Truth table

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

A truth table has one column for each input variable (for example, P and Q), and one final column showing all of the possible results of the logical operation that the table represents (for example, P XOR Q). Each row of the truth table contains one possible configuration of the input variables (for instance, P=true Q=false), and the result of the operation for those values. See the examples below for further clarification. Ludwig Wittgenstein is generally credited with inventing and popularizing the truth table in his *Tractatus Logico-Philosophicus*, which was completed in 1918 and published in 1921.[1] Such a system was also independently proposed in 1921 by Emil Leon Post.[2] An even earlier iteration of the truth table has also been found in unpublished manuscripts by Charles Sanders Peirce from 1893, antedating both publications by nearly 30 years.[3]

Contents

Unary operations

There are 4 unary operations:

- Always true
- Never true, unary *falsum*
- Unary *Identity*
- Unary *negation*

Logical true

The output value is always true, regardless of the input value of p

Logical True

p	*T*
T	T
F	T

Logical false

The output value is never true: that is, always false, regardless of the input value of p

Logical False

p	*F*
T	F
F	F

Logical identity

Logical identity is an operation on one logical value p, for which the output value remains p.

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

Logical Identity

p	*p*
T	T
F	F

Logical negation

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

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

Logical Negation

p	*¬p*
T	F
F	T

Binary operations

There are 16 possible truth functions of two binary variables:

Truth table for all binary logical operators

Here is an extended truth table giving definitions of all possible truth functions of two Boolean variables P and Q:[note 1]

p	*q*		F[0]	NOR[1]	←[2]	¬p[3]	→[4]	¬q[5]	XOR[6]	NAND[7]		AND[8]	XNOR[9]	q[10]	→[11]	p[12]	←[13]	OR[14]	T[15]
T	**T**		F	F	F	F	F	F	F	F		T	T	T	T	T	T	T	T
T	**F**		F	F	F	F	T	T	T	T		F	F	F	F	T	T	T	T
F	**T**		F	F	T	T	F	F	T	T		F	F	T	T	F	F	T	T
F	**F**		F	T	F	T	F	T	F	T		F	T	F	T	F	T	F	T
Com			✓	✓					✓	✓		✓	✓					✓	✓
L id					F				F			T	T	T,F	T			F	
R id							F		F			T	T			T,F	T	F	

where

T = true.
F = false.
The **Com** row indicates whether an operator, **op**, is commutative - **P op Q = Q op P**.
The **L id** row shows the operator's left identities if it has any - values **I** such that **I op Q = Q**.
The **R id** row shows the operator's right identities if it has any - values **I** such that **P op I = P**.[note 2]

The four combinations of input values for p, q, are read by row from the table above. The output function for each p, q combination, can be read, by row, from the table.

Key:

The following table is oriented by column, rather than by row. There are four columns rather than four rows, to display the four combinations of p, q, as input.

p: T T F F
q: T F T F

There are 16 rows in this key, one row for each binary function of the two binary variables, p, q. For example, in row 2 of this Key, the value of Converse nonimplication ('↚') is solely T, for the column denoted by the unique combination p=F, q=T; while in row 2, the value of that '↚' operation is F for the three remaining columns of p, q. The output row for ↚ is thus

2: F F T F

and the 16-row[4] key is

	[4]		operator	Operation name
0	(F F F F)(p, q)	⊥	false, **Opq**	Contradiction
1	(F F F T)(p, q)	NOR	**p** ↓ **q**, **Xpq**	Logical NOR
2	(F F T F)(p, q)	←/	**p** ←/ **q**, **Mpq**	Converse nonimplication
3	(F F T T)(p, q)	**¬p**, **~p**	**¬p**, **Np**, **Fpq**	Negation
4	(F T F F)(p, q)	→/	**p** →/ **q**, **Lpq**	Material nonimplication
5	(F T F T)(p, q)	**¬q**, **~q**	**¬q**, **Nq**, **Gpq**	Negation
6	(F T T F)(p, q)	XOR	**p** ⊕ **q**, **Jpq**	Exclusive disjunction
7	(F T T T)(p, q)	NAND	**p** ↑ **q**, **Dpq**	Logical NAND
8	(T F F F)(p, q)	AND	**p** ∧ **q**, **Kpq**	Logical conjunction
9	(T F F T)(p, q)	XNOR	**p** If and only if **q**, **Epq**	Logical biconditional
10	(T F T F)(p, q)	**q**	**q**, **Hpq**	Projection function
11	(T F T T)(p, q)	**p** → **q**	if **p** then **q**, **Cpq**	Material implication
12	(T T F F)(p, q)	**p**	**p**, **Ipq**	Projection function
13	(T T F T)(p, q)	**p** ← **q**	**p** if **q**, **Bpq**	Converse implication
14	(T T T F)(p, q)	OR	**p** ∨ **q**, **Apq**	Logical disjunction
15	(T T T T)(p, q)	⊤	true, **Vpq**	Tautology

Logical operators can also be visualized using Venn diagrams.

Logical conjunction (AND)

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

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

Logical conjunction

p	***q***	***p ∧ q***
T	T	T
T	F	F
F	T	F
F	F	F

In ordinary language terms, if both *p* and *q* are true, then the conjunction $p \and q$ is true. For all other assignments of logical values to *p* and to *q* the conjunction $p \and q$ is false.

It can also be said that if *p*, then $p \and q$ is *q*, otherwise $p \and q$ is *p*.

Logical disjunction (OR)

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 $\mathbf{p \or q}$, **Apq**, **p || q**, or **p + q**) is as follows:

Logical disjunction

p	*q*	$p \or q$
T	T	T
T	F	T
F	T	T
F	F	F

Stated in English, if *p*, then $p \or q$ is *p*, otherwise $p \or q$ 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, which produces a value of *false* if the first operand is true and the second operand is false, and a value of *true* otherwise.

The truth table associated with the logical implication **p implies q** (symbolized as $\mathbf{p \Rightarrow q}$, or more rarely **Cpq**) is as follows:

Logical implication

p	*q*	$p \Rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

The truth table associated with the material conditional **if p then q** (symbolized as $\mathbf{p \to q}$) is as follows:

Material conditional

p	*q*	$p \to q$
T	T	T
T	F	F
F	T	T
F	F	T

It may also be useful to note that $\mathbf{p \Rightarrow q}$ and $\mathbf{p \to q}$ are equivalent to $\mathbf{\neg p \or q}$.

Logical equality

Logical equality (also known as biconditional) is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if both operands are false or both operands are true.

The truth table for **p XNOR q** (also written as $\mathbf{p \leftrightarrow q}$, **Epq**, **p = q**, or $\mathbf{p \equiv q}$) is as follows:

Logical equality

p	*q*	*p* ↔ *q*
T	T	T
T	F	F
F	T	F
F	F	T

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

Exclusive disjunction

Exclusive disjunction is an operation on two logical values, typically the values of two propositions, that produces a value of *true* if one but not both of its operands is true.

The truth table for **p XOR q** (also written as **p ⊕ q**) is as follows:

Exclusive disjunction

p	*q*	¬*p*v¬*q*
T	T	F
T	F	T
F	T	T
F	F	F

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

Logical NAND

The logical NAND is an operation on two logical values, typically the values of two propositions, that produces a value of *false* if both of its operands are true. In other words, it produces a value of *true* if at least one of its operands is false.

The truth table for **p NAND q** (also written as **p ↑ q**, **Dpq**, or **p | q**) is as follows:

Logical NAND

p	*q*	*p* ↑ *q*
T	T	F
T	F	T
F	T	T
F	F	T

It is frequently useful to express a logical operation as a compound operation, that is, as an operation that is built up or composed from other operations. Many such compositions are possible, depending on the operations that are taken as basic or "primitive" and the operations that are taken as composite or "derivative".

In the case of logical NAND, it is clearly expressible as a compound of NOT and AND.

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

p	*q*	*p* ∧ *q*	¬(*p* ∧ *q*)	¬*p*	¬*q*	(¬*p*) ∨ (¬*q*)
T	T	T	F	F	F	F
T	F	F	T	F	T	T
F	T	F	T	T	F	T
F	F	F	T	T	T	T

Logical NOR

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

The truth table for **p NOR q** (also written as **p ↓ q**, or **Xpq**) 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 $\neg(p \lor q)$, and the conjunction of negations $(\neg p) \land (\neg q)$ can be tabulated as follows:

p	q	$p \lor q$	$\neg(p \lor q)$	$\neg p$	$\neg q$	$(\neg p) \land (\neg q)$
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 $\neg(p \land q)$ as for $(\neg p) \lor (\neg q)$, and for $\neg(p \lor q)$ as for $(\neg p) \land (\neg q)$. 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:

Logical equivalence : $(p \Rightarrow q) \equiv (\neg p \lor q)$

p	q	$\neg p$	$\neg p \lor q$	$p \Rightarrow q$
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 $\neg p \lor q$.

Truth table for most commonly used logical operators

Here is a truth table that gives definitions of the 6 most commonly used out of the 16 possible truth functions of two Boolean variables P and Q:

P	Q	$P \land Q$	$P \lor Q$	$P \veebar Q$	$P \barwedge Q$	$P \Rightarrow Q$	$P \Leftarrow Q$	$P \Leftrightarrow Q$
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

where

T
: true

F
: false

$\land$
: AND (logical conjunction)

$\lor$
: OR (logical disjunction)

$\veebar$
: XOR (exclusive or)

$\barwedge$

XNOR (exclusive nor)

⇒

conditional "if-then"

⇐

conditional "then-if"

⇔

biconditional "if-and-only-if".

Condensed truth tables for binary operators

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

∧	F	T
F	F	F
T	F	T

∨	F	T
F	F	T
T	T	T

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

Truth tables in digital logic

Truth tables are also used to specify the function 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 $k = V_0 \times 2^0 + V_1 \times 2^1 + V_2 \times 2^2 + \cdots + V_n \times 2^n$.

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

Applications of truth tables in digital electronics

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

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

where

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

This truth table is read left to right:

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

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

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

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

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

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

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

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

History

Irving Anellis's research shows that C.S. Peirce appears to be the earliest logician (in 1893) to devise a truth table matrix.[3][5] 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.

Notes

1. Information about notation may be found in Bocheński (1959), Enderton (2001), and Quine (1982).
2. The operators here with equal left and right identities (XOR, AND, XNOR, and OR) are also commutative monoids because they are also associative. While this distinction may be irrelevant in a simple discussion of logic, it can be quite important in more advanced mathematics. For example, in category theory an enriched category is described as a base category enriched over a monoid, and any of these operators can be used for enrichment.

See also

- Boolean domain
- Boolean-valued function
- Espresso heuristic logic minimizer
- Excitation table
- First-order logic
- Functional completeness
- Karnaugh maps
- Logic gate
- Logical connective
- Logical graph
- Method of analytic tableaux
- Propositional calculus
- Truth function

References

1. Georg Henrik von Wright (1955). "Ludwig Wittgenstein, A Biographical Sketch". *The Philosophical Review*. **64** (4): 527–545 (p. 532, note 9). doi:10.2307/2182631 (https://doi.org/10.2307%2F2182631). JSTOR 2182631 (https://www.jstor.org/stable/2182631).
2. Emil Post (July 1921). "Introduction to a general theory of elementary propositions". *American Journal of Mathematics*. **43** (3): 163–185. doi:10.2307/2370324 (https://doi.org/10.2307%2F2370324). JSTOR 2370324 (https://www.jstor.org/stable/2370324).
3. Anellis, Irving H. (2012). "Peirce's Truth-functional Analysis and the Origin of the Truth Table". *History and Philosophy of Logic*. **33**: 87–97. doi:10.1080/01445340.2011.621702 (https://doi.org/10.1080%2F01445340.2011.621702).
4. Ludwig Wittgenstein (1922) *Tractatus Logico-Philosophicus* Proposition 5.101 (http://www.gutenberg.org/files/5740/5740-pdf.pdf)
5. Peirce's publication included the work of Christine Ladd (1881): Peirce's Ph.D. student Christine Ladd-Franklin found the truth table in *Tractatus Logico-Philosophicus* Proposition 5.101, 40 years earlier than Wittgenstein. Christine Ladd (1881), "On the Algebra of Logic", p.62 (https://books.google.com/books?id=A48XAAAAIAAJ&pg=PA62), *Studies in Logic*, C. S. Peirce ed., 1883

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) [1994], "Truth table" (https://www.encyclopediaofmath.org/index.php?title=p/t094370), *Encyclopedia of Mathematics*, Springer Science+Business Media B.V. / Kluwer Academic Publishers, ISBN 978-1-55608-010-4
- Truth Tables, Tautologies, and Logical Equivalence (http://sites.millersville.edu/bikenaga/math-proof/truth-tables/truth-tables.html)
- PEIRCE'S TRUTH-FUNCTIONAL ANALYSIS AND THE ORIGIN OF TRUTH TABLES (https://arxiv.org/ftp/arxiv/papers/1108/1108.2429.pdf) by Irving H. Anellis
- Converting truth tables into Boolean expressions (http://www.allaboutcircuits.com/vol_4/chpt_7/9.html)