GENERALIZED RESOLUTION AND CUTTING PLANES*

J.N. HOOKER

Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, USA

Abstract

This paper illustrates how the application of integer programming to logic can reveal parallels between logic and mathematics and lead to new algorithms for inference in knowledge-based systems. If logical clauses (stating that at least one of a set of literals is true) are written as inequalities, then the resolvent of two clauses corresponds to a certain cutting plane in integer programming. By properly enlarging the class of cutting planes to cover clauses that state that at least a specified number of literals are true, we obtain a generalization of resolution that involves both cancellation-type and circulant-type sums. We show its completeness by proving that it generates all prime implications, generalizing an early result by Quine. This leads to a cutting-plane algorithm as well as a generalized resolution algorithm for checking whether a set of propositions, perhaps representing a knowledge base, logically implies a given proposition. The paper is intended to be readable by persons with either an operations research or an artificial intelligence background.

Keywords

Propositional logic, resolution, integer programming, theorem proving, artificial intelligence.

0. Introduction

George Boole believed that mathematical rigor can be achieved in logic, and if we ignore some minor lapses, he proved it with his famous logical algebra. However, Boolean algebra is "mathematical" only in the sense that it is rigorous and permits one to calculate logical consequences. Although Boole styled logical operations after

*This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University. Reproduction in whole or in part is permitted for any purpose of the U.S. Government.

arithmetical ones, he did not really seek to make logic a branch or application of mathematics traditionally conceived, and it did not worry him that his calculations differed from ordinary addition and multiplication ([4], opening section). He wanted only to realize Leibniz's idea of a calculus of reasoning, a *calculus ratiocanator,* for the logic of classes and of propositions.

Recent work in automatic deduction and knowledge-based systems has revived Boole's project. Although most modern methods are like Boole's in that they involve symbolic processing rather than traditional calculations with numbers, one approach really *is* mathematical. It applies one of the standard techniques of operations research, mathematical programming, to logical inference. Papers by Blair, Jeroslow and Lowe [3] and Lowe [18], as well as this author [11], suggest that by solving a certain integer program, one can determine quickly whether a given proposition logically follows from a large set of propositions, perhaps representing a knowledge base. Other papers by Jeroslow [12–14] show how mixed integer linear models can be applied to inferences involving uncertainty, inferences in the predicate calculus, database queries, etc., although it is unclear at this stage how efficient these methods are.

These truly mathematical models of logic are in a way more Boolean than the others in that they, like Boole's original system, use intermediate steps that have no interpretation other than as technical devices for obtaining the solution. For instance, if one uses the simplex method to help solve an integer programming model, he would not necessarily try to find logical meaning in the simplex pivots. The presence of uninterpreted steps has traditionally been regarded as undesirable, and Boole's immediate successors eliminated them from his system (see [16], sect. VI.3-4). This attitude may be appropriate if one aim of a logical calculus is to serve as an explanatory model of deduction in some sense, but if the aim is merely to provide a fast procedure for checking the validity of an inference, such scruples are out of place. As it happens, the uninterpreted steps of Boole's procedure serve no useful purpose and are just as well eliminated. However, a willingness to tolerate them when they are part of a powerful mathematical model can bring two advantages. It can lead one to see connections between logic and mathematics that Boole never imagined, and these connections can suggest new algorithms, both mathematical and symbolic.

The intent of this paper is to illustrate these two advantages by applying mathematics to the technique of resolution in propositional logic. We will observe that resolvents have a simple but striking resemblance to a special class of Chvátal cutting planes in integer programming, and we will show that by enlarging this class we can generalize resolution to cover a larger class of logical formulae. Ordinary resolution applies to statements in conjunctive normal form, i.e. to conjunctions of clauses, each of which asserts that at least one of a set of literals is true. Our generalized resolution will apply to conjunctions of clauses, each of which asserts that at least a certain number of a set of literals in true.

We will prove the completeness of generalized resolution by showing that it yields all the prime implications of any consistent set of clauses to which it is applied.

The proof will be built upon Quine's [22,23] early proof of the same for ordinary resolution. Moreover, these prime implications supply all of the cutting planes one needs to solve an integer programming model of logical inference. Thus, the connection between resolution and integer programming yields both a symbolic algorithm (generalized resolution) and a mathematical algorithm (integer programming with cutting planes) for logical inference.

1. Clauses

We will restrict ourselves to logical formulae that are conjunctions of clauses. Suppose we begin with a set of atomic propositions $x_1, \ldots, x_n$. A *clause* in the ordinary sense is a statement like

$$x_1 \vee \neg x_4 \vee \neg x_5 \vee x_8 \,, \tag{1}$$

where "$\vee$" is an inclusive "or" and "$\neg$" means "not". Statement (1) asserts that at least one of the *literals* $x_1, \neg x_4, \neg x_5, x_8$ is true.

We will adopt the common device of writing (1) as a mathematical inequality,

$$x_1 + (1-x_4) + (1-x_5) + x_8 \geqslant 1. \tag{2}$$

Here, x_j is a mathematical variable interpreted as having the value 1 when the proposition x_j in (1) is true and 0 when x_j is false. Thus, if we suppose that each x_j is binary (i.e. can take only the values 0 and 1), (2) again asserts that at least one of the four literals in (1) is true. In what follows, we will loosely refer to an inequality representing a clause as a clause.

Note that (2) can be written $x_1 - x_4 - x_5 + x_8 \geqslant 1 - 2$. This suggests the following general notation for clauses:

$$cx \geqslant \beta - n(c), \tag{3}$$

where c is a row vector and x a column vector, and where $n(c)$ is the number of negative components in the vector c. Each c_j is 1 to indicate that the literal x_j appears, -1 to indicate that $\neg x_j$ appears, or 0 to indicate that neither appears; we suppose that not all c_j's are zero. This notation cannot represent tautological clauses, but this will be convenient.

In ordinary clauses, $\beta = 1$, but we wish to allow β to take larger integer values, in which case (3) asserts that at least β of the literals are true. Naturally, we require that β be no greater than the number of nonzero components of a. We refer to (3) as a *β-clause*, or a *clause of degree β*; a *clause* is any β-clause. We refer to a clause in the traditional sense as a *1-clause.* (Kovács [17] has called (3) an *extended covering con-*

straint, and Balas and Jeroslow [1] refer to the halfspace it defines as a *canonical cut* for the unit cube.)

There are at least two benefits in introducing β-clauses. One is that a single β-clause can replace several 1-clauses and thereby represent information more compactly. For instance, $x_1 + x_2 + x_3 \geqslant 2$ can replace $x_1 + x_2 \geqslant 1, x_1 + x_3 \geqslant 1$, and $x_2 + x_3 \geqslant 1$. Another is that β-clauses are a natural way of expressing some thoughts. One may simply want to assert that at least β of a set of literals are true; or in a rule-based system, one may want to say of a rule, "if x_1 and ... and x_m, then y_1 and ... and y_β", that no more consequents are false than antecedents (which reduces to the assertion of an ordinary rule when there is one consequent). Such an assertion can be expressed with the β-clause, "$-x_1 - \ldots -x_m + y_1 + \ldots + y_\beta \geqslant \beta - m$."

There are various procedures for writing an arbitrary logical formula as a conjunction of 1-clauses; one may, for instance, use De Morgan's and distributive laws [19]. If the procedure adds no new variables, its running time and the length of the resulting conjunction are, in the worst case, exponential functions of the number of variables *in the given formula.* There are procedures that achieve linear time in the worst case by adding new variables [3,6], but there is usually no compelling reason to use them. Since a knowledge base is the conjunction of its formulae, one can put the entire knowledge base into conjunctive normal form simply by doing so for its individual formulae. The formulae are usually short, so that even exponential time is small. Also, formulae in many applications, notably knowledge-based systems, are typically production rules like, "if $\neg x_1$ and x_4 and x_5, then x_8", which is already essentially a 1-clause because it is readily rewritten, "$x_1 \vee \neg x_4 \vee \neg x_5 \vee x_8$".

We will say that a set S of inequalities *logically implies* another set T if any binary solution of S is a binary solution of T. If S and T are sets of 1-clauses, this corresponds to implication in the propositional calculus.

2. An integer programming model

Suppose we want to determine whether a clause $cx \geqslant \beta - n(c)$ logically follows from a set of m clauses. The set of clauses can be written as a system of linear inequalities $Ax \geqslant a$, where A is an $m \times n$ matrix, each row of which corresponds to a clause. If we formulate the integer program

$$\begin{aligned} &\text{minimize} && cx \\ &\text{subject to} && Ax \geqslant a \\ & && x_j \in \{0,1\}, \quad j = 1,\ldots,n, \end{aligned} \tag{4}$$

$Ax \geqslant a$ logically implies $cx \geqslant \beta - n(c)$ if and only if the minimum value of the objective function in (4) is at least $\beta - n(c)$.

Another test for implication is simply to use integer programming to check whether $Ax \geqslant a$, $cx \leqslant \beta - n(c) - 1$ has a feasible binary solution, but the model (4) has two advantages. The objective function in (4) provides a natural search direction, a perennial concern in resolution-based methods. Also in applications, $Ax \geqslant a$ will presumably represent a database known to be consistent. Thus, a feasible binary solution of $Ax \geqslant a$ can be stored and serve as a starting point for solving (4).

To check whether $Ax \geqslant a$ logically implies an arbitrary formula F, we can write $\neg F$ as a conjunction of clauses, let us say the clauses in the system $Bx \geqslant b$. Then there is implication if and only if the system $Ax \geqslant a$, $Bx \geqslant b$ has no feasible binary solution. If we know that $Ax \geqslant a$ is consistent, we can solve the problem by checking whether $Ax \geqslant a$, $x_0 + Bx \geqslant b$ logically implies x_0; if so, $Ax \geqslant a$ logically implies F. The absence of the artificial variable x_0 from $Ax \geqslant a$ exploits the fact that $Ax \geqslant a$ is consistent and directs the search in a natural way.

One way to solve (4) is to use cutting planes. We first solve the *linear programming relaxation* of (4), which is obtained by replacing $x_j \in \{0, 1\}$ with $0 \leqslant x_j \leqslant 1$. If the solution is integer, we are finished; otherwise, we add a *cut* (an inequality that any integer solution satisfies) that the current solution violates. We keep adding cuts and re-solving the resulting linear programs until an integer solution is achieved. One type of cut is a *Chvátal cut,* obtained by taking a positive linear combination of two or more constraints. The resulting coefficients and right-hand side are rounded up if non-integer, so as to produce a new constraint that the current solution may violate but that is clearly valid. Chvátal [5] proved that any valid inequality with integer coefficients and RHS can be obtained by repeating this procedure often enough, so that one can always solve (4) by adding the right Chvátal cuts.

3. Ordinary resolution and cutting planes

Robinson [24] showed that resolution, conceived as a refutation method, is *complete* for first-order predicate calculus and *a fortiori* complete for propositional calculus (i.e. for "ground clauses" in the predicate calculus). That is, he proved that a formula is inconsistent if and only if the resolution algorithm determines as much. (Resolution is used to check whether A logically implies B by testing $A \,\&\, \neg B$ for consistency.) A decade earlier, however, Quine [23,24] had proved a result that subsumes Robinson's when the latter is restricted to ground clauses. Quine proved that a procedure essentially the same as resolution, albeit not called resolution at the time, yields all of the "prime implications" of a 1-clause. We will work with Quine's resolution procedure, since it is more conducive to our purpose, and show presently that a refutation method follows as a special case. (Since Quine worked with disjunctive rather than conjunctive normal form, we consistently describe the dual of his procedures and results.)

Given two 1-clauses between which exactly one variable changes sign, such as,

$$\begin{aligned} x_1 + x_2 + x_3 \quad - x_5 - x_6 \quad &\geqslant 1 - 2 \\ -x_1 + x_2 \quad + x_4 - x_5 \quad - x_7 &\geqslant 1 - 3, \end{aligned} \tag{5}$$

their *resolvent* is obtained by cancelling the variable that changes sign and retaining on the left-hand side all of the other variables that appear in either clause:

$$x_2 + x_3 + x_4 - x_5 - x_6 - x_7 \geqslant 1 - 3.$$

The right-hand side of the inequality is adjusted to reflect the number of negative terms on the left-hand side. It is not difficult to see that the resolvent logically follows from the conjunction of the clauses resolved, but not from either clause individually.

We will say that a clause (of any degree) *dominates* another when the former logically implies the latter; that is, every Boolean vector satisfying the former satisfies the latter. Clearly, a 1-clause $bx \geqslant 1 - n(b)$ dominates another $cx \geqslant 1 - n(c)$ if and only if the former *absorbs* the latter; i.e. each b_j is either c_j or 0, which is to say that the terms occurring in $bx \geqslant 1 - n(b)$ form a subset of those occurring in $cx \geqslant 1 - n(c)$.

The *resolution algorithm* begins with a set of 1-clauses, which can be regarded as a formula in conjunctive normal form. It finds any pair that yield a resolvent that is dominated by no clause already in the set. The resolvent is added to the set and another pair found. The algorithm terminates when no more such pairs can be found. Sometimes a resolvent dominates one or more clauses already in the set, and these can be deleted so as to reduce the size of the set. In fact, Quine put forward his procedure as part of a method for simplifying formulae in normal form.

A set $Ax \geqslant a$ of 1-clauses is said to have a 1-clause C as a *prime implication* when $Ax \geqslant a$ logically implies C, but no other 1-clause that dominates C. Quine proved that if $Ax \geqslant a$ is consistent, the resolution algorithm generates all prime implications of $Ax \geqslant a$. Having carried out the algorithm, one can readily determine whether $Ax \geqslant a$ logically implies a given 1-clause C by checking whether any clause in $Ax \geqslant a$ or any clause so generated dominates C.

We can use the resolution algorithm to test for the consistency of $Ax \geqslant a$ by adding a new term x_{n+1} to every clause. The resulting system $Ax + ex_{n+1} \geqslant a$ (where e is a column vector of m ones) is logically consistent because it is satisfied by setting $x_{n+1} = 1$. The original system $Ax \geqslant a$ is logically inconsistent if and only if $Ax + ex_{n+1} \geqslant a$ logically implies x_{n+1}, and thus (by Quine's result), if and only if resolution applied to $Ax + ex_{n+1} \geqslant a$ yields the only 1-clause that dominates $x_{n+1} \geqslant 1$, $x_{n+1} \geqslant 1$ itself. We therefore see that the completeness of resolution as a refutation procedure follows from Quine's result.

The connection between resolution and cutting planes has been remarked elsewhere [8,26] and is as follows. Suppose that the two clauses (5) occur in the integer program (4). We can obtain a Chvátal cut by taking the following sum of the constraints in (5) and bounds of the form $0 \leqslant x_j \leqslant 1$:

$$\begin{array}{rcl}
x_1 + x_2 + x_3 \qquad - x_5 - x_6 \qquad & \geqslant & 1-2 \\
x_4 \qquad\qquad & \geqslant & 0 \\
- x_7 & \geqslant & -1 \\
-x_1 + x_2 \qquad + x_4 - x_5 \qquad - x_7 & \geqslant & 1-3 \\
x_3 \qquad\qquad\qquad & \geqslant & 0 \\
- x_6 \qquad & \geqslant & -1 \\
\hline
2x_2 + 2x_3 + 2x_4 - 2x_5 - 2x_6 - 2x_7 & \geqslant & 2-7.
\end{array} \tag{6}$$

Note that bounds are added so as to make all the coefficients on the left-hand side equal to 2. Dividing the result by 2 and rounding up the right-hand side, we obtain a Chvátal cut that is identical to the resolvent of the clauses in (5). In general, if $bx \geqslant 1 - n(b)$ and $cx \geqslant 1 - n(c)$ have a resolvent, it is the result of (a) adding these two clauses, $x_j \geqslant 0$ for all j for which $b_j + c_j = 1$, and $-x_j \geqslant -1$ for all j for which $b_j + c_j = -1$; and (b) dividing the sum by 2 and rounding up the RHS. Thus, resolvents form a special class of Chvátal cuts.

Quine's result applies here as well. $Ax \geqslant a$ logically implies $cx \geqslant 1 - n(c)$ if and only if the resolution algorithm eventually obtains from $Ax \geqslant a$ a resolvent that dominates $cx \geqslant 1 - n(c)$. When such a resolvent is added to the constraint set of (4), then obviously the minimum value of the objective function in the LP relaxation of (4) is at least $1 - n(c)$.

This means that we can determine whether $Ax \geqslant a$ logically implies $cx \geqslant 1 - n(c)$ by (partially) solving (4) with a cutting plane method that uses a restricted class of cuts: namely, Chvátal cuts that correspond to resolvents. We keep adding resolvents until (i) $cx + n(c) > 0$, in which case we have implication (since an integer solution would necessarily put $cx + n(c) \geqslant 1$); (ii) the solution is integer, in which case we have implication iff $cx + n(c) \geqslant 1$; or (iii) no further resolvents can be added, in which case we again have implication iff $cx + n(c) \geqslant 1$. This may not yield a solution of the integer program itself, because x may be noninteger when case (i) or even case (iii) halts the algorithm. As an example, suppose we want to determine whether $x_1 + x_2 + x_3 \geqslant 1$ follows from the clauses in the constraint set below:

$$\begin{array}{ll} \text{minimize} & x_1 + x_2 + x_3 \\ \text{subject to} & x_1 + x_2 \qquad \geqslant 1 \\ & x_1 \qquad + x_3 \geqslant 1 \\ & \qquad x_2 + x_3 \geqslant 1 \\ & x_1, x_2, x_3 \in \{0, 1\}. \end{array} \tag{7}$$

The minimum value of the objective function in the relaxed problem is 3/2, which already indicates that $x_1 + x_2 + x_3 \geqslant 1$ is logically implied. However, the solution is noninteger, even though no further resolvents can be added to the constraint set.

4. Domination between clauses

Recall that one clause *dominates* another if the former logically implies the latter; i.e. any Boolean vector satisfying the former satisfies the latter. The concept of an inequality being a "weakening" of another [15] is related to domination but differs from it. An inequality $cx \geqslant c_0$ is a *weakening* of $bx \geqslant b_0$ when $b_0 \geqslant c_0$ and $b_j \leqslant c_j$ for all j. Any $x \geqslant 0$ satisfying $bx \geqslant b_0$ satisfies any weakening of $bx \geqslant b_0$. This is in particular true of any binary x, so that a clause dominates all its weakenings. However, the converse is not true, since $x_1 - x_2 \geqslant 1$ dominates $-x_2 \geqslant 0$ even though the latter is not a weakening of the former.

It is useful to remark some properties of domination between clauses. Note first that $-x_1 + x_2 + x_3 + x_4 \geqslant 3 - 1$ dominates $x_3 + x_4 \geqslant 1$ because we can add the valid bounds $x_1 \geqslant 0$ and $-x_2 \geqslant -1$ to the former to get the latter. Thus, we can remove 2 literals from the former and obtain a clause it dominates, provided we reduce the degree by 2. We will say that the latter is a *reduction* of the former, and in general that $cx \geqslant \beta_2 - n(c)$ is a reduction of $bx \geqslant \beta_1 - n(b)$ when $cx \geqslant \beta_2 - n(c)$ is the result of adding $bx \geqslant \beta_1 - n(b)$ and zero or more clauses of the form $x_j \geqslant 0$ where $b_j = -1$, and of the form $-x_j \geqslant -1$ where $b_j = 1$; or, equivalently, when c can be obtained by setting exactly $\beta_1 - \beta_2$ nonzero components of b to zero ($\beta_1 \geqslant \beta_2$). A clause is a reduction of itself, and a clause clearly dominates all its reductions. We can also say that one β-clause $bx \geqslant \beta_1 - n(b)$ *absorbs* another $cx \geqslant \beta_2 - n(c)$ when $\beta_1 \geqslant \beta_2$ and $b_j = c_j$ or $b_j = 0$ for all j. Clearly, a clause dominates any clause it absorbs. To prove the following lemma, it is useful to adopt the notation:

$$x(S) = \sum_{j \in S} x_j,$$

where $S \subset \{1, \ldots, n\} = N$ is a set of indices.

LEMMA 1

Consider two clauses, which we may respectively write:

$$x(S) - x(T) + x(P_1) - x(N_1) \geqslant \beta_1 - |T| - |N_1|, \tag{8}$$

$$-x(S) + x(T) + x(P_2) - x(N_2) \geqslant \beta_2 - |S| - |N_2|, \tag{9}$$

where S, T, P_1, N_1 are pairwise disjoint, and similarly for S, T, P_2, N_2, for P_1, N_2, and for P_2, N_1. Then the following are equivalent:

(a) $\beta_1 - \beta_2 \geqslant |S| + |T| + |P_1 \backslash P_2| + |N_1 \backslash N_2|$;

(b) some reduction of (8) absorbs (9);

(c) (8) dominates (9).

(See Kovács [17] for a related result.)

Proof

To see that (a) implies (b), consider the clause

$$x(P_1 \cap P_2) - x(N_1 \cap N_2)$$
$$\geqslant (\beta_1 - |S \cup T| - |P_1 \backslash P_2| - |N_1 \backslash N_2|) - |N_1 \cap N_2|. \tag{10}$$

Clearly, (10) is a reduction of (8); but (a) implies that (10) absorbs (9), and (b) follows.

Obviously, (b) implies (c).

To show that (c) implies (a), suppose that (a) is false. It suffices to exhibit a binary $x^\star$ that satisfies (8) and violates (9). We can define $x^\star$ so that:

$$x^\star(S) = |S|,\ \ x^\star(T) = 0,\ \ x^\star(P_1 \backslash P_2) = 0,\ \ x^\star(N_1 \backslash N_2) = |N_1 \backslash N_2|.$$

If we let $\delta = |S| + |T| + |P_1 \backslash P_2| + |N_1 \backslash N_2|$, then since (8) contains at least β_1 terms, we can futher define $x^\star$ so that

$$x^\star(P_1 \cap P_2) + [|N_1 \cap N_2| - x^\star(N_1 \cap N_2)] = \max\{0, \beta_1 - \delta\}, \tag{11}$$

and $x^\star$ satisfies (8). However, since (a) is false, we have $\beta_1 - \delta < \beta_2$, so that

$$-x^\star(S) + x^\star(T) + \beta_1 - \delta < \beta_2 - |S|. \tag{12}$$

Equations (11) and (12) yield that

$$-x^*(S) + x^*(T) + x^*(P_1 \cap P_2) - x^*(N_1 \cap N_2) \geqslant \beta_2 - |S| - |N_1 \cap N_2|. \quad (13)$$

We are still free to define x^* so that $x^*(P_2 \backslash P_1) = 0$ and $x^*(N_2 \backslash N_1) = |N_2 \backslash N_1|$. Adding these equations to (13), we obtain the denial of (9). □

5. Generalized resolution by cancellation

We will see that generalized resolution requires two types of sums, a sum that involves cancellation of terms and a sum of clauses showing a circulant pattern. We deal with cancellation in this section, and circulant sums in the next.

We require that generalized resolution be *complete* in that it generate all prime implications of a consistent set of clauses to which it is applied. Here, a clause C is a *prime implication* of a system $Ax \geqslant a$ of clauses if $Ax \geqslant a$ logically implies C but no other clause that dominates C.

If the resolvent of two 1-clauses is a Chvátal cut obtained by adding the clauses and certain bounds, one may expect to obtain a useful resolution of β-clauses by taking a similar sum. We will pursue this idea so as to generalize resolution by cancellation.

Let us now try taking a sum analogous to (6). We let (8) and (9) represent any two clauses we wish to resolve, and we add them to the bounds

$$\begin{aligned} x_j &\geqslant 0, \quad \text{all } j \in P_1 \otimes P_2, \\ -x_j &\geqslant -1, \quad \text{all } j \in N_1 \otimes N_2, \end{aligned}$$

where $\otimes$ indicates symmetric differences. The sum divided by 2 with the RHS rounded up (if necessary) is:

$$x(P_1 \cup P_2) - x(N_1 \cup N_2) \geqslant \lceil(\beta_1 + \beta_2 - |S \cup T|)/2\rceil - |N_1 \cup N_2|, \quad (14)$$

where $\lceil\alpha\rceil$ indicates the smallest integer greater than or equal to α. We want neither (8) nor (9) to dominate (14), since otherwise we have achieved nothing. However, note that (8) and (9) contain exactly $|S \cup T|$ terms that do not occur in (14). Thus, by lemma 1, (8) dominates (14) iff

$$\beta_1 - |S \cup T| \geqslant \lceil(\beta_1 + \beta_2 - |S \cup T|)/2\rceil. \quad (15)$$

We can suppose, without loss of generality, that $\beta_1 \geqslant \beta_2$, so that (8) dominates (14) if either (8) or (9) does. We achieve nothing, then, if (15) holds, but if $\beta_1 + \beta_2 - |S \cup T|$ is even, (15) is equivalent to $\beta_1 - \beta_2 \geqslant |S \cup T|$. If $\beta_1 + \beta_2 - |S \cup T|$ is odd, (15) is equivalent to $\beta_1 - \beta_2 \geqslant |S \cup T| + 1$, which is again equivalent to $\beta_1 - \beta_2 \geqslant |S \cup T|$, since we cannot have $\beta_1 - \beta_2 = |S \cup T|$ when $\beta_1 + \beta_2 - |S \cup T|$ is odd. We conclude that (14) should be regarded as a resolvent of (8) and (9) only when $\beta_1 - \beta_2 < |S \cup T|$ and (of course) $\beta_1 + \beta_2 - |S \cup T| \geqslant 1$.

We will see, however, that resolution when $\beta_1 \neq \beta_2$ is redundant of a second (*indirect*) kind of resolution we must introduce momentarily. Let us say, therefore, that (14) is a *direct cancellation resolvent* of (8) and (9) when $\beta_1 = \beta_2$ and $\beta_1 + \beta_2 > |S \cup T| \geqslant 1$. We have proved:

LEMMA 2

The direct cancellation resolvent of two clauses is logically implied by the conjunction of the clauses, but is dominated by neither.

It will turn out that generalized resolution is still complete even if we directly resolve (8) and (9) only when $|S \cup T| = 1$; we may, however, reduce the number of resolution steps by resolving when $|S \cup T| > 1$ as well.

Direct cancellation resolution is not by itself enough. Consider, for example, the clauses:

$$x_1 + x_2 + x_3 + x_4 + x_5 \geqslant 4, \tag{16}$$

$$-x_1 - x_2 + x_3 \geqslant 2 - 2. \tag{17}$$

Since $\beta_1 - \beta_2 = |S \cup T|$, no cancellation of the types discussed so far (even the type that permits $\beta_1 \neq \beta_2$) is possible. However, one implication, in fact a prime implication, of (16) and (17) is $x_3 \geqslant 1$, and it is dominated by neither (16) nor (17). We note, however, that one reduction of (16) is $x_1 + x_2 + x_3 \geqslant 2$. This and (17) have $x_3 \geqslant 1$ as their direct cancellation resolvent. We will say that $x_3 \geqslant 1$ is an *indirect* cancellation resolvent of (16) and (17).

In general, an *indirect cancellation resolvent* of (8) and (9) is the direct cancellation resolvent (if any) of any two clauses of the form

$$x(S') - x(T') + x(P_1') - x(N_1') \geqslant \gamma - |T'| - |N_1'|, \tag{18}$$

$$-x(S') + x(T') + x(P_2') - x(N_2') \geqslant \gamma - |S'| - |N_2'|, \tag{19}$$

where (18) is a reduction of (8), (19) is a reduction of (9), $S' \subset S$, $T' \subset T$, and $P_i \subset P_i'$ and $N_i \subset N_i'$ for $i = 1, 2$. We also require that $P_1' \cap P_2' = P_1 \cap P_2$ and $N_1' \cap N_2' = N_1 \cap N_2$. When (18) and (19) are, respectively, identical to (8) and (9), the resolution is of course direct.

It is not difficult to see that any cancellation resolvent is a Chvátal cut. If $bx \geqslant \beta_1 - n(b)$ and $cx \geqslant \beta_2 - n(c)$ have $dx \geqslant \gamma - n(d)$ as a cancellation resolvent, then $dx \geqslant \gamma - n(d)$ is the result of (a) adding $bx \geqslant \beta_1 - n(b)$, $cx \geqslant \beta_2 - n(c)$, $x_j \geqslant 0$ for all j for which $b_j + c_j < 2d_j$ and $-x_j \geqslant -1$ for all j for which $b_j + c_j > 2d_j$; and (b) dividing the sum by 2 and rounding up the RHS.

THEOREM 1

A cancellation resolvent (direct or indirect) of two clauses is logically implied by the conjunction of the clauses, but dominated by neither.

Proof

The resolvent is implied by the conjunction of the clauses because it is a Chvátal cut derived from them and bounds from $0 \leqslant x \leqslant 1$. To show that it is dominated by neither, we note that a resolvent has the form

$$x(P_1' \cup P_2') - x(N_1' \cup N_2') \geqslant \gamma - \lfloor |S' \cup T'|/2 \rfloor - |N_1' \cup N_2'|, \qquad (20)$$

where $\lfloor \alpha \rfloor$ is the greatest integer less than or equal to α. Suppose contrary to the claim that, say, (8) dominates (20). Then by lemma 1, some reduction of (8) absorbs (20). It is clear that any reduction of (8) that absorbs (20) must absorb

$$x(P_1') - x(N_1') \geqslant \gamma - \lfloor |S' \cup T'|/2 \rfloor - |N_1'|. \qquad (21)$$

However, a reduction of (8) absorbs (21) only if

$$\gamma - \lfloor (|S' \cup T'|/2) \rfloor \leqslant \beta_1 - |S \cup T| - |P_1 \backslash P_1'| - |N_1 \backslash N_1'| = \gamma - |S' \cup T'|,$$

where the equality is due to the fact that (18) is a reduction of (8). This is, however, impossible because $|S' \cup T'| \geqslant 1$. Thus, (8) cannot dominate (21) and therefore cannot dominate any clause that (21) absorbs, which means that (8) cannot dominate (20). By similar argument, (9) cannot dominate (20), either. □

6. Generalized resolution by circulant summation

Cancellation resolution alone is not enough to generate all prime implications, as the three clauses of (7) demonstrate. One (prime) implication of these clauses is $x_1 + x_2 + x_3 \geqslant 2$, which none of the clauses dominates. We can obtain $x_1 + x_2 + x_3 \geqslant 2$, however, by summing the three inequalities, dividing the sum by 2, and rounding up the RHS. The result, which we call a *circulant resolvent,* is logically implied by the conjunction of the clauses, but is dominated by none. It is clear that this sort of Chvátal cut can be obtained whenever one adds clauses exhibiting a circulant pattern like that in (7). Since (4) is related to set covering and set packing problems, it is not surprising that similar circulant patterns have played a role in their investigation [10,20,21].

We can also obtain a circulant resolvent when two or more of the clauses in the circulant pattern are implicit in a single clause, i.e. if we start with clauses

$$\begin{aligned} x_1 + x_2 \quad & \geqslant 1 \\ x_3 & \geqslant 1. \end{aligned} \tag{22}$$

The clause $x_3 \geqslant 1$ absorbs the last two clauses of (7), so that they are implicitly present in (22). We can therefore derive the resolvent $x_1 + x_2 + x_3 \geqslant 2$, which neither clause of (22) dominates.

We need one further extension of circulant resolution, as the following example makes clear:

$$x_1 + x_2 - x_4 - x_5 \geqslant 3 - 2\,, \tag{23}$$

$$x_1 + x_3 - x_4 \geqslant 2 - 1\,, \tag{24}$$

$$-x_1 + x_2 + x_3 - x_4 - x_5 \geqslant 4 - 3\,. \tag{25}$$

The clause $x_1 + x_2 + x_3 \geqslant 2$ is a prime implication of (23)–(25), and we have provided no way to obtain it by resolution. To remedy this, we combine the reduction $x_1 + x_2 \geqslant 1$ of (23), the reduction $x_1 + x_3 \geqslant 1$ of (24), and the reduction $x_2 + x_3 \geqslant 1$ of (25) to produce the circulant resolvent $x_1 + x_2 + x_3 + x_4 + x_5 \geqslant 2$. A cancellation resolvent of this with (23), (24) or (25) is $x_1 + x_2 + x_3 + x_4 \geqslant 2$, whose cancellation resolvent with (23), (24) or (25) is the desired $x_1 + x_2 + x_3 \geqslant 2$.

In the above example, we do not derive $x_1 + x_2 + x_3 \geqslant 2$ directly as a circulant resolvent, because we require that the terms removed to effect the reductions have signs opposite those of terms in the resolvent. This simplifies the search for resolvents without sacrificing the completeness of generalized resolution.

For an index set $J \subset N$, define c_J by $(c_J)_j = c_j$ for $j \in J$ and $(c_J)_j = 0$ for $j \in N \backslash J$. Then a clause $cx \geqslant (\gamma + 1) - n(c)$ is a *circulant resolvent on* J of a set S of clauses $a^i x \geqslant \beta_i - n(a^i)$, $i \in I$, if each $a^i x \geqslant \beta_i - n(a^i)$ has a reduction $b^i x \geqslant \gamma - n(b^i)$ that absorbs $c_J x \geqslant \gamma - n(c_J)$, such that for all j, (a) $b^i_j = 0$ for some $i \in I$, and (b) $a^i_j = b^i_j$ or $a^i_j - b^i_j = -c_j$ for all $i \in I$.

We can show that the circulant resolvent $cx \geqslant (\gamma + 1) - n(c)$ is a Chvátal cut. For each $k \in J$, pick an $i \in I$ for which $b^i_k = 0$ and consider the sum C_k of (a) $a^i x \geqslant \beta_i - n(a^i)$; (b) $x_j \geqslant 0$ for all j for which $a^i_j - b^i_j = -1$, and $-x_j \geqslant -1$ for all j for which $a^i_j - b^i_j = 1$; and (c) $x_j \geqslant 0$ for all $j \in J\backslash\{k\}$ for which $c_j - b^i_j = 1$, and $-x_j \geqslant -1$ for all $j \in J\backslash\{k\}$ for which $c_j - b^i_j = -1$. The sum of C_k over all $k \in J$, $(|J| - 1)x_j \geqslant 0$ for all $j \in N\backslash J$ for which $c_j = 1$, and $-(|J| - 1)x_j \geqslant -1$ for all $j \in N\backslash J$ for which $c_j = -1$ is $(|J| - 1)cx \geqslant |J|\gamma - (|J| - 1)n(c)$. Dividing by $|J| - 1$ and rounding up the RHS, we get the resolvent $cx \geqslant (\gamma + 1) - n(c)$, which is therefore a Chvátal cut.

THEOREM 2

A circulant resolvent is logically implied by the conjunction of the clauses resolved, but is dominated by none of them.

Proof

A circulant resolvent $cx \geqslant (\gamma + 1) - n(c)$ is logically implied by the set S of clauses resolved because it is a Chvátal cut derived from S and bounds in $0 \leqslant x \leqslant 1$. To show that no clause in S dominates $cx \geqslant (\gamma + 1) - n(c)$, we note that any $(\gamma + 1)$-clause that is a reduction of a clause in S must contain at least one variable x_j whose coefficient has a sign opposite that of c_j. Thus, no reduction of a clause in S can absorb $cx \geqslant (\gamma + 1) - n(c)$, which means by lemma 1 that no clause of S can dominate $cx \geqslant (\gamma + 1) - n(c)$. □

7. A generalized resolution algorithm

In this section, we state a generalized resolution algorithm that generates all prime implications of a consistent set S of logical formulae. We must convert any formula of S that is not a clause to a conjunction of 1-clauses using any convenient algorithm. S thus becomes a system $Ax \geqslant a$.

We can use the generalized resolution algorithm to check whether $Ax \geqslant a$ logically implies a logical formula C. If C is not a clause, we convert it to a conjunction of 1-clauses, and we separately check whether each is logically implied. To check whether $Ax \geqslant a$ implies a clause $cx \geqslant \beta - n(c)$, we let the algorithm proceed until it generates a clause that dominates $cx \geqslant \beta - n(c)$. If it has not done so at termination, $Ax \geqslant a$ does not logically imply $cx \geqslant \beta - n(c)$.

Step 0. Let $Ax \geqslant a$ be the original system of clauses, and delete all clauses that are dominated by others.

Step 1. If some clause in $Ax \geqslant a$ dominates $cx \geqslant \beta - n(c)$, stop; the original system implies $cx \geqslant \beta - n(c)$.

Step 2. If possible, find a pair of clauses of $Ax \geqslant a$ having at least one cancellation resolvent that is not already dominated by clauses in $Ax \geqslant a$. Delete from $Ax \geqslant a$ all clauses dominated by the undominated cancellation resolvents of this pair, and augment $Ax \geqslant a$ by the resolvents.

Step 3. If possible, find a set S of clauses of $Ax \geqslant a$ that has a circulant resolvent that is not already dominated by a clause in $Ax \geqslant a$. Delete from $Ax \geqslant a$ all clauses dominated by this resolvent, and augment $Ax \geqslant a$ by the resolvent.

Step 4. If no resolvents were added to $Ax \geqslant a$ in steps 2 or 3, stop; the original system does not imply $cx \geqslant \beta - n(c)$. Otherwise, go to step 1.

The above procedure is clearly finite, because any resolvent generated must have degree n or less, and there are finitely many such clauses.

As an example, suppose we wish to generate all the prime implications of the system below. For brevity, we indicate only the matrix A on the left-hand side and the vector a on the right-hand side.

$$\begin{array}{rrrrrrrrr}
1 & -1 & 1 & 1 & 1 & 0 & 3 & -1 & (26)\\
1 & -1 & -1 & 0 & 0 & 1 & 2 & -2 & (27)\\
1 & 0 & 1 & 0 & 0 & 0 & 1 & & (28)\\
-1 & -1 & 1 & 0 & 0 & 0 & 2 & -2 & (29)\\
1 & -1 & 0 & 1 & 0 & 0 & 2 & -1 & (30)\\
1 & 0 & 1 & 1 & 0 & 0 & 2 & & (31)\\
0 & -1 & 1 & 1 & 0 & 0 & 2 & -1\,. & (32)
\end{array}$$

We can apply cancellation resolution to (26) and (27) to obtain the following resolvents, which we add to the system:

$$\begin{array}{rrrrrrrrr}
1 & -1 & 0 & 1 & 0 & 1 & 2 & -1 & (33)\\
1 & -1 & 0 & 0 & 1 & 1 & 2 & -1 & (34)\\
1 & -1 & 0 & 0 & 0 & 0 & 1 & -1\,. & (35)
\end{array}$$

We can derive from (35), (28) and (29) a circulant resolvent on $J = \{1,2,3\}$,

$$\begin{array}{rrrrrrrr}
1 & -1 & 1 & 0 & 0 & 0 & 2 & -1\,.
\end{array} \tag{36}$$

We delete (35) and (28) because (36) dominates them. We resolve (27) and (29) to get:

$$\begin{array}{rrrrrrrr}
0 & -1 & 0 & 0 & 0 & 1 & 1 & -1\,.
\end{array} \tag{37}$$

We delete clauses (36) and (30) – (32) and substitute their circulant resolvent on $J = \{1,2,3,4\}$,

$$\begin{array}{rrrrrrrr}
1 & -1 & 1 & 1 & 0 & 0 & 3 & -1\,.
\end{array} \tag{38}$$

We delete (26) and (33) because (38) dominates them. We delete (29) in favor of its one undominated cancellation resolvent with (38),

$$\begin{array}{rrrrrrrr}
0 & -1 & 1 & 0 & 0 & 0 & 2 & -1\,,
\end{array} \tag{39}$$

and we delete (37) because (39) dominates it. Finally, we replace (27) with its resolvent with (38) or (39),

$$1 \quad -1 \quad 0 \quad 0 \quad 0 \quad 1 \qquad 2-1. \tag{40}$$

Removing (34) because (40) dominates it, we obtain the following system:

$$1 \quad -1 \quad 1 \quad 1 \quad 0 \quad 0 \qquad 3-1 \tag{38}$$
$$0 \quad -1 \quad 1 \quad 0 \quad 0 \quad 0 \qquad 2-1 \tag{39}$$
$$1 \quad -1 \quad 0 \quad 0 \quad 0 \quad 1 \qquad 2-1. \tag{40}$$

Since there are no further resolvents, these are the prime implications of (26)–(32), and they are logically equivalent to (26)–(32).

We can use the above algorithm to test for the consistency of a system $Ax \geqslant a$. If β is the largest degree of the clauses in $Ax \geqslant a$, we form the system:

$$Ax + Ex' \geqslant a, \tag{41}$$

where E is an $m \times \beta$ matrix of ones, and $x' = (x_{n+1}, \ldots, x_{n+\beta})$. The system (41) is consistent because setting $x' = (1, \ldots, 1)$ yields a binary solution (for any x). $Ax \geqslant a$ is inconsitent iff some component of x' is nonzero in any binary solution of (41). Thus, $Ax \geqslant a$ is inconsistent iff some prime implication of (41) dominates $x_{n+1} + \ldots + x_{n+\beta} \geqslant 1$.

8. Proof of completeness

The proof that generalized resolution is complete begins with two lemmas whose proof is an extension of Quine's original proof [22,23].

LEMMA 3

Let $dx \geqslant \beta - n(d)$ be a longest β-clause logically implied by a consistent system $Ax \geqslant a$ but dominated by no clause in $Ax \geqslant a$, and suppose that $d_k = 0$ for some k. Then one can derive from clauses in $Ax \geqslant a$ a cancellation resolvent that dominates $dx \geqslant \beta - n(d)$.

Proof

Since $dx \geqslant \beta - n(d)$ is a longer β-clause logically implied by $Ax \geqslant a$ but dominated by no clause in $Ax \geqslant a$, the longer β-clauses $x_k + dx \geqslant \beta - n(d)$ and $-x_k + dx \geqslant \beta - n(d) - 1$, which $Ax \geqslant a$ logically implies, must be dominated by clauses in $Ax \geqslant a$. We will show that these dominating clauses yield a cancellation resolvent that dominates $dx \geqslant \beta - n(d)$.

Any clause dominating $x_k + dx \geqslant \beta - n(d)$ must contain x_k, else it would dominate $dx \geqslant \beta - n(d)$, and similarly, any clause dominating $-x_k + dx \geqslant \beta - n(d) - 1$ must contain $-x_k$. Thus, if we write $dx \geqslant \beta - n(d)$ as

$$x(P) - x(N) \geqslant \beta - |N|, \tag{42}$$

where $x_k \notin P, N$, then we may write the two dominating clauses as, respectively,

$$x_k + x(P_1) - x(N_1) \geqslant \beta_1 - |N_1|, \tag{43}$$

$$-x_k + x(P_2) - x(N_2) \geqslant \beta_2 - |N_2| - 1. \tag{44}$$

Since (43) dominates $x_k + dx \geqslant \beta - n(d)$, lemma 1 implies $\beta_1 - |P_1 \backslash P| - |N_1 \backslash N| \geqslant \beta$. However, since (43) does not dominate (42), we have $\beta_1 - |P_1 \backslash P| - |N_1 \backslash N| - 1 < \beta$, so that $\beta_1 - |P_1 \backslash P| - |N_1 \backslash N| = \beta$. Similarly, $\beta_2 - |P_2 \backslash P| - |N_2 \backslash N| = \beta$. Thus, the following are reductions of (43) and (44), respectively:

$$x_k + x(P_1 \cap P) - x(N_1 \cap N) \geqslant \beta - |N_1 \cap N|, \tag{45}$$

$$-x_k + x(P_2 \cap P) - x(N_2 \cap N) \geqslant \beta - |N_2 \cap N| - 1. \tag{46}$$

The following clause is the direct cancellation resolvent of (45) and (46) and therefore a cancellation resolvent of (43) and (44):

$$x((P_1 \cup P_2) \cap P) - x((N_1 \cup N_2) \cap N) \geqslant \beta - |(N_1 \cup N_2) \cap N|. \tag{47}$$

However, (47) absorbs and therefore dominates (42). □

The next lemma will serve as the initial step of an inductive proof of the main theorem. Let us say that a consistent system $Ax \geqslant a$ is *complete* with respect to cancellation resolution if all cancellation resolvents of clauses in $Ax \geqslant a$ are dominated by clauses in $Ax \geqslant a$.

LEMMA 4

A consistent system that is complete with respect to cancellation resolution contains a dominating clause for any given 1-clause logically implied by the system.

Proof

Let $Ax \geqslant a$ be complete with respect to cancellation resolution. Suppose, contrary to the claim, that $Ax \geqslant a$ logically implies some 1-clause that no clause of $Ax \geqslant a$ dominates. We will derive a contradiction.

Let $dx \geqslant 1 - n(d)$ be the longest 1-clause that $Ax \geqslant a$ logically implies, but no clause of $Ax \geqslant a$ dominates. We claim that $d_k = 0$ for some k. For suppose otherwise. The unique binary value x^* of x that violates $dx \geqslant 1 - n(d)$ is given by $x_j^* = 0$ if $d_j = 1$ and $x_j^* = 1$ if $d_j = -1$. Since $Ax \geqslant a$ logically implies $dx \geqslant 1 - n(d)$, x^*

must violate $Ax \geqslant a$ and therefore some clause $bx \geqslant \beta - n(b)$ of $Ax \geqslant a$. This means that we can have $b_j = -d_j \neq 0$ for at most $\beta - 1$ j's. Thus, by lemma 1 $bx \geqslant \beta - n(b)$ dominates $b'x \geqslant 1 - n(b')$, where $b = b'$ except that $b'_j = 0$ whenever $b_j = -d_j \neq 0$. Since $b'x \geqslant 1 - n(b')$ absorbs and therefore dominates $dx \geqslant 1 - n(d)$, we have that $bx \geqslant \beta - n(b)$ dominates $dx \geqslant 1 - n(d)$, which is contrary to the hypothesis that no clause of $Ax \geqslant a$ dominates $dx \geqslant 1 - n(d)$.

Since $d_k = 0$, we conclude by lemma 3 (setting $\beta = 1$) that we can derive a cancellation resolvent that dominates $dx \geqslant 1 - n(d)$. However, since $Ax \geqslant a$ is complete, some clause of $Ax \geqslant a$ dominates this resolvent, which means that some clause dominates $dx \geqslant 1 - n(d)$, a contradiction. □

Since testing for consistency involves checking whether a certain 1-clause is logically implied by a certain consistent system, we have the immediate corollary.

COROLLARY 1

Cancellation resolution alone suffices to check a system for consistency.

Let us say that a system is complete with respect to generalized resolution if all generalized resolvents of clauses in the system are dominated by clauses in the system. The main theorem below says that the algorithm of the previous section delivers all prime implications.

THEOREM 3

A consistent system that is complete with respect to generalized resolution contains all of its prime implications.

Proof

Let $Ax \geqslant a$ be a system that is complete with respect to generalized resolution. We will prove by induction on γ that any given γ-clause that $Ax \geqslant a$ logically implies is dominated by some clause in $Ax \geqslant a$. It follows that $Ax \geqslant a$ contains all of its prime implications.

By lemma 4, the claim is true for $\gamma = 1$.

We therefore assume the claim is true for γ and show it is true for $\gamma + 1$. Suppose to the contrary that $Ax \geqslant a$ logically implies some $(\gamma + 1)$-clause that no clause of $Ax \geqslant a$ dominates. We will derive a contradiction.

Let $cx \geqslant (\gamma + 1) - n(c)$ be a longest $(\gamma + 1)$-clause logically implied by $Ax \geqslant a$, but dominated by no clause in $Ax \geqslant a$. There are two cases.

Case I: $c_k = 0$ for some k. By lemma 3 we can obtain a resolvent that dominates $cx \geqslant (\gamma + 1) - n(c)$. By completeness, some clause of $Ax \geqslant a$ dominates the resolvent and therefore $cx \geqslant (\gamma + 1) - n(c)$, a contradiction.

Case II: $c_k \neq 0$ for all k. We have by lemma 1 that for each k, $Ax \geqslant a$ logically implies $c^k x \geqslant \gamma - n(c^k)$, where $c^k = c$ except that $c_k^k = 0$. Thus, by the inductive hypothesis, some clause of $Ax \geqslant a$ dominates $c^k x \geqslant \gamma - n(c^k)$.

For each $k \in N$, let $a^k x \geqslant \beta_k - n(a^k)$ be a clause in $Ax \geqslant a$ dominating $c^k \geqslant \gamma - n(c^k)$. By lemma 1, $a^k x \geqslant \beta_k - n(a^k)$ contains at most $\beta_k - \gamma$ terms absent from $c^k x \geqslant \gamma - n(c^k)$. However, since $a^k x \geqslant \beta_k - n(a^k)$ does not dominate $cx \geqslant (\gamma + 1) - n(c)$, it contains at least $\beta_k - \gamma$ terms absent from $cx \geqslant (\gamma + 1) - n(c)$ and therefore from $c^k x \geqslant \gamma - n(c^k)$. Thus, $a^k x \geqslant \beta_k - n(a^k)$ contains exactly $\beta_k - \gamma$ terms absent from $c^k x \geqslant \gamma - n(c^k)$. So if we let $b^k x \geqslant \gamma - n(b^k)$ contain the terms present in both $a^k x \geqslant \beta_k - n(a^k)$ and $c^k x \geqslant \gamma - n(c^k)$, it is a reduction of $a^k x \geqslant \beta_k - n(a^k)$.

We now show that $cx \geqslant (\gamma + 1) - n(c)$ is a circulant resolvent on $J = N$ of the clauses $a^k x \geqslant \beta_k - n(a^k)$, $k \in N$. It suffices to show that (a) $b_k^k = 0$ for all $k \in N$, and (b) every term $a_j x_j$ absent from $b^k x \geqslant \gamma - n(b^k)$ has a sign opposite that of $c_j x_j$. (a) is true by definition of b^k. Since each $c_j \neq 0$, (b) is clearly true when $j \neq k$. To see that $a_k^k = -c_k$, note that otherwise the clause $c_k x_k + b^k x \geqslant (\gamma + 1) - n(c_k) - n(b^k)$, which absorbs $cx \geqslant (\gamma + 1) - n(c)$, would be a reduction of $a^k x \geqslant \beta_k - n(a^k)$, which would therefore dominate $cx \geqslant (\gamma + 1) - n(c)$, contrary to hypothesis.

However, if $cx \geqslant (\gamma + 1) - n(c)$ is a circulant resolvent, it is dominated by some clause in the complete system $Ax \geqslant a$, contrary to hypothesis. □

The following corollary is evident in the previous three proofs.

COROLLARY 2

Generalized resolution is complete even when direct cancellation resolution is restricted to clauses between which exactly one variable changes sign.

9. The separation problem

A major task of a cutting plane algorithm is not just to find cuts, but to find *separating* cuts, or cuts that are violated by the current solution of the linear programming relaxation. Here, we prove some easily checked conditions that the parents of a separating resolution cut must satisfy. These conditions can speed the search for separating cuts.

For a given solution x of the LP relaxation of (4), define the *truth value* $v(t)$ of a literal t by $v(x_j) = x_j$ and $v(-x_j) = 1 - x_j$. Also, the truth value of a sum of literals is the sum of their truth values, and the truth value of a clause is the truth value of its left-hand side.

LEMMA 5

A cancellation resolvent on s variables can be separating only if the following conditions are satisfied:

(a) s is odd;
(b) each parent has a truth value strictly less than $\beta + (s+1)/2$, where β is the degree of the parent;
(c) the sum of the truth values of the variables on which the resolution takes place lies strictly between $(s-1)/2$ and $(s+1)/2$;
(d) each parent contains at least one variable with a fractional truth value.

Proof

To show (a), let the current solution x of (4) violate cancellation resolvent (20) of (8) and (9). Then $v(x(P_1' \cup P_2') - x(N_1' \cup N_2')) < \gamma - \sigma$, where $\sigma = \lfloor |S' \cup T'|/2 \rfloor = \lfloor s/2 \rfloor$. Thus, $v(x(P_1') - x(N_1')) < \gamma - \sigma$, and since x satisfies (18), we have $v(x(S') - x(T')) > \gamma - (\gamma - \sigma) = \sigma$. Similarly, since x satisfies (19), we get $v(x(T') - x(S')) > \sigma$, or $s - v(x(S') - x(T')) > \sigma$. Thus, $\sigma < v(x(S') - x(T')) < s - \sigma$, which implies that $2\sigma < s$. It follows that s is odd.

To show (b), observe that since s is odd, $\sigma = (s-1)/2$. Also, since $v(x(P_1') - x(N_1')) < \gamma - \sigma = \gamma - (s-1)/2$, the truth value τ of (8) is strictly less than $\gamma - (s-1)/2 + |S \cup T| + |P_1 \backslash P_1'| + |N_1 \backslash N_1'|$. However, since (18) is a reduction of (8), $|S \backslash S'| + |T \backslash T'| + |P_1 \backslash P_1'| + |N_1 \backslash N_1'| = \beta_1 - \gamma$, whence $\tau < \beta_1 - (s-1)/2 + |S' \cup T'| = \beta_1 + (s+1)/2$.

Finally, (c) follows from the facts that $\sigma < v(x(S') - x(T')) < s - \sigma$ and $\sigma = (s-1)/2$, and (d) follows immediately from (c). □

LEMMA 6

A circulant resolvent $cx \geqslant (\gamma + 1) - n(c)$ on J is separating only if $v(c_J x) < \gamma + 1$.

Proof

If the current solution x violates $cx \geqslant (\gamma+1) - n(c)$, then $v(cx) < \gamma + 1$, which implies $v(c_J x) < \gamma + 1$. □

10. A cutting plane algorithm

We can now generalize the simple cutting plane algorithm described in sect. 2. The object is to determine whether $Ax \geqslant a$ logically implies $cx \geqslant \beta - n(c)$. Recall that a knowledge base can be put into the form $Ax \geqslant a$ by converting each formula that is not a conjunction of clauses to a conjunction of 1-clauses. It may be worthwhile to apply the above generalized resolution algorithm to the conjunction of clauses representing each formula so as to obtain a more compact representation.

Step 0: Let $Ax \geqslant b$ be the original system, and delete all clauses dominated by other clauses.

Step 1: If $cx \geqslant \beta - n(c)$ is dominated by any clause of $Ax \geqslant a$, stop; the original system implies $cx \geqslant \beta - n(c)$.

Step 2: Solve the linear programming relaxation of (4), and let $x^\star$ be a solution. If $cx^\star + n(c) > \beta - 1$, stop; the original system implies $cx \geqslant \beta - n(c)$ (since any integer solution will put $cx + n(c) \geqslant \beta$).

Step 3: If $x^\star$ is integer, stop; the original system does not imply $cx \geqslant \beta - n(c)$.

Step 4: Generate the cancellation resolvents of $Ax \geqslant a$ that $x^\star$ violates. Do this by considering all pairs of clauses of $Ax \geqslant a$ that satisfy lemma 5 and generating all separating resolvents of which they are parents. Delete from $Ax \geqslant a$ all clauses the resolvents dominate, and add the resolvents to $Ax \geqslant a$.

Step 5: Generate circulant resolvents of $Ax \geqslant a$ that $x^\star$ violates. Do this by considering all possible resolvents that satisfy lemma 6 and that have K or fewer literals, where K is fixed beforehand. For each such resolvent C, search for a set of clauses and a set of K variables such that those clauses yield C when resolved on those variables. Delete from $Ax \geqslant a$ all clauses dominated by resolvents so obtained, and add the resolvents to $Ax \geqslant a$.

Step 6: If at least one separating resolvent was generated in steps 4 or 5, go to step 1. If not, generate some non-separating generalized resolvents of $Ax \geqslant a$. If such resolvents are found, delete from $Ax \geqslant a$ all clauses they dominate, add the resolvents to $Ax \geqslant a$, and go to step 1. Otherwise, stop; the original system does not imply $cx \geqslant \beta - n(c)$ (if $\beta \leqslant K$).

Although the non-separating cuts generates in step 6 do not force a new solution in step 2, the *next* execution of step 6 may find separating cuts. The details are omitted, and in practice one may wish to look for other sorts of cuts or revert to branch-and-bound.

It may be useful to keep track of the cuts whose generation involves exclusively the constraints in the original database (such as the "inference rules" of an expert system) and involve no constraints added for the purpose of the current inference (input data). These cuts can be made a permanent part of the knowledge base, and their presence may speed the solution of the next problem.

The separation algorithm in step 4 has polynomial complexity, since there are polynomially many pairs of constraints to consider. The algorithm in step 5 is likewise polynomial for fixed K. However, a more important question is how well the separation algorithm performs in practice. Computational tests have been carried out using only ordinary resolution cuts (direct cancellation resolvents of 1-clauses) [11]. Separating cuts were generated automatically as in step 4 above, but involving 1-clauses only. The tests indicate that in random problems, separating resolvents exist in about 80% of cases in which cuts are needed, tend to be easy to generate, and solve the problem quickly. In fact, this rudimentary cutting plane method solved random implication problems in which the premises fail to imply the conclusion at least 1000 times faster than set-of-support resolution, even when the problems had as few as 20 variables and 40 clauses. It solved problems in which the premises imply the conclusion only slightly faster than resolution, however. The latter result is due to the nature of random problems, in which the conclusion, if implied at all, is generally derivable from only one or two premises, so that relatively few resolutions are needed to detect the implication. Thus, the resolution algorithm is fast for these problems, nearly as fast as the cutting plane method.

These results indicate that even though resolution cuts tend to be weak – one of the reasons, in fact, that resolution-based symbolic methods are inefficient – *separating* resolution cuts can be quite effective. The discovery of separating cuts adds a powerful direction-finding capability to a cutting plane method, beyond that provided by the objective function. This suggests that the advantage of a cutting plane method, relative to resolution, could be even greater in "real" problems in which implications are less trivial than in random problems.

Nonetheless, stronger cuts than ordinary resolvents may be necessary for deeply-imbedded implications in highly-structured problems. An extreme example is the class of pidgeon-hole problems [25,7,9], in which resolution requires exponential time. A thorough investigation of this area has yet to be carried out.

References

[1] E. Balas and R.G. Jeroslow, Canonical cuts on the unit hypercube, SIAM J. Appl. Math. 23(1972)61.

[2] E. Balas and S.M. Ng, On the set covering polytope: I. All the facets with coefficients in {0, 1, 2}, Working paper 43-85-86, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA; revised February 1986.

[3] C.E. Blair, R.G. Jeroslow and J.K. Lowe, Some results and experiments in programming techniques for propositional logic, Working paper, College of Management, Georgia Institute of Technology, Atlanta, GA; revised July 1985.

[4] G. Boole, *The Mathematical Analysis of Logic* (Cambridge University Press, 1847, reprinted by Oxford University Press, 1948).

[5] V. Chvátal, Edmonds polytopes and a hierarchy of combinatorial problems, Discr. Math. 4(1973)305.

[6] S.A. Cook, The complexity of theorem-proving procedures, *Proc. 3rd ACM Symp. on the Theory of Computing* (1971) pp. 151 – 158.

[7] S.A. Cook and R.A. Reckhow, The relative efficiency of propositional proof systems, J. Symbolic Logic 44(1979)36.

[8] W. Cook, C.R. Coullard and G. Turán, On the complexity of cutting-plane proofs, Working paper, Cornell University, Ithaca, NY (1985).

[9] A. Haken, The intractability of resolution, Theor. Comp. Sci. 39(1985)297.

[10] P.L. Hammer, E.L. Johnson and U.N. Peled, Facets of regular 0-1 polytopes, Math. Progr. 8(1975)179.

[11] J.N. Hooker, Resolution vs. cutting plane solution of inference problems: Some computational experience, Oper. Res. Lett., to appear.

[12] R.G. Jeroslow, An extension of mixed-integer programming models and techniques to some database and artificial intelligence settings, Working paper, College of Management, Georgia Institute of Technology, Atlanta, GA (1985).

[13] R.G. Jeroslow, Computation-oriented reductions of predicate to propositional logic, Working paper, College of Management, Georgia Institute of Technology, Atlanta, GA (1985).

[14] R.G. Jeroslow, On monotone chaining procedures of the CF type, Working paper, College of Management, Georgia Institute of Technology, Atlanta, GA (1985).

[15] R.G. Jeroslow, The theory of cutting planes, in: *Combinatorial Optimization*, ed. N. Christofides et al. (Wiley, New York, 1979) pp. 21–72.

[16] W. Kneale and M. Kneale, *The Development of Logic* (Oxford University Press, London, 1962).

[17] L.B. Kovács, Extended set covering problem and logical programming, Computer and Automation Institute, Hungarian Academy of Sciences, Budapest, presented at the 5th Bonn Workshop on Combinatorial Optimization (1984).

[18] J.K. Lowe, Modelling with integer variables, Ph.D. thesis, Georgia Institute of Technology (1984).

[19] N.J. Nilsson, *Principles of Artificial Intelligence* (Tioga Publ. Co., Palo Alto, CA, 1980).

[20] M.W. Padberg, Characterisation of totally unimodular, balanced and perfect matrices, in: *Combinatorial Programming: Methods and Applications*, ed. B. Roy (1975) pp. 275–284.

[21] M.W. Padberg, Perfect zero-one matrices, Math. Progr. 6(1974)180.

[22] W.V. Quine, The problem of simplifying truth functions, Amer. Math. Monthly 59(1952) 521.

[23] W.V. Quine, A way to simplify truth functions, Amer. Math. Monthly 62(1955)627.

[24] J.A. Robinson, A machine-oriented logic based on the resolution principle, J. ACM 12 (1965)23.

[25] G.S. Tseitin, On the complexity of derivations in the propositional calculus, in: *Structures in Constructive Mathematics and Mathematical Logic*, Part II, ed. A.O. Slisenko (Consultants Bureau, New York, 1968) pp. 115–125 (translated from Russian).

[26] H.P. Williams, Linear and integer programming applied to the propositional calculus, Int. J. Syst. Res. and Inf. Sci. 2(1987)81.