CS262 Logic and Verification

Lecture 4: Normal forms

A set of connectives is said to be **complete** if we can represent every truth function $\{T, F\}^n \to \{T, F\}$ using only these connectives (there are 2^{2^n} such functions).

Example: Is $\{\neg, \wedge, \vee\}$ complete?

First look at all unary functions, i.e. the case $n = 1$:

p	f_1	f_2	f_3	f_4
T	T	T	F	F
F	T	F	T	F

This is not too difficult:

$f_1 = \top$
$f_2 = p$
$f_3 = \neg p$
$f_4 = \bot$

What about binary functions?

For $n = 2$ there are 16 possible functions:

p	q	f_1	f_2	f_3	f_4	f_5	f_6	f_7	$\dots$	f_{16}
T	T	T	T	T	T	F	T	T	$\dots$	F
T	F	T	T	T	F	T	T	F	$\dots$	F
F	T	T	T	F	T	T	F	F	$\dots$	F
F	F	T	F	T	T	T	F	T	$\dots$	F

How about the following:

$f_1 = \top$

$f_2 = p \lor q$

$f_3 = p \lor \neg q$

$\dots$

But there is still plenty to do and then there are the cases $n = 3$, $n = 4$, etc.

Proof by construction

To create a formula that is logically equivalent to any function f on variables $x_1, \dots x_n$ given by its truth table, do the following:

- for every valuation which maps to T construct the **conjunction** $L_1 \wedge \dots \wedge L_n$ where L_j is x_j if x_j is assigned T under the valuation and L_j is $\neg x_j$ if x_j is assigned F under the valuation
- take the **disjunction** of all conjunctions from the previous step
- for the function that is everywhere F, by convention take $\perp$

This shows that $\{\neg, \wedge, \vee\}$ is indeed complete.

The resulting formula is called a **disjunctive normal form (DNF)** of f.

Example

p	q	r	f
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	F
F	T	F	F
F	F	T	F
F	F	F	T

$$\begin{aligned} f(p,q,r) &= (p \wedge q \wedge r) \vee (p \wedge q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge \neg r) \\ &= (p \wedge q) \vee (\neg p \wedge \neg q \wedge \neg r) \end{aligned}$$

The disjunctive normal form of a function f is **not unique**.

Conjunctive normal form

A formula in **disjunctive normal form (DNF)** is a disjunction of conjunctions of literals. A **literal** is a variable or its negation, or $\bot$ or $\top$.

A formula in **conjunctive normal form (CNF)** is a conjunction of disjunctions of literals.

Examples:

DNF: $(p \wedge \neg q) \vee (\neg p \wedge q \wedge \neg r)$
CNF: $(\neg p \vee \neg q \vee r) \wedge (\neg p \vee q \vee \neg r)$
(these are two different functions)

Theorem: Every Boolean function has a DNF.

Theorem: Every Boolean function has a CNF.

First theorem follows from our construction before. Proof of second theorem in the exercises.

Further questions (see the exercises)

What other sets of connectives are complete?

How can you tell/prove that a set of connectives is not complete?

What is the minimum number of connectives needed?

Normal form algorithms

Problem: Given a formula, can we derive its DNF or CNF in a systematic way, other than by writing down the entire truth table?

This can be achieved by normal form algorithms.

The input is any propositional formula, the output is a semantically equivalent formula in DNF or CNF.

In every step, the algorithm applies a single rewriting rule given by one of the laws of Boolean algebra.