Conjunctive normal form

From Wikipedia, the free encyclopedia

In boolean logic, a formula is in **conjunctive normal form** (CNF or cnf) if it is a conjunction of clauses, where a clause is a disjunction of literals. As a normal form, it is useful in automated theorem proving. It is similar to the canonical 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 disjunctions 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.

For example, all of the following formulas are in CNF:

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

But the following are not:

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

The above three formulas are respectively equivalent to the following three formulas that are in conjunctive normal form:

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

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, the De Morgan's laws, and the distributive law.

Since all logical formulas 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 in CNF produces a formula with 2^n clauses:

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

Transformations of formulae in CNF preserving satisfiability rather than equivalence and not producing an exponential increase of size exist. These transformations are guaranteed to only linearly increase the size of

the formula, but introduce new variables.

Conjunctive normal form can be taken further to yield the clausal normal form of a logical formula, that is used to perform first-order resolution.

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 such that 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 solution in polynomial time.

Converting from first-order logic

To convert first-order logic to CNF:

1. Eliminate implications: convert $x \Rightarrow y$ to $\neg x \vee y$
2. Move NOTs inwards
3. Standardize variables
4. Skolemize the statement
5. Drop universal quantifiers
6. Distribute ORs over ANDs.

(Artificial Intelligence: A modern Approach [1995...] Russel and Norvig)

See also

- Disjunctive normal form
- Algebraic normal form
- Horn clause
- Quine-McCluskey algorithm

External links

- Scheme and Ocaml programs for converting propositional logic statements into CNF (http://www.g615.co.uk/riftor/content.php?view=5&type=1)

Retrieved from "http://en.wikipedia.org/wiki/Conjunctive_normal_form"

Categories: Logic stubs | Logic | Boolean algebra
