7 Propositional logic, truth tables, and resolution

7.1 Truth tables and propositional connectives

Propositional logic is concerned with truth-functions, functions whose values are the two truth-values 0, 1 (for false and true respectively), and whose arguments are also truth-values.

(7.1) Definition boolean variables are variables which are restricted to truth-values. A boolean expression, boolean formula, or formula for short, is a correctly formed expression involving boolean variables and boolean connectives.

Certain truth-functions are well-known.

\[ 0 \rightarrow 1, \quad 1 \rightarrow 0 \]

is simply negation (not). If \( X \) is a boolean variable then \( \neg X \) is its negation. Negation can be represented in a truth table as follows

<table>
<thead>
<tr>
<th>( X )</th>
<th>( \neg X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\( (0, 0) \rightarrow 0, \quad (0, 1) \rightarrow 0, \quad (1, 0) \rightarrow 0, \quad (1, 1) \rightarrow 1 \)

is conjunction (and). If \( X \) and \( Y \) are boolean variables, \( X \land Y \) represents their conjunction. Here is the truth table for conjunction.

<table>
<thead>
<tr>
<th>( X )</th>
<th>( Y )</th>
<th>( X \land Y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

It can also be displayed in a table as follows.

<table>
<thead>
<tr>
<th>( X \land Y )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Disjunction (or) is represented \( X \lor Y \) and has the following table.

<table>
<thead>
<tr>
<th>( X \lor Y )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

(7.2) Definition Two formulae are equivalent if they have the same truth-table.

Implication (if...then) is represented \( X \Rightarrow Y \) and has the following table.
It is just a way of connecting boolean variables, and in fact $X \Rightarrow Y$ is equivalent to $(\neg X) \lor Y$ — the two expressions have the same truth-table. I believe it is called the Philonian conditional.

(7.3) The Philonian conditional is the weakest kind of ‘implication’ which guarantees the following:

If $X$ is true and $(X \Rightarrow Y)$ is true then $Y$ is true.

(7.4) The propositional connectives have various familiar properties: $\land$ is commutative and associative, etcetera. Importantly,

(De Morgan laws.) $\neg(X \land Y)$ and $(\neg X) \lor (\neg Y)$ are equivalent, and $\neg(X \lor Y)$ and $(\neg X) \land (\neg Y)$ are equivalent.

(7.5) Conventions about precedence of connectives. Just as with arithmetic expressions, it is convenient to drop parentheses from boolean expressions. To avoid overload, we’ll not say what they are!

7.2 Truth-functions, tautologies, and contradictions

To begin with, every truth-function can be realised by a boolean expression using only $\land$, $\lor$, $\neg$.

(7.6) Definition If $X$ is a boolean variable, we sometimes write $\overline{X}$ to mean $\neg X$.

Both $X$ and $\overline{X}$ are called literals. If $L = \overline{X}$ then we define $X = \overline{L}$ (which is obviously correct, since $\neg X$ is equivalent to $X$.

We write $\pm L$ to mean $L$ or $\overline{L}$.

(7.7) Lemma Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function. There exists a formula formed from the boolean variables $X_1, \ldots, X_n$, which is equivalent to $f$.

Half proof. Suppose that in $k$ rows of the truth-table for $f$ the value of $f$ is 1. The formula has the form

$$D_1 \lor D_2 \lor \ldots \lor D_k$$

where each subformula $D_i$ is of the form

$$(\pm X_1 \land \ldots \land \pm X_n).$$

Suppose the values in the $i$-th such row are $x_1, \ldots, x_j$. Then $X_j$ occurs in the formula if $x_j = 1$, and $\overline{X_j}$ occurs if $x_j = 0$.

This breaks down if $k = 0$, so $f$ takes the constant value 0. $X_1 \land \overline{X}_1$ will do in this case.

(7.8) Definition Formulae of this kind:

$$(\pm X_{i_1} \land \ldots \land \pm X_{i_{k_1}}) \lor (\pm X_{i_{k_1}} \land \ldots \land \pm X_{i_{k_2}}) \ldots$$

are said to be in disjunctive normal form or DNF. They are a disjunction of conjunctions.
Example. Construct a DNF expression for $X_1 \Rightarrow X_2$.

$$\begin{array}{c|c|c}
X_1 & X_2 & X_1 \Rightarrow X_2 \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}$$

$$(\overline{X_1} \land \overline{X_2}) \lor (\overline{X_1} \land X_2) \lor (X_1 \land X_2).$$

Another example.

$$\begin{array}{c|c|c|c}
X_1 & X_2 & X_3 & f \\
0 & 0 & 0 & 0 \\
0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
\end{array} \quad \begin{array}{c|c|c|c}
X_1 & X_2 & X_3 & f \\
1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 \\
\end{array}$$

The DNF is easily formed by picking out the rows where the $f$-value is 1.

$$(\overline{X_1} \land \overline{X_2} \land X_3) \lor (\overline{X_1} \land X_2 \land \overline{X_3}) \lor (X_1 \land \overline{X_2} \land X_3)$$

(7.9) Definition A formula is in conjunctive normal form (CNF) if it is of the form

$$(L_1 \lor L_2 \lor \ldots \lor L_k) \land (L_{k+1} \lor L_{k+2} \lor \ldots \lor L_t) \land \ldots \land (L_{r+1} \lor L_{r+2} \lor \ldots \lor L_s)$$

where $L_1, \ldots, L_s$ are literals, not necessarily distinct.

(7.10) Corollary Every truth-function can be realised by a CNF.

Proof. Let $D$ be a DNF realising the negation of $f(T_1, \ldots, T_n)$. The formula $\neg D$ is easily converted into a CNF using De Morgan’s laws, and it realises $f$. Q.E.D.

7.3 The first goal of mathematical logic

(7.11) Definition Let $F$ be a boolean formula involving the variables $X_1, \ldots, X_n$. An interpretation or truth-assignment to $F$ is a map $X_1 \mapsto T_1, \ldots, X_n \mapsto T_n$, where $T_1, \ldots, T_n$ is a vector of truth-values. There are $2^n$ interpretations of $F$.

A boolean formula is a tautology if it is true in all interpretations, and it is inconsistent if it is false in all interpretations.

The first goal is to provide methods for proving true things which are true.

At present, the ‘things’ are Boolean formulae, and ‘true’ means ‘tautology.’

A certain way of proving something (containing $n$ Boolean variables) true is to check it against all $2^n$ interpretations — in other words, build the truth-table.

Resolution (see below) provides a generally more efficient method. It is not always very efficient, as was shown at different times by Tseitin, Galil, Haken, and Fouks. The P=NP? question makes it very doubtful that truly efficient methods exist.
7.4 Resolution proofs and refutations

There is an important proof method called *Robinson’s Resolution Principle*. It can be applied to a DNF to test for a tautology and to a CNF to test for inconsistency (it is easy, but not very useful, to test a CNF for tautology or a DNF for inconsistency). The method is essentially the same for each.

We consider testing a CNF for inconsistency (i.e., whether it is contradictory).

The subformulae $L_i \lor L_{i+1} \lor \ldots \lor L_j$ are called *clauses*. One regards each clause as a *set* of literals. This is acceptable because $\lor$ is commutative and associative. One also views the CNF as a *set* of clauses, and repeatedly adds *resolvents* to the set of clauses.

Given two clauses $C$ and $C'$, a *resolvent* of $C$ and $C'$ is constructed as follows. It is necessary that $C$ contains a literal $L$ whose complement $\overline{L}$ occurs in $C'$. In this case suppose

$$C = L_1 \lor \ldots \lor L_k \lor L$$

and

$$C' = L'_1 \lor \ldots \lor L'_m \lor \overline{L}$$

then the clause obtained by *resolving* $L$ and $\overline{L}$ is

$$L_1 \lor \ldots \lor L_k \lor L'_1 \lor \ldots \lor L'_m.$$

It is possible that $k = m = 0$, in which case the resolvent is not a conventional formula but is called the *empty* clause and written $\Box$.

Put differently:

$$C_1 \lor L, C_2 \lor \overline{L} \rightarrow C_1 \lor C_2$$

**Note.** We extend the definition of truth-value under a truth-assignment, by saying that a clause (in a CNF) is true if and only if at least one of the literals in the clause is true.

This extends the definition because $\Box$ is automatically false, whatever the interpretation.

It does no harm to regard a CNF as a *list* of clauses, or even a *set* of clauses in no particular order.

To construct a **Resolution refutation** of a CNF $F$ means to start with $F$ (as a list of clauses) and repeatedly add new clauses to the list by resolving clauses already present, until the list contains $\Box$.

For example, Modus Ponens is another ‘inference rule’ (see 7.3):

From $X$ and $X \rightarrow Y$, infer $Y$.

The following is a kind of justification of Modus Ponens: we show that

$$X, \overline{X} \lor Y, \overline{Y}$$

are inconsistent.

$$X, \overline{X} \lor Y, \overline{Y}$$

$$X, \overline{X} \lor Y, \overline{Y}, Y$$

$$X, \overline{X} \lor Y, \overline{Y}, Y, \Box$$
Or we may present the proof by listing the clauses as they are supplied or generated by resolution.

Given the CNF

\[ A \lor D, \quad \overline{A} \lor \overline{D}, \quad \overline{A} \lor B \lor C, \quad A \lor B \lor C, \quad \overline{B}, \quad D \lor C, \quad \overline{D} \lor \overline{C} \]

here is a resolution refutation (proof of inconsistency).

\[
\begin{align*}
A \lor B \lor C, & \quad \overline{B} \quad \rightarrow A \lor C \\
\overline{A} \lor B \lor C, & \quad \overline{B} \quad \rightarrow \overline{A} \lor \overline{C} \\
A \lor C, & \quad \overline{C} \lor \overline{D} \quad \rightarrow A \lor \overline{D} \\
A \lor D, & \quad A \lor \overline{D} \quad \rightarrow A \\
A, & \quad \overline{A} \lor \overline{D} \quad \rightarrow \overline{D} \\
C \lor D, & \quad \overline{D} \quad \rightarrow C \\
\overline{A} \lor C, & \quad C \quad \rightarrow \overline{A} \\
A, & \quad \overline{A} \quad \rightarrow \square
\end{align*}
\]
7.5 Proof trees

A resolution refutation can be given in a tree-like arrangement as illustrated in Figure 1. If we label the edges by the literals eliminated, and reverse the direction, and remove the resolvents labelling the tree nodes, we get a structure as illustrated in Figure 2. The interesting thing about this arrangement is that for every leaf in the tree, every literal in the clause labelling in the leaf occurs as an edge-label in the path from the root to that leaf.

(7.12) Lemma Suppose that $C_1 \lor L$ and $C_2 \lor \overline{L}$ are clauses, and $I$ is an interpretation satisfying both clauses. Then $I$ also satisfies the resolvent $C_1 \lor C_2$.

Proof. If $L$ is false under $I$, then $C_1$ must be true under $I$. If $L$ is true under $I$, then $C_2$ must be true under $I$. In either case, $C_1 \lor C_2$ is true under $I$.

(7.13) Lemma Given a CNF $S$, if $\square$ can be constructed from the clauses in $S$ using resolution, then $S$ is false in every interpretation.

Proof. So (using the above lemma with induction) if $I$ makes $S$ true, then it makes all resolvents derived from $S$ true, and makes $\square$ true. However, $\square$ is false under every interpretation.

(7.14) Definition Let $S$ be a set of clauses and $L$ a literal.

\[ S \setminus L = \{ \text{def} C \setminus \{L\} : C \in S \land \overline{L} \notin C \} \]

(7.15) Lemma If $D$ is a clause derivable from $S \setminus L$, then either $D$ or $D \lor L$ is derivable from $S$.

Proof. Induction on the length (number of clauses) of the derivation. The basis is where $D \in S \setminus L$, whence $D$ or $D \lor L \in S$.

If $D = D_1 \lor D_2$ derived by resolution from $D_1 \lor X$ and $D_2 \lor \overline{X}$, then by induction $D_1 \lor X$ or $D_1 \lor X \lor L$, and $D_2 \lor \overline{X}$ or $D_2 \lor \overline{X} \lor L$, are derivable from $S$, whence either $D_1 \lor D_2$ or $D_1 \lor D_2 \lor L$ are derivable from $S$.

(7.16) Lemma If $S$ is inconsistent then $S \setminus L$ is inconsistent.

Proof. Equivalently: if $S \setminus L$ is consistent, so is $S$. Let $X_1, \ldots, X_n$ be the boolean variables occurring in $S$, where $L = X_n$ or $L = \overline{X_n}$.

If $S \setminus L$ is consistent then there is a truth-assignment $I$ to $X_1, \ldots, X_{n-1}$, making $S \setminus L$ true. So $I$ makes all clauses in \[ \{ C \setminus \{L\} : C \in S \land \overline{L} \notin C \} \] true. In other words, every clause in $S \setminus L$ contains a literal $L'$ different from $L, \overline{L}$ such that $L'$ is true under $I$. and therefore if we extend $I$ to include $X_n$ and make $L$ false, $I$ still makes every clause in \[ \{ C \in S : \overline{L} \notin C \} \] true. Since it makes $L$ false, it also makes every clause \[ \{ C \in S : \overline{L} \in C \} \] true, so it makes $S$ true.
(7.17) **Theorem** A CNF $S$ is inconsistent if and only if the empty clause is in $S$ or can be generated from $S$ by resolution.

**Sketch proof.** The ‘if’ part has been mentioned already (7.13).

Only if: by induction on $n$, the number of boolean variables in $S$. Immediate if $n = 0$: $S = \emptyset$ (consistent) or $S = \{\Box\}$ (inconsistent). If $n = 1$, $S$ contains just one boolean variable $X$, and is inconsistent, then either $\Box \in S$ or $X, \overline{X} \in S$, and in any case $\Box$ can be generated.

Induction: Choose $X \in S$. By the above lemma, $S\setminus X$ and $S\setminus \overline{X}$ are inconsistent. By the inductive hypothesis $\Box$ can be generated both from $S\setminus X$ and $S\setminus \overline{X}$.

By the above Lemma, $\Box$ or $X$ can be derived from $S$, and $\Box$ or $\overline{X}$ can be derived from $S$; in any case, $\Box$ can be derived from $S$.  

**Example.** $S = XY, \overline{X}Y$. $S\setminus X = Y, \overline{Y} \rightarrow \Box$. Thus from $S$, $XY, \overline{Y} \rightarrow X$.

$S\setminus \overline{X} = Y, \overline{Y} \rightarrow \Box$. Thus from $S$, $\overline{X}Y, \overline{Y} \rightarrow X$.

Then $X, \overline{X} \rightarrow \Box$.  
