Negation

For other uses, see Negation (disambiguation) and NOT gate.

In logic, **negation**, also called **logical complement**, is an operation that takes a proposition *p* to another proposition "not *p*", written ¬*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

No agreement exists as to the possibility of defining negation, as to its logical status, function, and meaning, as to its field of applicability..., and as to the interpretation of the negative judgment, (F.H. Heinemann 1944).[1]

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 ¬*A* (pronounced "not A") would therefore be false; and conversely, if ¬*A* is false, then *A* would be true.

The truth table of ¬*p* is as follows:

Truth table of ¬p

p	¬p
True	False
False	True

Classical negation can be defined in terms of other logical operations. For example, ¬*p* can be defined as *p* → *F*, where "→" is logical consequence and *F* is absolute falsehood. Conversely, one can define *F* as *p* & ¬*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 paraconsistent logic, where contradictions are not necessarily false. But in classical logic, we get a further identity: *p* → *q* can be defined as ¬*p* ∨ *q*, where "∨" 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
~*p*	Not *p*
−*p*	Not *p*
Np	En *p*
¬*p*	Not *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 \ 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 $\neg p$ / $-p$ 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 :

$$\neg(a \lor b) \equiv (\neg a \land \neg b), \text{ and}$$
$$\neg(a \land b) \equiv (\neg a \lor \neg b).$$

Linearity

In Boolean algebra, a linear function is one such that:

If there exists $a_0, a_1, ..., a_n \in \{0,1\}$ such that $f(b_1, ..., b_n) = a_0 \oplus (a_1 \land b_1) \oplus ... \oplus (a_n \land b_n)$, for all $b_1, ..., b_n \in \{0,1\}$.

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:

$f(a_1, ..., a_n) = \sim f(\sim a_1, ..., \sim a_n)$ for all $a_1, ..., a_n \in \{0,1\}$. 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 ⊥. 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 $p \rightarrow \bot$. 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 `¬` for negation. Some modern computers and operating systems will display `¬` 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 "`~`" 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. (See also possible world semantics.)

See also

- Logical conjunction
- Logical disjunction
- NOT gate
- Bitwise NOT

- Ampheck
- Apophasis
- Cyclic negation
- Double negative elimination
- Grammatical polarity
- Negation (linguistics)
- Negation as failure
- Square of opposition
- Binary opposition

References

1. ↑ Horn, Laurence R (2001). "Chapter 1". *A NATURAL HISTORY OF NEGATION* (PDF). Stanford University: CLSI Publications. p. 1. ISBN 1-57586-336-7. Retrieved 29 Dec 2013.

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.
- Tettamanti, Marco; Manenti, Rosa; Della Rosa, Pasquale A.; Falini, Andrea; Perani, Daniela; Cappa, Stefano F.; Moro, Andrea (2008). "Negation in the brain: Modulating action representation". *NeuroImage*. **43** (2): 358–367. doi:10.1016/j.neuroimage.2008.08.004.

External links

- Horn, Laurence R.; Wansing, Heinrich. "Negation". *Stanford Encyclopedia of Philosophy*.
- Hazewinkel, Michiel, ed. (2001), "Negation", *Encyclopedia of Mathematics*, Springer, ISBN 978-1-55608-010-4
- NOT, on MathWorld

Logical connectives

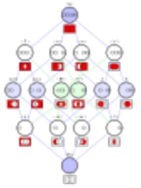
Tautology ⊤

NAND ↑ · Converse implication ← · Implication → · OR ∨

Negation ¬ · XOR ⊕ · Biconditional ↔ · Statement

NOR ↓ · Nonimplication ↛ · Converse nonimplication ↚ · AND ∧

Contradiction ⊥



Common logical symbols

∧ *or* &	∨	¬ *or* ~	→	⊃	≡	|	∀	∃	⊤	⊥	⊢	⊨	∴	∵
and	or	not	implies	implies, superset	iff	nand	universal quantification	existential quantification	tautology	false, contradiction	entails, proves	entails, therefore	therefore	because