Lecture 3
Equivalences and normal forms
Boolean algebras, equational reasoning, normal forms

Print version of the lecture in *Logic and Proof*

presented on 3 May 2019

by Dr Christoph Haase

3.1

One of the main topics studied in computational logic are decision problems. A decision problem is a computational problem whose output is either "yes" or "no". The decision problems most relevant to us are the following:

- **Satisfiability**: Given a formula F, is F satisfiable?
- **Validity**: Given a formula F, is F valid?
- **Entailment**: Given formulas F and G, does $F \models G$ hold?
- **Equivalence**: Given formulas F and G, does $F \equiv G$ hold?

In this lecture, we will focus on a method to *decide* the last problem. A "brute-force" method to show that two formulas are logically equivalent is to use truth tables. Instead we introduce an alternative approach that is more practical in many cases, namely *equational reasoning*. The idea is to start from some basic equivalences (the Boolean algebra axioms) and derive new equivalences using the closure of logical equivalence under substitution.

1 Boolean algebras

The following is a list of the Boolean algebra axioms:

$$
\begin{aligned}
F \wedge F &\equiv F \\
F \vee F &\equiv F && \textbf{Idempotence} \\
F \wedge G &\equiv G \wedge F \\
F \vee G &\equiv G \vee F && \textbf{Commutativity} \\
(F \wedge G) \wedge H &\equiv F \wedge (G \wedge H) \\
(F \vee G) \vee H &\equiv F \vee (G \vee H) && \textbf{Associativity} \\
F \wedge (F \vee G) &\equiv F \\
F \vee (F \wedge G) &\equiv F && \textbf{Absorption} \\
F \wedge (G \vee H) &\equiv (F \wedge G) \vee (F \wedge H) \\
F \vee (G \wedge H) &\equiv (F \vee G) \wedge (F \vee H) && \textbf{Distributivity} \\
\neg\neg F &\equiv F && \textbf{Double negation} \\
\neg(F \wedge G) &\equiv (\neg F \vee \neg G) \\
\neg(F \vee G) &\equiv (\neg F \wedge \neg G) && \textbf{De Morgan's laws} \\
F \vee \neg F &\equiv \mathit{true}
\end{aligned}
$$

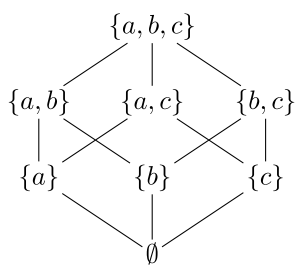
$$F \wedge \neg F \equiv \mathit{false} \qquad \textbf{Complementation}$$
$$F \vee \mathit{true} \equiv \mathit{true}$$
$$F \wedge \mathit{false} \equiv \mathit{false} \qquad \textbf{Zero Laws}$$
$$F \vee \mathit{false} \equiv F$$
$$F \wedge \mathit{true} \equiv F \qquad \textbf{Identity Laws}$$

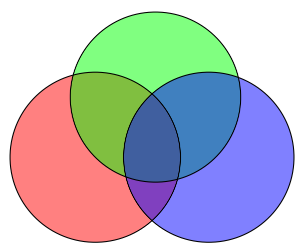
Using truth tables, it is possible to show that those axioms hold for all formulas F, G and H. Notice that the Boolean algebra axioms come in pairs: the equivalences in each pair are dual to each other in the sense that one is obtained from the other by interchanging $\vee$ and $\wedge$ and interchanging true and false.

Exercise 1. Given a formula F, define the **De Morgan dual** $\overline{F}$ by induction as follows. The base cases are that F has the form true or false, or has the form x or $\neg x$ for a propositional variable x. Here we define $\overline{\mathit{true}} := \mathit{false}$, $\overline{\mathit{false}} := \mathit{true}$, $\overline{x} := \neg x$, and $\overline{\neg x} := x$. Furthermore, for formulas F and G we define $\overline{G \vee H} := \overline{G} \wedge \overline{H}$, $\overline{G \wedge H} := \overline{G} \vee \overline{H}$, and $\overline{\neg G} = \neg \overline{G}$ if G is not a propositional variable. Show that $\overline{F} \equiv \neg F$.

A **Boolean algebra** is a set A together with two elements $\mathit{true}, \mathit{false} \in A$, one unary operation $\neg \colon A \to A$, and two binary operations $\wedge, \vee \colon A \times A \to A$ satisfying the Boolean algebra axioms. Here are two examples of Boolean algebras:

- $A = \{0, 1\}$ (that is the one we study in this course)
- For any set X, take $A = 2^X$ with $\mathit{true} = X$, $\mathit{false} = \emptyset$, $\wedge = \cap$, $\vee = \cup$, $\neg S = X \setminus S$. In fact, any finite Boolean algebra is of the form 2^X.





2 Equational reasoning

Equational reasoning is about transforming a formula into a sequence of equivalent formulas using the Boolean algebra axioms until a desired equivalent target formula is obtained. This captures the intuition of what a proof should be. The essence of equational reasoning is the substitution of equals for equals. To formalise this we first give a precise definition of substitution. Subsequently, we use the symbol $=$ to denote *syntactic equality*, i.e., $F = G$ means that F and G are the same formula.

Given a formula F and a formula H we define a new formula $G[F/H]$ (read "G with F **substituted** for all occurrences of H") by induction on the structure of G as follows:

- Informally, $G[F/H]$ means *"substitute F for H in G"*. E.g.:

$$(p_1 \wedge (p_2 \vee p_1))[\neg q_1/p_1] = \neg q_1 \wedge (p_2 \vee \neg q_1)$$

- Formally, $G[F/H] := F$ if $G = H$. Whenever $G \neq H$, we proceed by induction:
  - Base cases:

$$x[F/H] := x \qquad \text{for all } x \in X$$

– Induction steps:

$$\begin{aligned}(\neg G)[F/H] &:= \neg(G[F/H])\\ (G_1 \wedge G_2)[F/H] &:= G_1[F/H] \wedge G_2[F/H]\\ (G_1 \vee G_2)[F/H] &:= G_1[F/H] \vee G_2[F/H].\end{aligned}$$

The following theorem is central to equational reasoning. It proves what should intuitively be clear: whenever we substitute a subformula of a formula G by an equivalent one, the resulting formula is equivalent to G.

Theorem 2 (Substitution Theorem)**.** *Let F, G, G', H be formulas such that $G' = G[F/H]$ and $F \equiv H$. Then $G' \equiv G$.*

Proof. If $G = H$ then $G[F/H] = F$, and thus $G' = F \equiv H = G$. Hence it remains to show the statement for $G \neq H$. We proceed by induction on the structure of G. For the induction base case, let $G = x$. Since $H \neq G$, we have $G' = x$, and hence $G' \equiv G$.

For the induction step, let $G = \neg J$ and $G' = G[F/H] = \neg(J[F/H])$. Let $J' = J[F/H]$, by the induction hypothesis we have $J' \equiv J$, and consequently $G' = \neg J' \equiv \neg J = G$. For $G = G_1 \wedge G_2$, by the induction hypothesis, $G_1 \equiv G_1[F/H] = G_1'$ and $G_2 \equiv G_2[F/H] = G_2'$. Hence $G_1 \wedge G_2 \equiv G_1' \wedge G_2'$, and consequently $G' \equiv G$. The case $G = G_1 \vee G_2$ follows analogously. □

We can now apply Theorem 2 in order to perform equational reasoning. Here is an example:

$$(P \vee (Q \vee R) \wedge (R \vee \neg P)) \equiv R \vee (\neg P \wedge Q)\,.$$

has the following equational proof:

$$\begin{aligned}(P \vee (Q \vee R)) \wedge (R \vee \neg P) &\equiv ((P \vee Q) \vee R) \wedge (R \vee \neg P)\\ &\equiv (R \vee (P \vee Q)) \wedge (R \vee \neg P)\\ &\equiv R \vee ((P \vee Q) \wedge \neg P)\\ &\equiv R \vee (\neg P \wedge (P \vee Q))\\ &\equiv R \vee ((\neg P \wedge P) \vee (\neg P \wedge Q))\\ &\equiv R \vee (\mathit{false} \vee (\neg P \wedge Q))\\ &\equiv R \vee (\neg P \wedge Q).\end{aligned}$$

3 Normal forms

For algorithms reasoning about Boolean formulas, it is convenient to assume that formulas are presented in a unified form into which any arbitrary formula can be transformed to. Two of the most prominent normal forms are defined as follows:

- A **literal** is a propositional variable or the negation of a propositional variable:

$$x \text{ or } \neg x$$

- A formula F is in **conjunctive normal form (CNF)** if it is a conjunction of disjunctions of literals $L_{i,j}$:

$$F = \bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m_i} L_{i,j})$$

- A formula F is in **disjunctive normal form (DNF)** if it is a disjunction of conjunctions of literals $L_{i,j}$:

$$F = \bigvee_{i=1}^{n} (\bigwedge_{j=1}^{m_i} L_{i,j})$$

- Convention: $true$ is CNF with no clauses, $false$ is CNF with a single clause without literals

Example 3. The formulas representing the 3-colouring problem and the Sudoku problem in the previous lecture are both CNF formulas.

The following theorem formally show that any formula can be transformed into CNF and DNF, making them universally applicable normal forms.

Theorem 4 (Normalisation Theorem)**.** *For every formula there is an equivalent formula in CNF and an equivalent formula in DNF.*

Proof. We can transform a formula F into an equivalent CNF formula using equational reasoning as follows:

1. Using the Double Negation law and De Morgan's laws, substitute in F every occurrence of a subformula of the form

$$\begin{array}{rcl} \neg\neg G & \text{by} & G \\ \neg(G \land H) & \text{by} & (\neg G \lor \neg H) \\ \neg(G \lor H) & \text{by} & (\neg G \land \neg H) \\ \neg true & \text{by} & false \\ \neg false & \text{by} & true \end{array}$$

   until no such formulas occur (i.e., push all negations inward until negation is only applied to propositional variables).
2. Using the Distributivity laws, substitute in F every occurrence of a subformula of the form

$$\begin{array}{rcl} G \lor (H \land R) & \text{by} & (G \lor H) \land (G \lor R) \\ (H \land R) \lor G & \text{by} & (H \lor G) \land (R \lor G) \\ G \lor true & \text{by} & true \\ true \lor G & \text{by} & true \end{array}$$

   until no such formulas occur (i.e., push all disjunctions inward until no conjunction occurs under a disjunction).
3. Use the Identity and Zero laws to remove $false$ from any clause and to delete all clauses containing $true$.

The resulting formula is then in CNF.

The translation of F to DNF has the same first step, but *dualises* steps 2 and 3 (swap $\land$ and $\lor$, and swap *true* and *false*).

□

In summary, we see that CNF formulas and DNF formulas both have the same expressiveness as the class of all formulas. However you will see in Exercise Sheet 1 that they differ in succinctness: a CNF can be exponentially shorter than the corresponding DNF and *vice versa*. Note in relation to this that the SAT problem is trivial for DNF formulas. On the other hand, we will see later on that SAT for general formulas is easily reduced to SAT for CNF formulas.