Logic and Proof Hilary 2016

Equivalence and Normal Forms

James Worrell

1 Equational Reasoning

A "brute-force" method to show that two formulas are logically equivalent is to use truth tables. In this lecture 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.

1.1 Boolean Algebra Axioms

Proposition 1. The following equivalences hold for all formulas F, G and H:

$$F \wedge F \equiv F$$
$$F \vee F \equiv F \quad \text{(Idempotence)}$$

$$F \wedge G \equiv G \wedge F$$
$$F \vee G \equiv G \vee F \quad \text{(Commutativity)}$$

$$(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$$
$$(F \vee G) \vee H \equiv F \vee (G \vee H) \quad \text{(Associativity)}$$

$$F \wedge (F \vee G) \equiv F$$
$$F \vee (F \wedge G) \equiv F \quad \text{(Absorption)}$$

$$F \wedge (G \vee H) \equiv (F \wedge G) \vee (F \wedge H)$$
$$F \vee (G \wedge H) \equiv (F \vee G) \wedge (F \vee H) \quad \text{(Distributivity)}$$

$$\neg\neg F \equiv F \quad \text{(Double negation)}$$

$$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$$
$$\neg(F \vee G) \equiv (\neg F \wedge \neg G) \quad \text{(De Morgan's Laws)}$$

$$F \vee \neg F \equiv \mathbf{true}$$
$$F \wedge \neg F \equiv \mathbf{false} \quad \text{(Complementation)}$$

$$F \vee \mathbf{true} \equiv \mathbf{true}$$
$$F \wedge \mathbf{false} \equiv \mathbf{false} \quad \text{(Zero Laws)}$$

$$F \vee \mathbf{false} \equiv F$$
$$F \wedge \mathbf{true} \equiv F \quad \text{(Identity Laws)}$$

Proof. These can all be shown using truth tables. □

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 by interchanging $\vee$ and $\wedge$ and interchanging **true** and **false**.

Exercise 2. 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 P or $\neg P$ for a propositional

variable P. Here we define $\overline{\mathbf{true}} := \mathbf{false}$, $\overline{\mathbf{false}} := \mathbf{true}$, $\overline{P} := \neg P$, and $\overline{\neg P} := P$. 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 $\overline{F} \equiv \neg F$.

1.2 Substitution

The essence of equational reasoning is the substitution of equals for equals. To formalise this we first give a precise definition of substitution.

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 propositional variable P we define a new formula $G[F/P]$ (read "G with F substituted for all occurrences of P") by induction on the structure of G as follows. For a propositional variable Q we have

$$Q[F/P] := \begin{cases} F & \text{if } Q = P \\ Q & \text{if } Q \neq P \end{cases}$$

Otherwise we inductively define

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

For example, $(p_1 \wedge (p_2 \vee p_1))[\neg q_1/p_1] = \neg q_1 \wedge (p_2 \vee \neg q_1)$. Note that all occurrences of p_1 are substituted.

The above definition was purely about syntax. Next we define an *update* operation on assignments that can be seen as a semantic counterpart of substitution.

Given an assignment $\mathcal{A}$, propositional variable P, and truth value $b \in \{0, 1\}$, define the assignment $\mathcal{A}_{[P \mapsto b]}$ by

$$\mathcal{A}_{[P \mapsto b]}[\![Q]\!] = \begin{cases} b & \text{if } Q = P \\ \mathcal{A}[\![Q]\!] & \text{if } Q \neq P \end{cases}$$

for each propositional variable Q. In other words, this assignment behaves like $\mathcal{A}$ except that variable P gets mapped to b.

Substitution and update are related in the following lemma.

Lemma 3 (Translation Lemma)**.** Given formulas F, G, propositional variable P, and assignment $\mathcal{A}$, we have $\mathcal{A}[\![G[F/P]]\!] = \mathcal{A}_{[P \mapsto \mathcal{A}[\![F]\!]]}[\![G]\!]$.

Proof. The proof is by induction on G, exploiting the inductive definition of substitution.

The first base case is that $G = P$. Then we have

$$\mathcal{A}[\![P[F/P]]\!] = \mathcal{A}[\![F]\!] = \mathcal{A}_{[P \mapsto \mathcal{A}[\![F]\!]]}[\![P]\!]\,.$$

The second base case is that $G = Q$ for some propositional variable Q different from P. Then

$$\mathcal{A}[\![Q[F/P]]\!] = \mathcal{A}[\![Q]\!] = \mathcal{A}_{[P \mapsto \mathcal{A}[\![F]\!]]}[\![Q]\!]\,.$$

The induction case for conjunction is as follows. If $G = G_1 \wedge G_2$ then

$$\begin{aligned}
\mathcal{A} \models (G_1 \wedge G_2)[F/P] \text{ iff } & \mathcal{A} \models G_1[F/P] \wedge G_2[F/P] \\
\text{iff } & \mathcal{A} \models G_1[F/P] \text{ and } \mathcal{A} \models G_2[F/P] \\
\text{iff } & \mathcal{A}_{[P \mapsto \mathcal{A}[\![F]\!]]} \models G_1 \text{ and } \mathcal{A}_{[P \mapsto \mathcal{A}[\![F]\!]]} \models G_2 \qquad \text{induction hypothesis} \\
\text{iff } & \mathcal{A}_{[P \mapsto \mathcal{A}[\![F]\!]]} \models G_1 \wedge G_2
\end{aligned}$$

The induction cases for disjunction and negation are similar and are omitted. □

Theorem 4 (Substitution Theorem)**.** Let F_1, F_2, G_1, G_2 be formulas, and P a propositional variable. If $F_1 \equiv F_2$ and $G_1 \equiv G_2$, then $G_1[F_1/P] \equiv G_2[F_2/P]$.

Proof. The proof is a direct application of Lemma 3.

$$\begin{aligned}
\mathcal{A}[\![G_1[F_1/P]]\!] &= \mathcal{A}_{[P \mapsto \mathcal{A}[\![F_1]\!]]}[\![G_1]\!] && \text{by Lemma 3} \\
&= \mathcal{A}_{[P \mapsto \mathcal{A}[\![F_1]\!]]}[\![G_2]\!] && \text{since } G_1 \equiv G_2 \\
&= \mathcal{A}_{[P \mapsto \mathcal{A}[\![F_2]\!]]}[\![G_2]\!] && \text{since } F_1 \equiv F_2 \\
&= \mathcal{A}[\![G_2[F_2/P]]\!] && \text{by Lemma 3}
\end{aligned}$$

□

Typically one applies the Substitution Theorem in the special case that that $G_1 = G_2$ and there is a single occurrence of P in $G_1 = G_2$. In this case one starts with a formula G and obtains an equivalent formula by replacing one occurrence of a subformula F_1 by an equivalent formula F_2.

1.3 Example

Using the Substitution Theorem and the fact that logical equivalence is reflexive, symmetric, and transitive, we can derive new equivalences from the Boolean algebra axioms. In the following example, each line of the deduction is annotated with the Boolean-algebra axiom that is being used together with ST if the Substitution Theorem is being invoked.

Example 5. We give an equational derivation of the equivalence

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

We have

$$\begin{aligned}
(P \vee (Q \vee R)) \wedge (R \vee \neg P) &\equiv (P \vee Q) \vee R) \wedge (R \vee \neg P) && \text{(Assoc. and ST)} \\
&\equiv (R \vee (P \vee Q)) \wedge (R \vee \neg P) && \text{(Comm. and ST)} \\
&\equiv R \vee ((P \vee Q) \wedge \neg P) && \text{(Distr.)} \\
&\equiv R \vee (\neg P \wedge (P \vee Q)) && \text{(Comm. and ST)} \\
&\equiv R \vee ((\neg P \wedge P) \vee (\neg P \wedge Q)) && \text{(Distr. and ST)} \\
&\equiv R \vee (\mathbf{false} \vee (\neg P \wedge Q)) && \text{(Complement. and ST)} \\
&\equiv R \vee (\neg P \wedge Q) && \text{(Ident. and ST)}
\end{aligned}$$

The algebraic laws in Proposition 1 allow us to relax some distinctions among formulas. For example, the associativity law allows us to unambiguously write $\bigwedge_{i=1}^{n} F_i$ and $\bigvee_{i=1}^{n} F_i$ respectively for the conjunction and disjunction of $F_1, F_2, \ldots, F_n$.

2 Normal Forms

A *literal* is a propositional variable or the negation of a propositional variable. In the former case the literal is *positive* and in the latter case it is *negative*. A formula F is in *conjunctive normal form* (CNF) if it is a conjunction of *clauses*, where each clause is a disjunction 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 clauses, where each clause is a conjunction of literals $L_{i,j}$:

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

Note that we consider **true** to be a CNF formula with no clauses, and we consider **false** to be a CNF formula with a single clause, which contains no literals.

Example 6. The formulas representing the 3-colouring problem and the Sudoku problem in the previous lecture are both CNF formulas.

2.1 Equational Transformation to CNF and DNF

Theorem 7. For every formula F 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

$$\begin{array}{rcl} \neg\neg G & \text{by} & G \\ \neg(G \wedge H) & \text{by} & (\neg G \vee \neg H) \\ \neg(G \vee H) & \text{by} & (\neg G \wedge \neg H) \\ \neg\textbf{true} & \text{by} & \textbf{false} \\ \neg\textbf{false} & \text{by} & \textbf{true} \end{array}$$

   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

$$\begin{array}{rcl} G \vee (H \wedge R) & \text{by} & (G \vee H) \wedge (G \vee R) \\ (H \wedge R) \vee G & \text{by} & (H \vee G) \wedge (R \vee G) \\ G \vee \textbf{true} & \text{by} & \textbf{true} \\ \textbf{true} \vee G & \text{by} & \textbf{true} \end{array}$$

   until no such formulas occur (i.e., push all disjunctions inward until no conjunction occurs under a disjunction).

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Figure 1: Truth table for F

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 $\wedge$ and $\vee$, and swap **true** and **false**).

□

2.2 Normal Forms from Truth Tables

Alternatively, given a formula F, we can read off equivalent DNF and CNF formulas from its truth table.

To obtain a DNF formula, the idea is to have a (conjunctive) clause for each row of the truth table of F for which F has value 1, i.e., for each satisfying assignment of F. The clause corresponding to a given assignment is the unique clause that is satisfied by that assignment and no others. For example, if F has truth table as in Figure 1 we get an equivalent DNF formula

$$(\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge C) \vee (A \wedge \neg B \wedge \neg C) \vee (A \wedge B \wedge C)\,. \tag{1}$$

Dually, we can define a CNF formula that is equivalent to F that has a clause for each row of the truth table of F in which F has value 0. The (disjunctive) clause corresponding to a given row is the unique clause that takes value 0 in that row but 1 in all other rows. For example from Figure 1 we get the CNF formula

$$(A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C) \wedge (\neg A \vee B \vee \neg C) \wedge (\neg A \vee \neg B \vee C)\,. \tag{2}$$

2.3 Summary

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.