The resolution method

Andrés E. Caicedo

September 11, 2011

This note is based on lecture notes for the Caltech course Math 6c, prepared with A. Kechris and M. Shulman.

We would like to have a mechanical procedure (algorithm) for checking whether a given set of formulas logically implies another, that is, given $A_1, \dots, A_n, A$, whether

$$(A_1 \wedge \cdots \wedge A_n) \Rightarrow A$$

is a tautology (i.e., it is true under all truth-value assignments.)

This happens iff

$$A_1 \wedge \cdots \wedge A_n \wedge \neg A \text{ is unsatisfiable.}$$

So it suffices to have an algorithm to check the (un)satisfiability of a single propositional formula. The method of truth tables gives one such algorithm. We will now develop another method which is often (with various improvements) more efficient in practice.

It will be also an example of a *formal calculus*. By that we mean a set of rules for generating a sequence of strings in a language. Formal calculi usually start with a certain string or strings as given, and then allow the application of one or more "rules of production" to generate other strings.

A formula A is in*conjunctive normal form* iff it has the form

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n$$

where each A_i has the form

$$B_1 \vee B_2 \vee \cdots \vee B_k$$

and each B_i is either a propositional variable, or its negation. So A is in conjunctive normal form iff it is a conjunction of disjunctions of variables and negated variables. The common terminology is to call a propositional variable or its negation a *literal*.

Suppose A is a propositional statement which we want to test for satisfiability. First we note (without proof) that although there is no known efficient algorithm for finding A' in cnf (conjunctive normal form) equivalent to A, it is not hard to show that there is an efficient algorithm for finding A^* in cnf such that:

$$A \text{ is satisfiable iff } A^* \text{ is satisfiable.}$$

(But, in general, A^* has more variables than A.)

So from now on we will only consider As in cnf, and the Resolution Method applies to such formulas only. Say

$$A = (\ell_{1,1} \vee \cdots \vee \ell_{1,n_1}) \wedge \cdots \wedge (\ell_{k,1} \vee \cdots \vee \ell_{k,n_k})$$

with $\ell_{i,j}$ literals. Since order and repetition in each conjunct

$$\ell_{i,1} \vee \cdots \vee \ell_{i,n_i} \tag{$*$}$$

are irrelevant, we can replace ($*$) by the set of literals

$$c_i = \{\ell_{i,1}, \ell_{i,2}, \ldots, \ell_{i,n_i}\}.$$

Such a set of literals is called a *clause*. It corresponds to the formula ($*$). So the wff A above can be simply written as a set of clauses (again since the order of the conjunctions is irrelevant):

$$\begin{aligned} C &= \{c_1, \ldots, c_k\} \\ &= \{\{\ell_{i,1}, \ldots \ell_{i,n_1}\}, \ldots, \{\ell_{k,1}, \ldots, \ell_{k,n_k}\}\} \end{aligned}$$

Satisfiability of A means then simultaneous satisfiability of all of its clauses $c_1, \ldots, c_k$, i.e., finding a valuation ν which makes c_i true for each i, i.e., which for each i makes some $\ell_{i,j}$ true.

Example 1

$$\begin{aligned} A &= (p_1 \vee \neg p_2) \wedge (p_3 \vee p_3) \\ c_1 &= \{p_1, \neg p_2\} \\ c_2 &= \{p_3\} \\ C &= \{\{p_1, \neg p_2\}, \{p_3\}\}. \end{aligned}$$

From now on we will deal only with a set of clauses $C = \{c_1, c_2, \ldots, c_n\}$. It is possible to consider also infinite sets C, but we will not do that here.

Satisfying C means (again) that there is a valuation which satisfies all $c_1, c_2, \ldots,$ i.e. if $c_i = \ell_{i,1} \vee \cdots \vee \ell_{i,n_i}$, then for all i there is j so that it makes $\ell_{i,j}$ true.

Notice that if the set of clauses C_A is associated as above to A (in cnf) and C_B to B, then

$$A \wedge B \text{ is satisfiable iff } C_A \cup C_B \text{ is satisfiable.}$$

By convention we also have the *empty clause* $\square$, which contains no literals. The empty clause is (by definition) unsatisfiable, since for a clause to be satisfied by a valuation, there has to be some literal in the clause which it makes true, but this is impossible for the empty clause, which has no literals.

For a literal u, let $\bar{u}$ denote its "conjugate", i.e.

$$\begin{aligned} \bar{u} &= \neg p, \text{ if } u = p, \\ \bar{u} &= p \text{ if } u = \neg p. \end{aligned}$$

Definition 1 *Suppose now c_1, c_2, c are three clauses. We say that c is a resolvent of c_1, c_2 if there is a u such that $u \in c_1$, $\bar{u} \in c_2$ and*

$$c = (c_1 \setminus \{u\}) \cup (c_2 \setminus \{\bar{u}\}).$$

We denote this by the diagram

$$\begin{array}{c} c \\ \\ c_1 \quad c_2 \end{array}$$

We allow here the case $c = \Box$, i.e. $c_1 = \{u\},\ c_2 = \{\bar{u}\}$.

Example 2 *(i)*

$$\begin{array}{c} \{p, r\} \\ \\ \{p, \neg q, r\}\ \{q, r\} \end{array}$$

(ii)

$$\begin{array}{c} \{q, \neg q\} \\ \\ \{p, \neg q\}\ \{\neg p, q\} \end{array} \qquad\qquad \begin{array}{c} \{p, \neg p\} \\ \\ \{p, \neg q\}\ \{\neg p, q\} \end{array}$$

(iii)

$$\begin{array}{c} \Box \\ \\ \{p\}\ \{\neg p\} \end{array}$$

Proposition 2 *If c is a resolvent of c_1, c_2, then any assignment of truth values that makes both c_1 and c_2 true also makes c true. (We view here c_1, c_2, c as formulas.)*

Proof: Suppose a valuation ν satisfies both c_1, c_2 and let u be the literal used in the resolution. If $\nu(u) = 1$, then since $\nu(c_2) = 1$ we clearly have $\nu(c_2 \setminus \{\bar{u}\}) = 1$ and so $\nu(c) = 1$. If $\nu(u) = 0$, then $\nu(c_1 \setminus \{u\}) = 1$, so $\nu(c) = 1$. $\Box$

Definition 3 *Let now C be a set of clauses. A proof by resolution from C is a sequence $c_1, c_2, \ldots, c_n$ of clauses such that each c_i is either in C or else it is a resolvent of some c_j, c_k with $j, k < i$. We call c_n the goal or conclusion of the proof. If $c_n = \Box$, we call this a proof by resolution of a contradiction from C or simply a refutation of C.*

Example 3 *Let $C = \{\{p, q, \neg r\}, \{\neg p\}, \{p, q, r\}, \{p, \neg q\}\}$. Then the following is a refutation of C:*

$$
\begin{array}{ll}
c_1 = \{p, q, \neg r\} & \textit{(in } C\textit{)} \\
c_2 = \{p, q, r\} & \textit{(in } C\textit{)} \\
c_3 = \{p, q\} & \textit{(resolvent of } c_1, c_2 \textit{ (by } r\textit{))} \\
c_4 = \{p, \neg q\} & \textit{(in } C\textit{)} \\
c_5 = \{p\} & \textit{(resolvent of } c_3, c_4 \textit{ (by } q\textit{))} \\
c_6 = \{\neg p\} & \textit{(in } C\textit{)} \\
c_7 = \Box & \textit{(resolvent of } c_5, c_6 \textit{ (by } p\textit{)).}
\end{array}
$$

We can also represent this by a "tree": There are lines from $\Box$ to $\{p\}$ and $\{\neg p\}$, from $\{p\}$ to $\{p, q\}$ and $\{p, \neg q\}$, and from $\{p, q\}$ to $\{p, q, r\}$ and $\{p, q, \neg r\}$:

$$
\begin{array}{ccccc}
 & & & \Box & \\
 & & \{p\} & & \{\neg p\} \\
 & \{p, q\} & & \{p, \neg q\} & \\
\{p, q, r\} & & \{p, q, \neg r\} & &
\end{array}
$$

Terminal nodes correspond to clauses in C and each $\wedge$ creates a "branch" of the tree, corresponds to creating a resolvent. We call such a tree a *resolution tree.*

Example 4 *Let* $C = \{\{\neg p, s\}, \{p, \neg q, s\}, \{p, q, \neg r\}, \{p, r, s\}, \{\neg s\}\}$.

$$
\begin{array}{ccccc}
 & & \Box & & \\
 & \{s\} & & & \{\neg s\} \\
\{\neg p, s\} & & \{p, s\} & & \\
 & \{p, \neg q, s\} & & \{p, q, s\} & \\
 & & \{p, q, \neg r\} & & \{p, r, s\}
\end{array}
$$

This can be also written as a proof as follows:

$$
\begin{array}{lcl}
c_1 & = & \{p, q, \neg r\} \\
c_2 & = & \{p, r, s\} \\
c_3 & = & \{p, q, s\}
\end{array}
$$

$$\begin{array}{rcl} c_4 & = & \{p, \neg q, s\} \\ c_5 & = & \{p, s\} \\ c_6 & = & \{\neg p, s\} \\ c_7 & = & \{s\} \\ c_8 & = & \{\neg s\} \\ c_9 & = & \Box \end{array}$$

(This proof is not unique. For example, we could move c_8 before c_3 and get another proof corresponding to the same resolution tree. The relationship between proofs by resolution and their corresponding trees is similar to that between parsing sequences and parse trees.)

The goal of proofs by resolution is to prove unsatisfiability of a set of clauses. The following theorem tells us that they achieve their goal.

Theorem 4 *Let $C = \{c_1, c_2, \ldots, c_m\}$ be a set of clauses. Then C is unsatisfiable iff there is a refutation of C.*

The argument below uses the technique of *mathematical induction*, that we will study later. You do not need to read this proof now, but I am including it here so we can use it when the time comes. Feel free to stop by my office if you read the argument and have questions about it.

PROOF:

$\Leftarrow$: . This is usually called "Soundness of the proof system". ("Soundness" is another word for "correctness".)

Let $d_1, \ldots, d_n$ be a proof of resolution from C. Then we can easily prove, by induction on $1 \leq i \leq n$, that any assignment making all the c_i true, must also make d_i true. But if $d_n = \Box$, then d_n is unsatisfiable, and therefore C must also be unsatisfiable.

$\Rightarrow$:. This is usually called "Completeness of the proof system".

First we can assume that C has no clause c_i which contains, for some literal u, both u and $\bar{u}$ (since such a clause can be dropped from C without affecting its satisfiability).

Notation. If u is a literal, let $C(u)$ be the set of clauses resulting from C by canceling every occurrence of u within a clause of C and eliminating all clauses of C containing $\bar{u}$ (this effectively amounts to setting $u = 0$).

Example. Let $C = \{\{p, q, \neg r\}, \{p, \neg q\}, \{p, q, r\}, \{q, r\}\}$. Then

$$C(r) = \{\{p, \neg q\}, \{p, q\}, \{q\}\}$$
$$C(\bar{r}) = \{\{p, q\}, \{p, \neg q\}\}$$

Note that $u, \bar{u}$ do not occur in $C(u)$, $C(\bar{u})$. Note also that if C is unsatisfiable, so are $C(u)$, $C(\bar{u})$. Because if ν is a valuation satisfying $C(u)$, then, since $C(u)$ does not contain

$u, \bar{u}$, we can assume that ν does not assign a value to u. Then the valuation ν' which agrees on all other variables with ν and gives $\nu(u) = 0$ satisfies C. Similarly for $C(\bar{u})$.

So assume C is unsatisfiable, in order to construct a refutation of C. Say that all the propositional variables occurring in clauses in C are among $p_1, \ldots, p_n$. We prove then the result by induction on n. In other words, we show that for each n, if C is a finite set of clauses containing variables among $p_1, \ldots, p_n$ and C is unsatisfiable, there is a refutation of C.

$n = 1$. In this case, we must have $C = \{\{p_1\}, \{\neg p_1\}\}$, and hence we have the refutation $\{p_1\},\ \{\neg p_1\},\ \Box$.

$n \to n{+}1$. Assume this has been proved for sets of clauses with variables among $\{p_1, \ldots, p_n\}$ and consider a set of clauses C with variables among $\{p_1, \ldots, p_n, p_{n+1}\}$. Let $u = p_{n+1}$.

Then $C(u), C(\bar{u})$ are also unsatisfiable and do not contain p_{n+1}, so by induction hypothesis there is a refutation $d_1, \ldots d_m, d_{m+1} = \Box$ for $C(u)$ and a refutation $e_1, \ldots, e_k, e_{k+1} = \Box$ for $C(\bar{u})$.

Consider first $d_1, \ldots, d_{m+1}$. Each clause d_i is in $C(u)$ or comes as a resolvent of two previous clauses. Define then recursively $d'_1, \ldots, d'_m, d'_{m+1}$, so that either $d'_i = d_i$ or $d'_i = d_i \cup \{u\}$.

If $d_i \in C(u)$, then it is either in C and then we put $d'_i = d_i$ or else is obtained from some $d^*_i \in C$ by dropping u, i.e., $d_i = d^*_i \setminus \{u\}$. Then put $d'_i = d^*_i$.

The other case is where for some $j, k < i$, we have that d_i is a resolvent of d_j, d_k, and thus by induction d'_j, d'_k are already defined. The variable used in this resolution is in $\{p_1, \ldots, p_n\}$, so we can use this variable to resolve from d'_j, d'_k to get d'_i.

Thus $d'_{m+1} = \Box$ or $d'_{m+1} = \{p_{n+1}\}$, and $d'_1, \ldots d'_m, d'_{m+1}$ is a proof by resolution from C. If $d'_{m+1} = \Box$ we are done, so we can assume that $d'_{m+1} = \{p_{n+1}\}$, i.e., $d'_1, \ldots, d'_m, \{p_{n+1}\}$ is a proof by resolution from C. Similarly, working with $\bar{u}$, we can define $e'_1, \ldots e'_k, e'_{k+1}$, a proof by resolution from C with $e'_{k+1} = \Box$ or $e'_{k+1} = \{\neg p_{n+1}\}$. If $e'_{k+1} = \Box$ we are done, otherwise $e'_1, \ldots, e'_k, \{\neg p_{n+1}\}$ is a proof by resolution from C. Then

$$d'_1, \ldots, d'_m, \{p_{n+1}\}, e'_1, \ldots, e'_k, \{\neg p_{n+1}\}, \Box$$

is a refutation from C. $\dashv$

Example 5

$$\begin{aligned} C &= \{\{p, q, \neg r\}, \{\neg p\}, \{p, q, r\}, \{p, \neg q\}\} \qquad (u = r) \\ C(r) &= \{\{\neg p\}, \{p, q\}, \{p, \neg q\}\} \\ C(\neg r) &= \{\{p, q\}, \{\neg p\}, \{p, \neg q\}\} \end{aligned}$$

Refutation from $C(r)$	*Proof by resolution from C*	*Refutation from $C(\neg r)$*	*Proof by resolution from C*
$\{p,q\}$	$\rightarrow \{p,q,r\}$	$\{p,q\}$	$\rightarrow \{p,q,\neg r\}$
$\{p,\neg q\}$	$\rightarrow \{p,\neg q\}$	$\{p,\neg q\}$	$\rightarrow \{p,\neg q\}$
$\{p\}$	$\rightarrow \{p,r\}$	$\{p\}$	$\rightarrow \{p,\neg r\}$
$\{\neg p\}$	$\rightarrow \{\neg p\}$	$\{\neg p\}$	$\rightarrow \{\neg p\}$
$\Box$	$\rightarrow \{r\}$	$\Box$	$\rightarrow \{\neg r\}$

$\Box$

Remark 5 *Notice that going from n to $n+1$ variables "doubles" the length of the proof, so this gives an exponential bound for the refutation.*

Remark 6 *The method of refutation by resolution is non-deterministic–there is no unique way to arrive at it. Various strategies have been devised for implementing it.*

One is by following the recursive procedure used in the proof of theorem 4. Another is by brute force. Start with a finite set of clauses C. Let $C_0 = C$. Let $C_1 = C$ together with all clauses obtained by resolving all possible pairs in $C_0, C_2 = C_1$ together with all clauses obtained by resolving all possible pairs from C_1, etc. Since any set of clauses whose variables are among $p_1, \ldots, p_n$ cannot have more than 2^{2n} elements, this will stop in at most 2^{2n} many steps. Put $C_{2^{2n}} = C^*$. If $\Box \in C^*$ then we can produce a refutation proof of about that size (i.e., 2^{2n}). Otherwise, $\Box \notin C^*$ and C is satisfiable.

Other strategies are more efficient in special cases, but none are known to work in general.