick, Paul August, Peruvianlama, Petri Krohn, Philip Trueman, Pizza Puzzle, Poliron, Psimmler, Quondum, R000t, RedWolf, Reetep, Rich Farmbrough, Rick Norwood, Rjwilmsi, RobertMcGibbon, Robinh, Romanm, Salix alba, Samadam, Samuel Huang, Sarregouset, Sayiner, SchifityThree, Scientific29, Slashme, Slightsmile, Snoyes, Srlfeller, Starwiz, Stephen Poppitt, Stillnotlef, Szquirrel, Tarquin, The JPS, Thehakimboy, ThinkEnemies, Thomprod, Tobias Bergemann, Toby, Toby Bartels, Tony Sidaway, Touriste, Ubule, Ujoimro, Unmitigated Success, VKokielov, Vegaswikian, Vicarious, Waltpohl, Wayne Slam, Weisicong, Weston.pace, Wikiborg, Wmasterj, Wolfmankurd, Wshun, Xvani, YellowMonkey, Zundark, 188 anonymous edits

Commutativity of conjunction *Source:* <https://en.wikipedia.org/w/index.php?oldid=519901252> *Contributors:* Charles Matthews, CommBoy, Gamewizard71, Gregbard, Mboverload, Mihirk, OkPerson, PhilKnight, Reedy, Robertbowerman, 2 anonymous edits

Conjunction introduction *Source:* <https://en.wikipedia.org/w/index.php?oldid=563094196> *Contributors:* AllyUnion, Arthur Rubin, Big Bob the Finder, Bloodshedder, CBM, Constructive editor, Conversion script, Dooooot, Graham87, Gregbard, Jim.belk, Jiy, Justin Johnson, Kurykh, Oleg Alexandrov, Rich Farmbrough, Set theorist, Silverfish, 4 anonymous edits

Constructive dilemma *Source:* <https://en.wikipedia.org/w/index.php?oldid=561528458> *Contributors:* Arthur Rubin, Arunsingh16, Awis, Charles Matthews, Dooooot, Gamewizard71, Gregbard, Jim.belk, Jiy, Michael Hardy, Nozzer42, Oleg Alexandrov, PrinceXantar, Rich Farmbrough, Zenosparadox, 10 anonymous edits

Contraposition (traditional logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=549255744> *Contributors:* Amerindianarts, Aspects, BD2412, Barticus88, BenFrantzDale, CallMeLee, Charles Matthews, Cybercobra, Dendropithecus, Frietjes, Gregbard, Hansen Sebastian, JimStyle61093475, Julian Mendez, Mhss, Michael Hardy, Mild Bill Hiccup, N4nojohn, Oleg Alexandrov, OwenX, PsiXi, Red Slash, Robertbowerman, SGNDave, Singularity, Woohookity, 18 anonymous edits

Destructive dilemma *Source:* <https://en.wikipedia.org/w/index.php?oldid=499228688> *Contributors:* 478jjz, Adavidb, Cybercobra, Dooooot, Floridi, Greenm22, Gregbard, Jim.belk, Jiy, Lsuff, MagneticFlux, Michael Hardy, Oleg Alexandrov, Tesseran, Zenosparadox, Zensufi, 18 anonymous edits

Disjunction elimination *Source:* <https://en.wikipedia.org/w/index.php?oldid=563093538> *Contributors:* Arcadian, Arthur Rubin, CBM, Cimon Avaro, Conversion script, Cybercobra, Dooooot, Evercat, Graham87, Gregbard, GregorB, Hotteba, Jamesfisher, Jim.belk, Julian Mendez, Justin Johnson, Kurykh, Linas, Mets501, Michael Hardy, 2 anonymous edits

Disjunction introduction *Source:* <https://en.wikipedia.org/w/index.php?oldid=563093333> *Contributors:* Amillar, Calle, Conversion script, Esperant, Evercat, Graham87, Gregbard, Jim.belk, Jiy, Justin Johnson, Rectay, Sam Hocevar, Voomoo, 5 anonymous edits

Disjunctive syllogism *Source:* <https://en.wikipedia.org/w/index.php?oldid=540480341> *Contributors:* Anarchia, Arthur Rubin, AugPi, Bookandcoffee, Charles Matthews, Cimon Avaro, Conversion script, Dooooot, Dysprosia, ESKog, Evercat, Flash94, Gregbard, It Is Me Here, Jamelan, Jamesfisher, Jim.belk, Jiy, KSchutte, Kingdon, Kwhittingham, Lomn, Mhss, Nmajdan, Oleg Alexandrov, Pentasyllabic, Rich Farmbrough, Shadro, Sophomoric, Taak, Wlmg, 27 anonymous edits

Distributive property *Source:* <https://en.wikipedia.org/w/index.php?oldid=577493280> *Contributors:* 16@r, 2607:EA00:104:800:A440:7834:387E:DE46, Acroterion, Alecjansen, Alexb@cut-the-knot.com, Andrea105, Andres, Andrewman327, Andy85719, Anonymous Dissident, Arthena, AxelBoldt, BB-GUN101, Bando26, Baricus88, Ben Ben, Bentogoa, Bfigura, Blaxthos, Bobo192, Bsadowski1, CallofDutyboy9, Chewings72, Chris Roy, Cliff, DVdm, Davejohnsan, DavidLeighEllis, Dictouray, Discospinster, Dissident, Donner60, Dysprosia, ESKog, EdC, EmijJ, Engeele, Enviroboy, EuroCarGT, Evershade, Excirial, Exazakin, False vacuum, Favonian, Forkloop, FrozenMan, GaborLajos, Giftlike, Goldkingut5, Gphiliip, Grafen, Gregbard, Hello71, Ichudov, Ideyal, Isnow, Ivashikhmin, J36miles, Jackfork, Janice Vian, Jiddisch, Jojbutton, Jokes Free4Me, Jusdafax, Keenan Pepper, Khazar, Kitkat1234567880, Ladislav the Posthumous, Linas, Lycte, Malcohol, Marek69, Markus Krötzsch, Martin451, Melchoir, Mhaitham.shammaa, Michael Slone, Mm40, Montchav, Mykej, NERIC-Security, Nezzadar, Nodmonkey, NuclearWarfare, Numbo3, Octahedron80, Onkel Tuca, Orphan Wiki, Oxfordwang, Patrick, Paul August, Pichpitch, P Kramer2021, Pmi1924, Quondum, Rgdboer, Romanm, Ronjhones, Salgueiro, Salix alba, Sandeep.ps4, Saul34, Shauna9876, Simeon, Slon02, Smimram, Sp3dyp hil, Squandermania, TCN7JM, Tarquin, The Thing That Should Not Be, Tobias Bergemann, Toby Bartels, Trovatore, TyA, UNV, Vegaswikian, Vibhjain, Wywin, Xavic69, Youssefsan, Zarcadia, 265 anonymous edits

Double negative elimination *Source:* <https://en.wikipedia.org/w/index.php?oldid=573417564> *Contributors:* Adambiswanger1, AugPi, Blaisorblade, Brighterorange, CBM, Charles Matthews, Conversion script, Dysprosia, EmilJ, Evercat, Extrapiramidale, Fram, Gamewizard71, Grafen, Graham87, Gregbard, Hpvp, JMRyan, Jamelan, Julian Mendez, Justin Johnson, Kilva, Mets501, Mhym, Noamz, Oleg Alexandrov, Paul August, Poor Yorick, Populus, Sam Hoevar, Twilsonb, 5 anonymous edits

Existential generalization *Source:* <https://en.wikipedia.org/w/index.php?oldid=532332857> *Contributors:* Fan Singh Long, Gregbard, Loadmaster, Yarou

Existential instantiation *Source:* <https://en.wikipedia.org/w/index.php?oldid=549131788> *Contributors:* CBM, Gregbard, InverseHypercube, Quondum, Theinactivist, 1 anonymous edits

Exportation (logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=560971009> *Contributors:* Arthur Rubin, Charles.w.chambers, Dooooot, Gregbard, HenningThielemann, Jitse Niesen, Michael Hardy, SwisterTwister

Hypothetical syllogism *Source:* <https://en.wikipedia.org/w/index.php?oldid=549427764> *Contributors:* José, Algebraist, Arthena, Arthur Rubin, BobHelix, Brad7777, Chalst, Charles Matthews, Chenzw, Chzz, Cybergobra, Dooooot, Dysprosia, Flosfa, Gregbard, Haza-w, I R Solecism, Jamelan, JamesBWatson, Jim.belk, Jiy, Lsloan, Mel Etitis, Mhss, Noam, Oleg Alexandrov, RJFJR, Rossami, SchreiberBike, Simeon, Squids and Chips, Taak, The Parting Glass, 42 anonymous edits

List of valid argument forms *Source:* <https://en.wikipedia.org/w/index.php?oldid=566379229> *Contributors:* Gregbard, Josh3580, Pdyoung, Wavelength, 6 anonymous edits

Material implication (rule of inference) *Source:* <https://en.wikipedia.org/w/index.php?oldid=569956127> *Contributors:* Arthur Rubin, CarrieVS, Dooooot, Gregbard, Incnis Mrsi, Jason Quinn, Jochen Burghardt, KSchutte, Quondum, Toby Bartels, 13 anonymous edits

Modus non expiciens *Source:* <https://en.wikipedia.org/w/index.php?oldid=480226157> *Contributors:* Gregbard, Michael Hardy

Modus ponendo tollens *Source:* <https://en.wikipedia.org/w/index.php?oldid=547805939> *Contributors:* Aaagmnr, Amsoman, Anarchia, Arthur Rubin, Carolynparrishfan, Dooooot, Evercat, Gobonobo, Gregbard, Jim.belk, Macai, Mike Rosoft, Mikeblas, Rmtzr, Srne, 14 anonymous edits

Modus ponens *Source:* <https://en.wikipedia.org/w/index.php?oldid=556062970> *Contributors:* 20040302, ABF, Alexb@cut-the-knot.com, Anarchia, Andre Engels, Andyfugard, Antandrus, AugPi, AxelBoldt, BAxelrod, CBM, Cgwaldman, Charles Matthews, Chenzw, Classicaelecon, Conversion script, Cybercobra, DBigXray, Dandy, Don Warren, Dooooot, Dysprosia, Elwikipedista, Ercarer, Firefirefire2, Frecklefoot, Giftlite, Gobonobo, Graham87, Gregbard, Hakeem,gadi, Helia, Incnis Mrsi, Jraxis, Jamelan, Jim.belk, Jiy, Jo3sampl, Jonon, Jrquinpidista, KSmrq, Larry_Sanger, Leonard G., Leuko, Liftarn, Logperson, M7, MRG90, Machine Elf 1735, Magioladitis, Marabeen, Matt Crypto, Mbarbier, Mhss, Michael Hardy, Nathanieljones15, Neilc, Nicolas.Wu, Noam, Nortexoid, Notapipe, Obradovic Goran, Otto ter Haar, Pacogo7, Planeswalkerduke, Policron, RexNL, Rich Farmbrough, Rjwilmsi, Robofish, Romnempire, Rootbeer, Ruakh, Ruud Koot, Rygasus, Schoen, Shawn81, Siroxo, Spencerk, Spirita, Steel, Svartskägg, Svick, Sword, Tarquin, Toytoy, Trustedgunny, Trylks, Tyrol5, Vecter, Velvetron, Voidvector, Waldir, WhyBeNormal, Wingman417, Wvbailey, Yayay, Yintan, Yocko, Zundark, 100 anonymous edits

Modus tollens *Source:* <https://en.wikipedia.org/w/index.php?oldid=577393117> *Contributors:* 1exec1, 20040302, Alastair Carnegie, Anarchia, Andersæøå, Andrewa, Andyfugard, Anime Addict AA, Arno Peters, AugPi, Auntoff6, AxelBoldt, Banno, Bobblond, Brianjd, Cwgwaldman, Chaos5023, Charles Matthews, Charm, Chris the speller, Chuunun Baka, Conversion script, Doctezee, Dictionee, Dooooot, Dysprosia, El C, Elwikipedista, EricBright, Exfenestracide, FF2010, Farrell, Ferengi, Flosfa, François Robere, Geoffrey.landis, Gobonobo, Graham87, Gregbard, Hiddenhearts, Highlandwolf, Human step, Husond, Iantresman, Icelight, Izno, Jraxis, JSquish, Jamelan, Jim.belk, Jiy, Jorend, Jrquinlisk, KDS4444, KYPark, Keenan Pepper, Kenyon, Kephir, Kingturtle, KnightRider, Larry_Sanger, Leland McInnes, Liempt, Lightlowemon, Michaelwy, Mr Gronk, Nayuki, Neilc, Neutrality, Nivaca, Notapipe, Nullie, Obradovic Goran, Oleg Alexandrov, Otto ter Haar, Pampas Cat, Pgan002, Philbarker, Richard001, Rifleman 82, Robina, Robofish, Schoen, Shawnc, Spencerk, Steel, Sundar, Tesseract2, The Cunctator, The Wonky Gnome, TheJRLinguist, Tim bates, Tojasonharris, Toytoy, Voidvector, W33v11, Waldir, Wanderer57, Wejstheman, Wwoods, 135 anonymous edits

Negation as failure *Source:* <https://en.wikipedia.org/w/index.php?oldid=575900312> *Contributors:* Albertzeyer, Baosheng, CBM, Eusebius, Gregbard, GregorB, H2g2bob, Hamaryns, Kermesbeere, LittleDan, Logperson, Mhss, Nosbig, NotInventedHere, Oleg Alexandrov, Robert Kowalski, Robertbowerman, Runewiki777, Simeon, Tizio, Una Smith, Vlfscitz, 17 anonymous edits

Resolution (logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=574886367> *Contributors:* AHMartin, Acipsen, Antonielly, AugPi, Barnardb, Byelf2007, CBM, Ceilican, Charles Matthews, Cloudyed, D.Lazard, DVD R W, Ekspilo, Esoteric, Fresheneesz, Gaius Cornelius, Gamewizard71, Gregbard, GregorB, Gubbubu, Hannes Eder, Henning Makholm, Hquain, Irbisgreif, JLD, Jiy, Jochen Burghardt, Jorend, Jpbown, Kku, Kmweber, Lichn, Lagener, Lifetime, LittleWink, Macrakis, Matt Kovacs, MattGiua, Methossant, Mikolasj, MoraSique, Mu, Mukerjee, N4nojohn, Ogai, Pdbogen, Peter M Gerdes, Pgr94, Porphyrus, Prohlep, RatnimSnav, Ruud Koot, Shmenonpie, Simeon, Stephan Schulz, SwordAngel, Slawomir Bialy, T boyd, Termar, Thaukko, Tijfo098, Tizio, Tobias Bergemann, Toddst1, Wjejskenewr, Xiteddragon, Zero sharp, 60 anonymous edits

SLD resolution *Source:* <https://en.wikipedia.org/w/index.php?oldid=544824970> *Contributors:* Albertzeyer, AlexDitto, AugPi, Carbo1200, Eusebius, Grafen, Gregbard, Logperson, Momet, PaulBrinkley, Pgr94, Tijfo098, Tomaxer, 22 anonymous edits

Simplification *Source:* <https://en.wikipedia.org/w/index.php?oldid=548428901> *Contributors:* Algebraist, AnnaFrance, Arthur Rubin, CBM, Dfrg.msc, Dooooot, Father Goose, Fratrep, Gdr, Gregbard, Grumpyyoungman01, Jim.belk, Jiy, Markus Prokott, Nicko6, Oleg Alexandrov, Owen, Silvergoat, Smij, 7 anonymous edits

Structural rule *Source:* <https://en.wikipedia.org/w/index.php?oldid=543793945> *Contributors:* Byelf2007, Charles Matthews, Fplay, Gregbard, Hans Adler, Kaustuv, Mattg82, Mhss, Noamz, Pi zero, RDBury, RobinK, Ruakh, STHayden, Soumyasch, 2 anonymous edits

Tautology (rule of inference) *Source:* <https://en.wikipedia.org/w/index.php?oldid=558523383> *Contributors:* BiT, ENeville, Gregbard, Michael Hardy, Tbhotch, Tinton5, 2 anonymous edits

Transposition (logic) *Source:* <https://en.wikipedia.org/w/index.php?oldid=553188103> *Contributors:* Alan U. Kennington, Amerindianarts, AxelBoldt, BD2412, Bdesham, BenFrantzDale, Bgeer, Charles Matthews, Circus, Cmdrjameson, Crystallina, Dominus, Dooooot, Drae, Fluffernutter, Gaius Cornelius, Gerhardvalentin, Greenwoodtree, Gregbard, Kwamikagami, Mate2code, Mhss, Michael Hardy, Mrwojo, Noideta, Quondum, Reedy, 15 anonymous edits

Universal generalization *Source:* <https://en.wikipedia.org/w/index.php?oldid=543504971> *Contributors:* AugPi, CBM, Dcoetzee, Dieter Simon, Fyedernoggersnodden, Gregbard, Hans Adler, Ilmari Karonen, Jim.belk, Julian Mendez, Lotje, Mhss, Michael Hardy, Minimiscience, Nieske, Op47, Peterdjones, Raymer, Rettetast, Vegaswikian, Vsaraph, 8 anonymous edits

Universal instantiation *Source:* <https://en.wikipedia.org/w/index.php?oldid=541182677> *Contributors:* Active Banana, Akrabbim, Awarded, Burket, Byelf2007, CBM, Charles Matthews, Eusebius, Fan Singh Long, Gregbard, JamesBWatson, Jim.belk, Kku, Mhss, Michael Hardy, Nortexoid, RatnimSnav, Simeon, Vsaraph, 7 anonymous edits

Case analysis *Source:* <https://en.wikipedia.org/w/index.php?oldid=535462436> *Contributors:* Anonymous Dissident, Charles Matthews, Classicaelecon, Dbtfz, Fvw, Gamewizard71, Gregbard, John184, Michael Hardy, PamD, Phantomsteve, 7 anonymous edits

Consequuntia mirabilis *Source:* <https://en.wikipedia.org/w/index.php?oldid=567868115> *Contributors:* AvicAWB, Byelf2007, Dhanyaavaada, Evud, Gobonobo, Gregbard, JoDonHo, Kilbosh, Machine Elf 1735, Magmalox, Owlbuster, Peter Damian (temporary), RL0919, Rich Farmbrough, Vanished User 12039213092139821, 3 anonymous edits

Contraposition *Source:* <https://en.wikipedia.org/w/index.php?oldid=574296249> *Contributors:* Amerindianarts, Andkore, Antonone, Aquillyne, AtiwH, BD2412, Bamgooly, Blindman shady, Burning.flamer, Byelf2007, CBM, Carabinieri, Dashed, Dasunstr3r, Deleet, Dendropithicus, Dycedarg, Elliottwolf, Epbr123, Fratrep, Gimmetrow, Googl, Gregbard, Harold f, Henneyj, HereToHelp, Iridescent, Ifxld64, JSquish, Javalenok, Jaymax, Jaysn1, Jheiv, Jimmaths, John, Katovatzschyn, Kwanmmi, Makeswell, McSly, Metaprimer, Mild Bill Hiccup, NY Amateur, NickPenguin, Okloster, Otto ter Haar, Pointylittlethingy, Prgrmr@wrk, Raeven0, Red Slash, SchreiberBike, ScottJ, Shinju, Simeon, Stevenj, Stismail, Tide rolls, Widdma, Wilson44691, XDanielx, 82 anonymous edits

Double negation *Source:* <https://en.wikipedia.org/w/index.php?oldid=531855067> *Contributors:* Angela, CBM, Cybercobra, David ekstrand, Dysprosia, Fratrep, Gregbard, TFCforever, Wik, Wvbailey, 3 anonymous edits

Frege's theorem *Source:* <https://en.wikipedia.org/w/index.php?oldid=564793687> *Contributors:* Algebraist, AugPi, CBM, CRGreathouse, Chatsam, Chinju, El Caro, Elwikipedista, Geometry guy, Gregbard, Joth, NawlinWiki, Neodop, NorthernThunder, Oleg Alexandrov, Rgheck, Shadow1, 4 anonymous edits

Idempotency of entailment *Source:* <https://en.wikipedia.org/w/index.php?oldid=575112044> *Contributors:* Chalst, GTBacchus, Gregbard, MadMax, Melaen, RJFJR, Spockticus, Wood Thrush, 4 anonymous edits

Law of excluded middle *Source:* <https://en.wikipedia.org/w/index.php?oldid=57181954> *Contributors:* Aberrantgeek, Altenmann, Ameulen11, Andre Engels, Andrewrp, Apostolos Margaritis, ArgiebargleIV, Asmeurer, AugPi, BIL, Before My Ken, Bgwhite, BirgitteSB, Byelf2007, CesarB, Charles Matthews, Cherkash, Chetvorno, Chinju, CiaPan, Colonies Chris, Conversion script, Cybercobra, Dbenbenn, Dcoetzee, DeaconJohnFairfax, Dmmaus, Dominus, Doradus, Drennie, Dweinberger, Eigenlambda, Elias, Error, Evercat, Flyingpasta, Freequark, Gaius Cornelius,

Gamewizard71, George100, Giftlite, Gnothiseautonpantarei, Gregbard, Grumpyyoungman01, Guppyfinsoup, Guy Peters, Guyd, Hackwrench, Hairy Dude, Houistesmoiras, Husond, IHendry, Iamthedeus, Ian Pitchford, Jon Awbrey, Jorend, Justin Johnson, KathrynLybarger, Khazar2, Kocio, LC, LOL, Lambiam, Laocoón11, Larek, Larry_Sanger, Leibniz, Lestrade, Lim Wei Quan, Linas, LittleWink, Loadmaster, Lysdexia, Magmalex, Mauro Lanari, Mets501, Mhss, Mild Bill Hiccup, Misza13, Mray1, Nagelfar, Nortexoid, Optigan13, Patrick, Pearle, Pegua, Peruvianllama, Petrus Damianus, Philippschaumann, Populus, RGGehue, Rangek, Red van man, Reedy, Riki, Rjwilmsi, Rtc, Scaro, Schneelocke, Sdorrance, Sethmahoney, Simetrical, Smartcat, Snoyes, Soler97, Sommers, Spontini, Squandermania, Stevnewb, StradivariusTV, Tarquin, Teemu Leisti, The Real Marauder, Tijfo098, TitoAssini, Tkuvhlo, Velho, Wapcaplet, Wham Bam Rock II, WhyBeNormal, WikiPendant, With goodness in mind, Wjmallard, Wvbailey, Xanzzibar, Yinweichen, Zonko, Zundark, 86 anonymous edits

Law of identity *Source:* <https://en.wikipedia.org/w/index.php?oldid=574411798> *Contributors:* 2607:F4F0:8:922:258C:BFDD:49EF:D834, Academic Challenger, Amicuspublius, Andyjsmith, Andysoh, Avedomni, BD2412, Bballguy7100, BenB4, Billinghamst, Brian0918, Byelf2007, C6541, Causa sui, Cauté AF, Charles Matthews, Csisitic, Cybercobra, DAGwyn, Der Blaue Wolf, Doidimais Brasil, Ermalve, Exomniun, Fishtron, Frzl, Fvw, Galoubet, Gamewizard71, Gehue, George100, Gerhardvalentin, Giraffedata, GoThe, Gonzonoir, Gregbard, Gwalla, Hayats, Herb-Sewell, Herbee, Hofman stern Ihcoy, InverseHypercube, Jan eissfeldt, Karbinski, Kentym, KoshVorlon, Lambiam, LoveMonkey, Manfi, Manoperson, Masterpiece2000, Metaprimer, Mgreenbe, Mhss, Michael Rogers, MilesAgain, Mladek, Miinck, N4nojohn, Nabeth, Nbarth, NinjaLore, Oknavezad, Parodoxe allemand, Peter Damian, Petri Krohn, Profaneprimate, RGGehue, RL0919, RiggdzinPhurba, Scarpy, Shii, Skarebo, Skomorokh, Smyth, Steven J. Anderson, Stevertigo, Superbeecat, Teemu Leisti, Tim bates, Tito-, Trivialist, WikHead, WikiPendant, WikiSlasher, Willking1979, Wtmitchell, Xenfreak, XxAvalanchexX, Zurishaddai, 80 anonymous edits

Law of noncontradiction *Source:* <https://en.wikipedia.org/w/index.php?oldid=567623953> *Contributors:* 198.207.223.xxx, Agüeybaná, Alexf, AnOddName, Anomalocaris, AugPi, BenMcLean, Charles Matthews, Conversation script, Daniel J. Forman, Dod1, Don of Cherry, Dysprosia, Ermalve, Evercat, Farzaneh, Gamewizard71, Giorgiomugnaini, Gregbard, Hairy Dude, Heyitspeter, Infinite.magic, Isaac Dupree, Jagged 85, Janburse, Jon Awbrey, Jonathancamp, Julian Mendez, LC, Larek, Larry_Sanger, Lenticel, Mdd, Messenger88, Metapunk, Mhss, Michael Hardy, NY Amateur, OAC, OldNick, Paralipsis, Peter Damian (old), Pnrr, Retodon8, Ryguasu, Schneelocke, SchreiberBike, Serenity id, Smyth, Storm77, Tagishsimon, Teemu Leisti, Thane, TheRepairMan, Tobias Hoevekamp, Unzerlegbarkeit, Velho, Wapcaplet, WikiPendant, With goodness in mind, Wragge, YellowMonkey, Zakinstein, 38 anonymous edits

Monotonicity of entailment *Source:* <https://en.wikipedia.org/w/index.php?oldid=543793834> *Contributors:* Chalst, Constructive editor, Floridi, Gregbard, Helvetius, JanusDC, Josh Parris, Joyous!, Meredyth, Open2universe, Sabik, Smmurphy, 9 anonymous edits

Principle of explosion *Source:* <https://en.wikipedia.org/w/index.php?oldid=571720545> *Contributors:* An0nym0us7r011, AugPi, Btyner, Byelf2007, Cayafas, Cheesefather, Chrismorey, DD2K, Dbtfz, Dominus, Doublethink1984, Dualus, Duland21, Epastore, Fibonacci, Fundoctor, Gregbard, Hairy Dude, Heyitspeter, ILikeThings, IdNotFound, Incnis Mrsi, Jon Awbrey, Joshua Issac, Jushi, Kachoomey, Lambiam, Leibniz, LesterRoquefort, Lysdexia, Magmalex, Majorly, Mark Arsten, Mhss, Michael Hardy, Nick Number, Otr500, Punk physicist, Rh, Ruakh, Ruud Koot, Schneelocke, Shadowcrimejas, Smyth, Snied, Tgeorgescu, ThirdParty, Thumperward, Toddst1, 72 anonymous edits

Proof by contradiction *Source:* <https://en.wikipedia.org/w/index.php?oldid=577174701> *Contributors:* 20040302, Aerobird, Aflyhorse, AgadaUrbanit, Alexander.mitsos, Algebraist, Altenmann, AltiusBimm, Andersersej, Andre Engels, AndrewA, Arkwatem, AxelBoldt, Axeman89, B4hand, BD2412, Barbara Shack, Bart133, Bastian964, BIT, Bkell, Brian0918, Burn, Byelf2007, CBM, CZeke, Centrx, Chamaeleon, Chamelein, Chancemill, Charles Matthews, Chas zzz brown, Chrismear, Chun-hian, Colinebn, Conversation script, Cortina, Cretog8, Dave3457, Dbtfz, Dcljr, DexDor, Dkg11hu, Doctor Whom, Dominus, DopefishJustin, Dr.K., DrewRobinson, Dungodung, Duoduo, EdH, Electriceel, Emx, Eprb123, Everyking, Fibonacci, Flarity, FvdP, Giftite, GlassFET, Glenn L, Graham87, Gregbard, Grossdomestic, Grumpyyoungman01, Gubbubu, Harvestdancer, Hdante, Hopiakuta, Hydrox, Ideyal, Ioscius, Janet Davis, Jason Quinn, Jdl22, Jimmaths, Jochen Burghardt, Jonsafaria, JuanPaBJ16, Kajisol, Katznik, Kazvorpal, Kerowren, Kesa, Kidburla, Kobayan, Lacrimous, Lambiam, Larry Sanger, Lee Daniel Crocker, LegendFSL, Lemuel Gulliver, Leonariso, Liko81, Loodog, LookingGlass, Luciengav, MagicOgre, Magnus Manske, Majestic-chimp, Makcat, Man vyi, MarcellLionheart, Markkbilbo, Mattbuck, Mattman00000, Mav, Mehdi, Melsaran, Mgiganteus1, Mhym, Michael Hardy, Michaelcarraher, Mindmatrix, Mohan ravichandran, Nbarth, Ncosmob, Neale Monks, NeighborTotoro, Neonguru, Nikie42, Nsaa, Oliver Pereira, Omphaloscope, OpenFuture, Out180, Pampas Cat, Password is DOB, Patrick, Patrick Zanon, Paul August, Pedro, Peter Kwok, PetroniusArb, Pgk, Pliny, Pmbcomm, Populus, Psiphorg, PurplePlatypus, Qwasty, RayShimon, Rd232, Rdsmith4, Reetep, Revolver, Roadrunner, Robofish, Rodrigo Rocha, Romann, Roybb95, SV Resolution, Sam Spade, Sampi, Scott Martin, ScottForschler, Seanjacksonc, Severa, Shirulashem, Simpsons contributor, Skysmith, TERdON, Tarquin, The Anome, The Evil Spartan, TheBusiness, Toytoy, UtherSRG, Vclam068, Vesta Zenobia, Vicki Rosenzweig, Wafulz, Wje, X1011, Xiaphias, Xin-Xin W, Zafiroblue05, Zundark, 177 anonymous edits

Reductio ad absurdum *Source:* <https://en.wikipedia.org/w/index.php?oldid=576585216> *Contributors:* 20040302, Aflyhorse, AgadaUrbanit, Anarchangel, Angel ivanov angelov, Asnac, BenAveling, BertSeghers, Bmearns, Bomac, Byelf2007, Cardamon, Cengime, Centrx, Chetvorno, Ciphers, Dashed, Dave3457, Diyan.boyanov, Diza, Dominus, E1618978, ForestAngel, GDW13, Gapooh007, Giftlite, Gobonobo, Gregbard, Hello71, HereToHelp, Hibbleton, HumphreyW, Iamthedeus, InverseHypercube, JSWIFT13, JamesBWatson, Jordan Brown, Jordgette, Jrjspencer, KirbenS, LilyKitty, Loodog, Lugia2453, Majestic-chimp, Mandarax, Martarius, Matthew Fennell, Mendaliv, Michael Hardy, Mindmatrix, Much noise, N2e, NY Amateur, Nevi1, Newagelink, Occultations, Ohms law, OpenFuture, PamD, Pappaj33, Paul August, Pollinossss, Reydeyo, Richard Arthur Norton (1958-), Ronjhones, Savethemooses, Scythematic, Sensemaker, Snow Blizzard, Spezed, SpikeToronto, SudoZero, TheAlphaWolf, TheConduqtor, Thedarkknigh491, Theinsomniac4life, Thnidu, Tired time, Tojasonharris, Transity, Uranium grenade, Williamjacobs, Wizard191, XP1, Y, Yamaha5, You Can Act Like A Man, Zebrasil, Όλοτρος, 101 anonymous edits

Image Sources, Licenses and Contributors

Image:Hasse diagram of powerset of 3.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Hasse_diagram_of_powerset_of_3.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* KSmrq

File:T 30.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:T_30.svg *License:* Creative Commons Zero *Contributors:* User:Mini-floh

File:Vennandornot.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Vennandornot.svg> *License:* Public Domain *Contributors:* Lipedia

File:LogicGates.GIF *Source:* <https://en.wikipedia.org/w/index.php?title=File:LogicGates.GIF> *License:* Creative Commons Attribution 3.0 *Contributors:* Vaughan Pratt

File:DeMorganGates.GIF *Source:* <https://en.wikipedia.org/w/index.php?title=File:DeMorganGates.GIF> *License:* Creative Commons Attribution 3.0 *Contributors:* Vaughan Pratt

Image:Venn11.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn11.svg> *License:* Public Domain *Contributors:* Lipedia

Image:Venn01.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn01.svg> *License:* Public Domain *Contributors:* Lipedia

Image:Venn00.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn00.svg> *License:* Public Domain *Contributors:* Lipedia

Image:Venn10.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn10.svg> *License:* Public Domain *Contributors:* Lipedia

Image:Venn0001.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0001.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Tony Wills, Waldir, 9 anonymous edits

Image:Venn1110.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1110.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn0111.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0111.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn1000.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1000.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn1011.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1011.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn0110.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0110.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Herbythyme, Jarekt, Mate2code, Waldir, 3 anonymous edits

Image:Venn1001.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1001.svg> *License:* Public Domain *Contributors:* CommonsDelinker, J.delanoy, Mate2code, Waldir, 1 anonymous edits

Image:Venn1101.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1101.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn0101.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0101.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir, Wknight94, 1 anonymous edits

Image:Venn0011.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0011.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Begriffsschrift connective1.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Begriffsschrift_connective1.svg *License:* Public Domain *Contributors:* Guus Hoekman

File:Solar eclipse 1999 4 NR.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Solar_eclipse_1999_4_NR.jpg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* Luu Viatour

File:ICE 3 Fahlenbach.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:ICE_3_Fahlenbach.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Sebastian Terfloth User:Sese_Ingolstadt

File:Necessary and sufficient venn (set) diagram.svg *Source:* [https://en.wikipedia.org/w/index.php?title=File:Necessary_and_sufficient_venn_\(set\)_diagram.svg](https://en.wikipedia.org/w/index.php?title=File:Necessary_and_sufficient_venn_(set)_diagram.svg) *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Javalemok

Image:Venn0000.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0000.svg> *License:* Public Domain *Contributors:* ABF, CommonsDelinker, Mate2code, Waldir

Image:Venn0100.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0100.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn0010.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0010.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn1111.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1111.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir, 2 anonymous edits

Image:Venn1010.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1010.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

Image:Venn1100.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1100.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:OEISicon light.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:OEISicon_light.svg *License:* Public Domain *Contributors:* Billingham, Mate2code, Senator2029

Image:Rotate left logically.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_left_logically.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

Image:Rotate right arithmetically.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_right_arithmetically.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

Image:Rotate right logically.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_right_logically.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

Image:Rotate left.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_left.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

Image:Rotate right.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_right.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

Image:Rotate left through carry.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_left_through_carry.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

Image:Rotate right through carry.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Rotate_right_through_carry.svg *License:* Creative Commons Attribution-ShareAlike 3.0 Unported *Contributors:* en:User:Cburnett

File:Sat reduced to Clique from Sipser.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Sat_reduced_to_Clique_from_Sipser.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Thore Husfeldt (talk)

File:Schaefer's 3-SAT to 1-in-3-SAT reduction.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Schaefer's_3-SAT_to_1-in-3-SAT_reduction.gif *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Jochen Burghardt

File:Boolean satisfiability vs true literal counts.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Boolean_satisfiability_vs_true_literal_counts.gif *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Jochen Burghardt

File:Backtracking-no-backjumping.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Backtracking-no-backjumping.svg> *License:* Public Domain *Contributors:* Tizio, WikipediaMaster

Image:LAlphabet T table.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_T_table.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet T.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_T.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet NAND table.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_NAND_table.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet NAND.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_NAND.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet IFTHEN table.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_IFTHEN_table.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet IFTHEN.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_IFTHEN.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet NOTP table.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_NOTP_table.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet NOTP.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_NOTP.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

Image:LAlphabet FI table.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_FI_table.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis, Stephen

Image:LAlphabet FI.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:LAlphabet_FI.jpg *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* Smerdis

File:K-map 2x2 1,3,4.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:K-map_2x2_1,3,4.svg *License:* GNU Free Documentation License *Contributors:* en:User:Cburnett

File:K-map 2x2 2,3,4.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:K-map_2x2_2,3,4.svg *License:* GNU Free Documentation License *Contributors:* en:User:Cburnett

File:K-map 2x2 1,2,3,4.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:K-map_2x2_1,2,3,4.svg *License:* GNU Free Documentation License *Contributors:* en:User:Cburnett

File:BDD2pdag.png *Source:* <https://en.wikipedia.org/w/index.php?title=File:BDD2pdag.png> *License:* GNU Free Documentation License *Contributors:* Original uploader was RUN at en.wikipedia

File:BDD2pdag simple.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:BDD2pdag_simple.svg *License:* GNU Free Documentation License *Contributors:* User:Selket and User:RUN (original)

File:Venn diagram gr la ru.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_diagram_gr_la_ru.svg *License:* Public Domain *Contributors:* AnonMoos, MagnusFit, Mate2code, Timwi

file:Nuvola apps atlantik.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Nuvola_apps_atlantik.png *License:* GNU Lesser General Public License *Contributors:* AVRS, Alno, Alphax, Bayo, Drilnoth, Hyju, It Is Me Here, Mindmatrix, Rocket000, Tbleher

Image:venn-diagram-AB.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn-diagram-AB.svg> *License:* GNU Free Documentation License *Contributors:* Ezekiel25q, Jusjih, SilverStar, THEN WHO WAS PHONE?, 4 anonymous edits

File:Venn0001.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0001.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Tony Wills, Waldir, 9 anonymous edits

File:Venn0111.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0111.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn0110.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0110.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Herbythyme, Jarekt, Mate2code, Waldir, 3 anonymous edits

File:Venn0010.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0010.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn1010.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1010.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn 1000 0000 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1000_0000_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0110 1000 1000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0110_1000_1000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0100 0000 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0100_0000_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0010 0000 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0010_0000_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 1000 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_1000_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0100 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0100_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0010 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0010_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0001 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0001_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0100 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0000_0100_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0010 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0000_0010_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0001 0110.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0001_0110.png *License:* Creative Commons Attribution 3.0 *Contributors:* Mate2code

File:Venn 0000 0001 0000 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0001_0000_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0001 0000.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0000_0001_0000.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0000 0100.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0000_0000_0100.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0000 0010.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0000_0000_0010.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

File:Venn 0000 0000 0000 0001.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0000_0000_0001.png *License:* Creative Commons Attribution 3.0 *Contributors:* Lipedia

Image:Venn4.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn4.svg> *License:* Public Domain *Contributors:* Original uploader was at

Image:Venn5.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn5.svg> *License:* Public Domain *Contributors:* Original uploader was at

Image:Venn6.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn6.svg> *License:* Public Domain *Contributors:* Original uploader was Kopophex at en.wikipedia

Image:Venn's four ellipses construction.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn's_four_ellipse_construction.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* RupertMillard

Image:CirclesN4xb.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:CirclesN4xb.svg> *License:* Public Domain *Contributors:* User:Marekich

File:Symmetrical 5-set Venn diagram.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Symmetrical_5-set_Venn_diagram.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Cmglee

File:6-set_Venn_diagram.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:6-set_Venn_diagram.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Cmglee

Image:Venn-three.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn-three.svg> *License:* Public Domain *Contributors:* Antti Tanhuapää

Image:Edwards-Venn-four.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Edwards-Venn-four.svg> *License:* GNU Free Documentation License *Contributors:* User:HB

Image:Edwards-Venn-five.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Edwards-Venn-five.svg> *License:* GNU Free Documentation License *Contributors:* User:HB

Image:Edwards-Venn-six.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Edwards-Venn-six.svg> *License:* GNU Free Documentation License *Contributors:* Darapti, Interiot

File:Venn3tab.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn3tab.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:JohnJones

File:Venn1000.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1000.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn01.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn01.svg> *License:* Public Domain *Contributors:* Lipedia

File:Venn10.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn10.svg> *License:* Public Domain *Contributors:* Lipedia

File:Venn1011.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1011.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn0100.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0100.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn1100.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1100.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Boolean functions like 1000 nonlinearity.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Boolean_functions_like_1000_nonlinearity.svg *License:* Public Domain *Contributors:* Lpedia

File:0001 0001 0001 1110 nonlinearity.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:0001_0001_0001_1110_nonlinearity.svg *License:* Public Domain *Contributors:* Mate2code

File:Venn1101.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1101.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Venn 0001 1011.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0001_1011.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0001 0001.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0001_0001.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0000 1010.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_1010.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0000 0001.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0001.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0110 1001.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0110_1001.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0110 0110.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0110_0110.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0000 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Multigrade operator XOR.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Multigrade_operator_XOR.svg *License:* Public Domain *Contributors:* Logan, Mate2Code, Mate2code

File:X-or.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:X-or.svg> *License:* Public Domain *Contributors:* User:The Unbound

File:Venn 0101 0101.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0101_0101.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0011 1100.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0011_1100.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn00.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn00.svg> *License:* Public Domain *Contributors:* Lipedia

File:Venn 1011 1011.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1011_1011.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1011 1101.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1011_1101.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0101 1010.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0101_1010.svg *License:* Public Domain *Contributors:* User:Lipedia

File:XOR ANSI Labelled.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:XOR_ANSI_Labelled.svg *License:* Public Domain *Contributors:* Inductiveload

File:Z2^4; Cayley table; binary.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Z2^4;_Cayley_table;_binary.svg *License:* Creative Commons Attribution 3.0 *Contributors:* Mate2code

Image:Indicator function illustration.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Indicator_function_illustration.png *License:* Public Domain *Contributors:* Oleg Alexandrov

File:Multigrade operator XNOR.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Multigrade_operator_XNOR.svg *License:* Public Domain *Contributors:* Logan, Mate2Code, Mate2code

File:Multigrade operator all or nothing.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Multigrade_operator_all_or_nothing.svg *License:* Public Domain *Contributors:* CommonsDelinker, Mate2Code, Mate2code, Tony Wills

File:Venn1001.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1001.svg> *License:* Public Domain *Contributors:* CommonsDelinker, J.delanoy, Mate2code, Waldir, 1 anonymous edits

File:Venn 1001 1001.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1001_1001.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1000 0001.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1000_0001.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1100 0011.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1100_0011.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn11.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn11.svg> *License:* Public Domain *Contributors:* Lipedia

File:Venn 1101 1011.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1101_1011.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1010 0101.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1010_0101.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Multigrade operator AND.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Multigrade_operator_AND.svg *License:* Public Domain *Contributors:* Logan, Mate2Code, Mate2code

File:Venn 0000 0011.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0011.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0011 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0011_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0001 0101.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0001_0101.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0000 0101.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0000_0101.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0001 0100.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0001_0100.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0011 0000.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0011_0000.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0001 0000.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0001_0000.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1111 1011.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1111_1011.svg *License:* Public Domain *Contributors:* User:Lipedia

File:AND Gate diagram.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:AND_Gate_diagram.svg *License:* Public Domain *Contributors:* Juiced lemon, Kilom691, Palmtree3000

File:Venn 0111 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0111_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Multigrade operator OR.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Multigrade_operator_OR.svg *License:* Public Domain *Contributors:* Logan, Mate2Code, Mate2code

File:Venn 0111 0111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0111_0111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0101 0111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0101_0111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 0101 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_0101_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1101 0111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1101_0111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1100 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1100_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1101 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1101_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Venn 1011 1111.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_1011_1111.svg *License:* Public Domain *Contributors:* User:Lipedia

File:Or-gate-en.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Or-gate-en.svg> *License:* GNU Free Documentation License *Contributors:* User:Helix84

File:XNOR ANSI Labelled.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:XNOR_ANSI_Labelled.svg *License:* Public Domain *Contributors:* Inductiveload

File:Venn0101.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn0101.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir, Wknight94, 1 anonymous edits

File:Venn1110.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn1110.svg> *License:* Public Domain *Contributors:* CommonsDelinker, Mate2code, Waldir

File:Square of opposition, set diagrams.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Square_of_opposition,_set_diagrams.svg *License:* Public Domain *Contributors:* Mate2code

Image:PeirceAlphaGraphs.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:PeirceAlphaGraphs.svg> *License:* GNU Free Documentation License *Contributors:* Poccil

Image:Laws of Form - cross.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Laws_of_Form_-_cross.gif *License:* GNU Free Documentation License *Contributors:* Sam (talk) (Uploads)

Image:Laws of Form - double cross.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Laws_of_Form_-_double_cross.gif *License:* GNU Free Documentation License *Contributors:* Maksim, 1 anonymous edits

Image:Laws of Form - not a.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Laws_of_Form_-_not_a.gif *License:* GNU Free Documentation License *Contributors:* Maksim, McZusatz, 1 anonymous edits

Image:Laws of Form - a or b.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Laws_of_Form_-_a_or_b.gif *License:* GNU Free Documentation License *Contributors:* Maksim, 1 anonymous edits

Image:Laws of Form - if a then b.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Laws_of_Form_-_if_a_then_b.gif *License:* GNU Free Documentation License *Contributors:* Maksim, 1 anonymous edits

Image:Laws of Form - a and b.gif *Source:* https://en.wikipedia.org/w/index.php?title=File:Laws_of_Form_-_a_and_b.gif *License:* GNU Free Documentation License *Contributors:* Maksim, 1 anonymous edits

file:Commons-logo.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Commons-logo.svg> *License:* logo *Contributors:* Anomie

File:Predicate logic; 2 variables; A1ne.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_A1ne.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki

File:Predicate logic; 2 variables; A2ne.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_A2ne.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki

File:Predicate logic; 2 variables; 12diag_ne.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_12diag_ne.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki, 1 anonymous edits

File:Predicate logic; 2 variables; 12diag_f.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_12diag_f.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki, 1 anonymous edits

File:Predicate logic; 2 variables; 12ne.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_12ne.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki

File:Predicate logic; 2 variables; 12f.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_12f.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki, 1 anonymous edits

File:Predicate logic; 2 variables; implications.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_implications.svg *License:* Public Domain *Contributors:* Jeff G., Mate2code

File:Predicate logic; 2 variables; E2f.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_E2f.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki

File:Predicate logic; 2 variables; E1f.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Predicate_logic;_2_variables;_E1f.svg *License:* unknown *Contributors:* CommonsDelinker, Mate2code, Micki

File:Prop-tableau-4.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Prop-tableau-4.svg> *License:* GNU Free Documentation License *Contributors:* Original uploader was Tizio at en.wikipedia. Later version(s) were uploaded by RobHar at en.wikipedia.

File:Hasse2Free.png *Source:* <https://en.wikipedia.org/w/index.php?title=File:Hasse2Free.png> *License:* Public Domain *Contributors:* CommonsDelinker, Darapti, Juiced lemon, Kilom691, Maksim, Mate2code, Mdd, 2 anonymous edits

File:Free-Boolean-algebra-unit-sloppy.png *Source:* <https://en.wikipedia.org/w/index.php?title=File:Free-Boolean-algebra-unit-sloppy.png> *License:* Public Domain *Contributors:* Daniel Brown

File:Free-Boolean-algebra-unit.png *Source:* <https://en.wikipedia.org/w/index.php?title=File:Free-Boolean-algebra-unit.png> *License:* Public Domain *Contributors:* Daniel Brown

Image:Rieger-Nishimura.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Rieger-Nishimura.svg> *License:* GNU Free Documentation License *Contributors:* EmilJ

File:Skew diag.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Skew_diag.png *License:* Creative Commons Zero *Contributors:* Joaopitacosta

File:Skew pullback.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Skew_pullback.png *License:* Creative Commons Zero *Contributors:* Joaopitacosta

File:Skew lsection.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Skew_lsection.png *License:* Creative Commons Zero *Contributors:* Joaopitacosta

File:Skew primitive.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Skew_primitive.png *License:* Creative Commons Zero *Contributors:* Joaopitacosta

File:Skew diamond.png *Source:* https://en.wikipedia.org/w/index.php?title=File:Skew_diamond.png *License:* Creative Commons Zero *Contributors:* Joaopitacosta

Image:And-inverter-graph.png *Source:* <https://en.wikipedia.org/w/index.php?title=File:And-inverter-graph.png> *License:* GNU Free Documentation License *Contributors:* Florian Pigorsch.

File:Circuit-minimization.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Circuit-minimization.svg> *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* Steaphan Greene (talk)

File:74LS192_Symbol.png *Source:* https://en.wikipedia.org/w/index.php?title=File:74LS192_Symbol.png *License:* Public Domain *Contributors:* Original uploader was Swtpc6800 at en.wikipedia

File:AND ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:AND_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:AND IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:AND_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:OR ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:OR_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:OR IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:OR_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:NOT ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:NOT_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:NOT IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:NOT_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:NAND ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:NAND_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:NAND IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:NAND_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:NOR ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:NOR_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:NOR IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:NOR_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:XOR ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:XOR_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:XOR IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:XOR_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:XNOR ANSI.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:XNOR_ANSI.svg *License:* Public Domain *Contributors:* jjbeard

File:XNOR IEC.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:XNOR_IEC.svg *License:* Public Domain *Contributors:* jjbeard

File:7400.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:7400.jpg> *License:* GNU Free Documentation License *Contributors:* 1-1111, Audriusa, Bomazi, Dicklyon, DL2000, Foroa, InductiveLoad, Pengo

File:Tristate buffer.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Tristate_buffer.svg *License:* Public Domain *Contributors:* Traced by User:Stannered

File:De Morgan Augustus.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:De_Morgan_Augustus.jpg *License:* Public Domain *Contributors:* Sophia Elizabeth De Morgan

File:Augustus De Morgan.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Augustus_De_Morgan.jpg *License:* Public Domain *Contributors:* unknown

File:Charles Sanders Peirce.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Charles_Sanders_Peirce.jpg *License:* Public Domain *Contributors:* Archeologo, FSII, Ineuw, Kalki, Materialscientist, Mu

File:Charles Sanders Peirce's birthplace building.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Charles_Sanders_Peirce's_birthplace_building.jpg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* User:Gruebleen

File:Milford and NYC and Cambridge.GIF *Source:* https://en.wikipedia.org/w/index.php?title=File:Milford_and_NYC_and_Cambridge.GIF *License:* Public Domain *Contributors:* The Tetrastr

File:JulietteAndCharles.JPG *Source:* <https://en.wikipedia.org/w/index.php?title=File:JulietteAndCharles.JPG> *License:* Public Domain *Contributors:* The Tetrastr made a jpg from the gif at an archive.org file of _Spanning the Gap_, newsletter of the Delaware Water Gap National Recreation Area (Vol. 22 No. 3 Fall 2000) of the National Park Service, U.S. Department of the Interior. The photo image is not only in archive.org files of the gov't _Spanning the Gap_ newsletters but also in a pdf maintained online by the National Park Service at <http://www.nps.gov/dewa/historyculture/upload/cmssstgPEIRC.pdf>, see pdf's page 2. The archive.org version (a gif in an html file) was of higher quality. There is no credit for the particular photo image and no copyright notice in either the pdf or in any of the archive.org files (http://web.archive.org/web/*http://www.nps.gov/dewa/InDepth/Spanning/stgPEIRC.html) of the article "Philosopher Charles Peirce" which contains the photo image. The original physical photograph itself must have been made before Charles Peirce's death in 1914.

File:Charles S. Peirce House near Milford PA.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Charles_S._Peirce_House_near_Milford_PA.jpg *License:* Creative Commons Attribution-Share Alike *Contributors:* Beyond My Ken

Image:Peirce-quincuncial-bright-lines.gif *Source:* <https://en.wikipedia.org/w/index.php?title=File:Peirce-quincuncial-bright-lines.gif> *License:* Public Domain *Contributors:* Peirce-quincuncial-projection.jpg: User:Mdf derivative work: The Tetrastr (talk)

File:Peircelines.PNG *Source:* <https://en.wikipedia.org/w/index.php?title=File:Peircelines.PNG> *License:* Public Domain *Contributors:* myself

File:George Boole color.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:George_Boole_color.jpg *License:* Public Domain *Contributors:* Leyo, Wdwd

File:BoolePlaque.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:BoolePlaque.jpg> *License:* Public Domain *Contributors:* Logicus

File:Boole House Cork.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Boole_House_Cork.jpg *License:* Creative Commons Zero *Contributors:* SandStone

File:2010-05-26 at 18-05-02.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:2010-05-26_at_18-05-02.jpg *License:* Creative Commons Attribution 3.0 *Contributors:* Marcovanhogna

File:BooleWindow(bottom third).jpg *Source:* [https://en.wikipedia.org/w/index.php?title=File:BooleWindow\(bottom_third\).jpg](https://en.wikipedia.org/w/index.php?title=File:BooleWindow(bottom_third).jpg) *License:* Public Domain *Contributors:* Logicus

File:BoolePlaque2.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:BoolePlaque2.jpg> *License:* Public Domain *Contributors:* Logicus

File:Wikisource-logo.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Wikisource-logo.svg> *License:* logo *Contributors:* Guillom, INeverCry, Jarekt, Leyo, MichaelMaggs, NielsF, Rei-artur, Rocket000

File:John Venn.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:John_Venn.jpg *License:* Public Domain *Contributors:* User:OrphanBot

Image:Venn John signature.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_John_signature.jpg *License:* Public Domain *Contributors:* uploaded to Wikipedia by Astrochemist

Image:Venn Building, University of Hull.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_Building,_University_of_Hull.jpg *License:* Creative Commons Attribution-ShareAlike 2.0 Generic *Contributors:* Bkell, FrancisTyers, Jarekt, Keith D, Micki, Reedy, Shortfatlad

Image:Venn-stainedglass-gonville-caius.jpg *Source:* <https://en.wikipedia.org/w/index.php?title=File:Venn-stainedglass-gonville-caius.jpg> *License:* Creative Commons Attribution-Sharealike 2.5 *Contributors:* User:Schutz. The stained glass was designed by Maria McClafferty and installed in 1989.

File:Picture of jevons.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Picture_of_jevons.jpg *License:* Public Domain *Contributors:* Clusternote, Julian Felsenburgh

File:Jevon's signature.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Jevon's_signature.jpg *License:* Public Domain *Contributors:* Julian Felsenburgh

File:William Stanley Jevons.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:William_Stanley_Jevons.jpg *License:* Public Domain *Contributors:* Julian Felsenburgh

File:PSM V11 D660 William Stanley Jevons.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:PSM_V11_D660_William_Stanley_Jevons.jpg *License:* Public Domain *Contributors:* Ineuw, Kilom691

File:Jevon's logic piano.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Jevon's_logic_piano.jpg *License:* Public Domain *Contributors:* Clusternote, Julian Felsenburgh

File:Jevons's piano.jpg *Source:* https://en.wikipedia.org/w/index.php?title=File:Jevons's_piano.jpg *License:* Public Domain *Contributors:* Clusternote, Julian Felsenburgh

File:PD-icon.svg *Source:* <https://en.wikipedia.org/w/index.php?title=File:PD-icon.svg> *License:* Public Domain *Contributors:* Alex.muller, Anomie, Anonymous Dissident, CBM, MBisanz, PBS, Quadell, Rocket000, Strangerer, Timotheus Canens, 1 anonymous edits

Image:Graham's Hierarchy of Disagreement1.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Graham's_Hierarchy_of_Disagreement1.svg *License:* Public Domain *Contributors:* Rocket000 (Transferred by Cloudbound/Originally uploaded by Reisio)

File:Proofstrength.png *Source:* <https://en.wikipedia.org/w/index.php?title=File:Proofstrength.png> *License:* Public Domain *Contributors:* Bynne (talk)

Image:Tsitkin frames.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Tsitkin_frames.svg *License:* Creative Commons Attribution-Sharealike 3.0 *Contributors:* EmilJ

File:Commutative Word Origin.PNG *Source:* https://en.wikipedia.org/w/index.php?title=File:Commutative_Word-Origin.PNG *License:* Public Domain *Contributors:* Francois Servois

File:Symmetry Of Addition.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Symmetry_Of_Addition.svg *License:* Public Domain *Contributors:* Weston.pace

File:Venn A subset B.svg *Source:* https://en.wikipedia.org/w/index.php?title=File:Venn_A_subset_B.svg *License:* Public Domain *Contributors:* Booyabazooka, Darapti, Frank C. Müller, Mate2code, Roomba, 1 anonymous edits

License

Creative Commons Attribution-Share Alike 3.0
[//creativecommons.org/licenses/by-sa/3.0/](http://creativecommons.org/licenses/by-sa/3.0/)