9 Resolution

9.1 The first goal of mathematical logic

(9.1) 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.

9.2 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 $L_i \vee L_{i+1} \vee \ldots \vee L_j$ are called *clauses.* One regards each clause as a *set* of literals. This is acceptable because $\vee$ 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

$$C = L_1 \vee \ldots \vee L_k \vee L \quad \text{and} \quad C' = L'_1 \vee \ldots \vee L'_m \vee \overline{L}$$

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

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

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:

$$C_1 \vee L,\ C_2 \vee \overline{L} \quad \mapsto \quad C_1 \vee C_2$$

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 **??**):

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

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

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

are inconsistent.

$$\begin{array}{r} X, \overline{X} \vee Y, \overline{Y} \\ X, \overline{X} \vee Y, \overline{Y}, Y \\ X, \overline{X} \vee Y, \overline{Y}, Y, \Box \end{array}$$

Or we may present the proof by listing the clauses as they are supplied or generated by resolution.

Given the CNF

$$A \vee D, \quad \overline{A} \vee \overline{D}, \quad \overline{A} \vee B \vee \overline{C}, \quad A \vee B \vee C, \quad \overline{B}, \quad D \vee C, \quad \overline{D} \vee \overline{C}$$

here is a resolution refutation (proof of inconsistency).

$$\begin{array}{rrcr} A \vee B \vee C, & \overline{B} & \mapsto & A \vee C \\ \overline{A} \vee B \vee \overline{C}, & \overline{B} & \mapsto & \overline{A} \vee \overline{C} \\ A \vee C, & \overline{C} \vee \overline{D} & \mapsto & A \vee \overline{D} \\ A \vee D, & A \vee \overline{D} & \mapsto & A \\ A, & \overline{A} \vee \overline{D} & \mapsto & \overline{D} \\ C \vee D, & \overline{D} & \mapsto & C \\ \overline{A} \vee \overline{C}, & C & \mapsto & \overline{A} \\ A, & \overline{A} & \mapsto & \Box \end{array}$$

9.3 Proof trees

A resolution refutation can be given in a tree-like arrangement as illustrated in Figure 1.

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

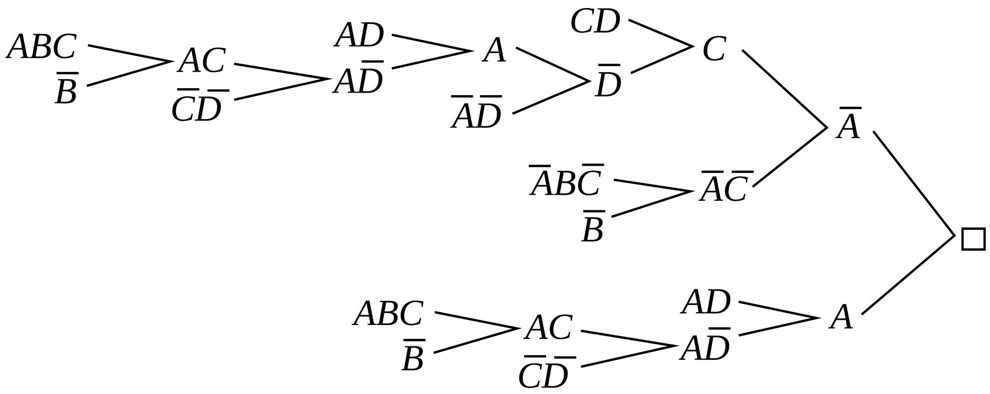

Figure 1: resolution proof tree

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, $C_1 \vee C_2$ is true under I. ∎

(9.3) 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. ∎

(9.4) Definition *Let S be a set of clauses and L a literal.*

$$S \backslash L = (\mathit{def}) \quad \{C \backslash \{L\} : \quad C \in S \wedge \overline{L} \notin C\}.$$

(9.5) Lemma *If D is a clause derivable from $S \backslash L$, then either D or $D \vee L$ is derivable from S.*

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

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

(9.6) Lemma *If S is inconsistent then $S \backslash L$ is inconsistent.*

Proof. Equivalently: if $S \backslash L$ is consistent, so is S.

Let I be a truth-assignment which makes every clause in $S \backslash L$ true. Every such clause is of the form D or $D \vee L$ where L does not occur in D. In any case, if we extend I so that it makes L *false*, it makes D and $D \vee L$ true.

This covers every clause in S which does not contain $\overline{L}$, and the ones which do contain it are also true under I. ∎

(9.7) 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 (9.3).

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

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

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

Example. $S = XY, \overline{X}Y, \overline{Y}$. $S \backslash X = Y, \overline{Y} \mapsto \square$. Thus from S, $XY, \overline{Y} \mapsto X$.

$S \backslash \overline{X} = Y, \overline{Y} \mapsto \square$. Thus from S, $\overline{X}Y, \overline{Y} \mapsto \overline{X}$.

Then $X, \overline{X} \mapsto \square$.