Program Verification Group
Report 10

Computer Science Department
Report No. STAN-CS-78-689
[illegible]

November 1978

AN $n^{\log n}$ ALGORITHM FOR THE TWO-VARIABLE-PER-CONSTRAINT LINEAR PROGRAMMING SATISFIABILITY PROBLEM

by
Charles G. Nelson

Research sponsored by
National Science Foundation

COMPUTER SCIENCE DEPARTMENT
Stanford University

An $n^{\log n}$ algorithm for the two-variable-per-constraint linear programming satisfiability problem

Charles G. Nelson

Stanford Verification Group

Artificial Intelligence Laboratory

Stanford University

Abstract

A simple algorithm is described which determines the satisfiability over the reals of a conjunction of linear inequalities, none of which contains more than two variables. In the worst case the algorithm requires time $O(mn^{\lceil \log_2 n \rceil + 3} \log n)$, where n is the number of variables and m the number of inequalities. Several considerations suggest that the algorithm may be useful in practice: it is simple to implement, it is fast for some important special cases, and if the inequalities are satisfiable it provides valuable information about their solution set. The algorithm is particularly suited to applications in mechanical program verification.

This research was supported *by the Fannie* and John Hertz Foundation and by the National Science Foundation under grant ***MCS-7802825.***

Introduction

Consider the problem of verifying that one linear inequality is a consequence of several other linear inequalities—for example, that $x \geq 0$ is a consequence of $x \geq z$ and $x \geq -z$, or that $y < 2x + 1$ is a consequence of $y \leq x$ and $y \geq -x$. A practical algorithm for this problem is essential to mechanical program verification, since reasoning about linear inequalities occurs so frequently in proofs of program correctness. The problem arises in many other areas-indeed, it is equivalent to the ubiquitous general linear programming problem-but the instances of the problem which arise in verification work seem to differ significantly from those which arise elsewhere, so verification researchers have designed new methods, or modified traditional linear programming methods, to handle the cases they encounter in practice. (See references [1, 4, 5, 6, 7, 8].) The continuing activity in this area is sufficient evidence that none of the methods, new or old, has been entirely satisfactory for verification work.

The results in this paper extend earlier work by V. Pratt and R. Shostak on a subcase of the general problem. Pratt [6] reports that most of the inequalities encountered in verification work are of the form $x \leq y + k$ (where x and y are variables and k is a constant), and gives a polynomial-time algorithm for this special case. Shostak [7] generalizes Pratt's method to allow inequalities of the form $ax + by \leq k$ (or $ax + by < k$) where x and y are variables and a, b, and k are constants. The algorithm described in this paper handles the same subcase considered by Shostak, but it is simpler than Shostak's algorithm and faster in the worst case.

In several of the classical linear programming problems there are only two variables per constraint. This is true of the transportation problem, of some more general network problems, and of the problem Dantzig [2] calls the weighted distribution problem. The algorithm described in this paper can be used to find an initial "feasible" point for such problems, but it cannot be used for maximizing an arbitrary objective function.

If C is a conjunction of linear inequalities and L is a linear inequality, then C implies L if and only if the conjunction of C with the logical negation of L is unsatisfiable. The logical negation of L can be written as a linear inequality (strict if L is non-strict, non-strict if L is strict). Thus the problem of verifying implications between inequalities can be reduced to that of establishing the unsatisfiability of conjunctions of inequalities. By a constraint we mean a linear inequality. By a *two-variable* constraint we mean one which contains at most two variables. The algorithm described in this paper determines the satisfiability over the reals of a conjunction of two-variable constraints.

An important feature of the algorithm is that it computes the two-dimensional

coordinate projections of the solution set. That is, if C is a **satisfiable** conjunction of two-variable constraints, and x and y are any two variables, then at the conclusion of the algorithm the set of all constraints involving only x and y which are consequences of C is explicitly represented, in the form of a polygonal set in the xy plane.

This feature is useful in verification work in at least three ways. First, it is often essential to determine the equalities between variables which are implied by the constraints. The algorithm determines these equalities at no extra cost, since if, say, $x = y$ is a consequence of C, this fact will be manifest from the projection of the solution set on the xy plane. Second, if a number of implications with the same **antecedant** are to be tested, say C $\supset L_1, C \supset L_2, \ldots$, where C is a conjunction of two-variable constraints and the L's are **two-variable** constraints, the algorithm can be used to construct the solution set of C, by means of which each implication can be tested rapidly. Third, if some two-dimensional projection contains no integer lattice point, then it follows that the constraints are not satisfiable over the integers; this piece of information is often valuable. (Of course, if every such projection contains an integer lattice point, the constraints may or may not be satisfiable over the integers.) It is interesting to compare this method with the **Sup-Inf** method of **W. Bledsoe** (see Bledsoe [1] and Shostak [8]), which computes the minimum and maximum of every variable, thus effectively computing the projection of the solution set onto every one-dimensional coordinate axis.

The two-dimensional coordinate projections of the solution set might be useful in operations research applications, since they give the "trade-off' between any two variables.

The worst-case time bound for the algorithm is non-exponential, being approximately $mn^{\log n}$, where m is the number of constraints and n is the number of variables. It is the only non-exponential algorithm for this problem reported in the literature. Yossi Shiloach [9] has described to me an ingenious method he is working on for solving the problem in polynomial time, but his approach is very complicated. The worst case time bound for the algorithm described in this paper can be improved in special cases. If all coefficients are one or minus one, the algorithm takes $O(n^3)$ time; asymptotically this is as fast as the algorithms designed for this special case. If all coefficients are in the set $\{\pm 2, \pm 1, \pm 1/2\}$, the time bound is $O(n^4 \log n)$. In general if the coefficients are restricted to any finite set of reals, the running time can be bounded by a polynomial function of n. The degree of the polynomial is three greater than the rank of the multiplicative group generated by the absolute values of the coefficients. Even with arbitrary coefficients, I have not been able to construct examples showing that the bound is tight. It is not impossible that the algorithm is polynomial.

1. The Algorithm

We begin with a description of the Fourier-Motzkin elimination method for determining the satisfiability of a set of constraints. This method was discovered by J. Fourier in 1826, and reintroduced by T. Motzkin in 1936. (For a comprehensive bibliography of early work in linear inequality systems, see Dantzig [2].)

From a constraint involving a variable x, either an upper bound or a lower bound on x can be obtained, but never both. In either case, the bound obtained (regarded as a function of the other variables) is unique; that is it does not depend on how the original constraint is written. For example, the constraint $x - 2y + 2 \geq 0$ bounds x from below by $2y - 2$, and bounds y from above by x/2 + 1. In general, the constraint $\sum a_i x_i \leq k$ bounds x_i from above if $a_i > 0$, from below if $a_i < 0$.

If we are given two constraints which are not strict, one of which bounds x from below and the other of which bounds x from above, we can write them in the form $L \leq x$, $x \leq U$, where L and U are expressions which do not contain x. We define the x-resultant of the two constraints to be the constraint $L \leq U$. Evidently the x-resultant is a logical consequence of the two constraints, and it is the strongest consequence in the following sense: any assignment of values to the variables in L and U which satisfies the resultant can be extended to assign a value to x which satisfies the original two constraints (simply by choosing for x any value between the values of L and U).

We also define the xc-resultant when one or more of the constraints is strict: the x-resultant of $L < x$ and $x < U$, or of $L \leq x$ and $x < U$, or of $L < x$ and $x \leq U$, is $L < U$.

Note that the resultant of two two-variable constraints is a two-variable constraint.

Proposition. *Let* S be a conjunction of constraints, each of which contains **a** variable x, *T* a conjunction of ***contraints,*** none of which contains x, and *W the* conjunction of ***all*** x-resultants of pairs of constraints of S. Then $S \wedge T$ is satisfiable if and *only* if $W \wedge T$ is satisfiable.

Proof. For notational convenience we assume that none of the constraints **in S is** strict; the proof is similar if the restriction in lifted.

We can write the constraints of S in the form:

$$\begin{array}{ll} L_1 \leq x & x \leq U_1 \\ L_2 \leq x & x \leq U_2 \\ \vdots & \vdots \\ L_n \leq x & x \leq U_m \end{array} \qquad (1)$$

where the L's and U's do not contain x. Then W is the conjunction of the nm x-resultants of these equations, that is

$$\bigwedge_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}} L_i \leq U_j. \tag{2}$$

Now given an assignment of values to the variables other than x which satisfies $W \wedge T$, choose i to maximize the value of L_i under the assignment, and j to minimize the value of U_j under the assignment. The value of L_i does not exceed that of U_j, since the assignment satisfies W, which contains the constraint $L_i \leq U_j$. Hence we can choose x between the values of L_i and U_j. This choice satisfies the sharpest upper and lower bounds of (1), hence it satisfies all the bounds of (1). The converse is trivial, so the proof of the proposition is complete. ∎

The replacement of (1) by (2) is the general step of the Fourier-Motzkin method. By eliminating variables one by one, we eventually obtain a system of constraints which contains only numerical constants and is satisfiable if and only if the original conjunction is. Unfortunately, the number of constraints increases quadratically with each application of the general step, so if originally there are m constraints in n variables, it appears that the only bounds we can get on the number of constraints at successive steps are approximately $m, m^2, m^4, m^8, \ldots, m^{2^n}$. Possibly more sophisticated analysis would lead to a less astronomical bound, but clearly this method cannot be used on large problems.

The algorithm given below for the two-variable-per-constraint satisfiability problem is somewhat similar to the Fourier-Motzkin method. Instead of forming resultants from only one variable at each step, the algorithm forms resultants from every variable at every step, This means that more new constraints are generated at every step, but has the advantage that only about logn steps are required instead of n, as will be shown. In some sense the algorithm eliminates the variables in parallel instead of sequentially. The algorithm also "filters out" manifestly redundant constraints.

If S is a set of constraints and V is a set of variables, we use the notation S_V to denote the subset of S consisting of all those constraints in which occur only variables of V.

If S is a set of constraints and C is a constraint, then C is a direct consequence of S if C is implied by S_V where V is the set of variables appearing in C. Thus $x \geq 1$ is a direct consequence of $\{x \geq 2\}$, but not of $\{x \geq y, y \geq 2\}$. Also, $x + y \geq 0$ is a direct consequence of $\{x \geq 0, y \geq 0\}$, but not of $\{x \geq 0, y \geq z, z \geq 0\}$.

If S is a set of constraints, to filter S is to repeatedly perform the following step: if S contains a constraint C which is a direct consequence of S- {C}, then remove C from S. The process terminates when S contains no such constraint.

We use "lg n" to denote the logarithm of n to the base two.

Satisfiability Algorithm. Given a set S of two-variable constraints among n variables $v_1, v_2, \ldots, v_n$, the following algorithm determines whether they are **satisfiable.**

```
begin
  for $1 \le i \le \lceil \lg n \rceil - 1$ do
    begin
      let $S'$ be the set of all resultants of pairs of
         elements of S, then replace S by $S \cup S'$;
      for $1 \le j < k \le n$ do
        if $S_{\{v_j, v_k\}}$ is unsatisfiable
          then return "unsatisfiable";
      filter S;
    end;
  return "satisfiable";
end
```

Obviously the algorithm is correct if it answers "unsatisfiable", since all the resultants which are generated are consequences of the original constraints.

We will call the body of the for-loop on i the general step of the **satisfiability** algorithm. To prove that the algorithm is correct if it answers "satisfiable", we must show that if the constraints are unsatisfiable then in $\lceil \lg n \rceil - 1$ application8 of the general step enough resultants will be generated to make some $S_{\{v_j, v_k\}}$ unsatisfiable. Here is an example which gives an idea why this is so: let the initial constraints be

$$1 \le x_1 \le x_2 \le \cdots \le x_n \le 0.$$

(This is an example of a chain of constraints. A formal definition for a chain is given in the next section.) After the first elimination step, the set of constraints will contain (among others) the constraints $1 \le x_2$, $x_2 \le x4$, . . . , $x_{n-2} \le x_n$, $x_n \le 0$ (if n is even), so the set of constraints will contain an unsatisfiable chain of length $n/2$. After about lgn applications of the general step the constraint set will contain an unsatisfiable chain with no more than two variables, and the unsatisfiability will be detected.

Unfortunately it is not the case that every unsatisfiable set of constraints contains an unsatisfiable chain. However, we will show in the next section that every unsatisfiable set of constraint8 contains an unsatisfiable subset which consists of several long chains hooked together, and this will be enough to establish the correctness of the algorithm.

The running time of the algorithm depends critically on the growth of the number of constraints. If the number of constraints grew quadratically with **each** step, then at successive steps the number could be as large as

$$m, m^2, m^4, \ldots, m^{2^{\lg n}} \approx m^n.$$

We will show in section 3 that the filtering step keeps the set from growing by more than a factor of n at each step; since there are about lgn steps, the final number of constraints is bounded by approximately $mn^{\lg n}$. In section 3 we will also consider a representation for the set of constraints which allows the general step to be performed rapidly.

2. The Proof of Correctness

The proof of correctness begins with the following famous theorem.

Theorem *(Helly).* Let F be a *finite family* of convex sets *in* d-dimensional space. If any $d + 1$ sets *of* F have a common *point,* then there *is* a *point common to* all *the sets of F.*

Figure 1 illustrates the two-dimensional case of this theorem.

The theorem was discovered by E. Helly in 1913; the first proof was given by J. Radon. Several proofs and generalizations are given in references [3] and [10]. ∎

The solution set of a single constraint is a half-space, which is a convex set. -Thus if **S** is a set of constraints on k variables, and if every subset of S with no more than $k + 1$ constraints is satisfiable, it follows from Helly's theorem that S is satisfiable. Thus a minimally unsatisfiable set of constraints on k variables contains no more than $k + 1$ constraints. Since each constraint contains at most

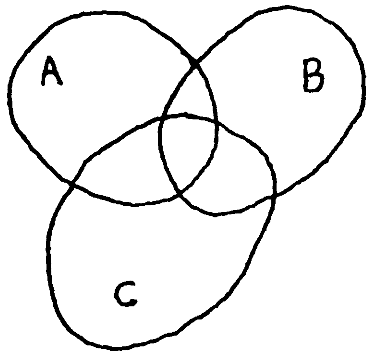


Figure 1. If a fourth convex set intersects the three regions $A \cap B, B \cap C,$ and $A \cap C$, then it must intersect $A \cap B \cap C$.

two variables, this suggests that most of the variables appear in only two of the constraints; that is that a large minimally unsatisfiable set of constraints must contain long chains, Theorem 1 below gives a complete classification of minimally unsatisfiable constraint sets, It is interesting that its proof uses nothing but Helly's theorem.

We say that a sequence of two-variable constraints $(C_0, \ldots, C_n)$ is a chain with internal variables $v_1, \ldots, v_n$ if for each i, $1 \leq i \leq n$, C_{i-1} and C_i have a v_i-resultant, and furthermore v_i does not occur in any of the other C's. C_0 **may** contain another variable besides v_1; if so we call this other variable the left **end** variable of the chain. If C_0 contains only one variable, we say that the chain has no left end variable. The **right** end variable of a chain is defined similarly. Note that the left and right end variables may be the same, in which case we say that the chain is circular. A circular chain has one end variable; a non-circular chain may have zero, one, or two end variables. A circular chain must contain at least one internal variable. Two chains **intersect** if they have a common internal variable. Note that the constraints outnumber the internal variables by one in any chain.

Theorem 1. Let S **be an** unsatisfiable **set** of two-variable **constraints such** that **every proper subset of S is satisfiable. Let** *V* **be the set of variables occurring in constraints of S. Then one of the following cases holds:**

- (trivial case) *V* is empty and S contain6 one constraint;
- (line case) There is a chain **with no end** variables which contains ***all the* constraints in S;**
- **(circle case) There is a circular chain which contains *all* the constraints *in S;***
- (circle-line case) S can **be** partitioned **into two non-intersecting chains, one circular with *end* variable u, the other non-circular with u as *its only end* variable;**
- **(theta case) S can be partitioned into three *pairwise* non-intersecting, *non-*circular chains, each of which have the same pair of end variables;**
- **(eyeglass case) S can be partitioned into three *pairwise* non-intersecting chains, two of which are circular with distinct end variables u and u, the third of which has *u* and u as end variables;**
- ***(figure* eight case) S can be partitioned into two non-intersecting circular chains which have the same end variable.**

Furthermore, each of the seven cases can occur.

The six non-trivial cases of the theorem are illustrated in figure 2.

Proof. **Let** k_i be the number of variables of *V* which occur in exactly i constraints of S. Clearly $k_1 = 0$, by the minimality of S. By **Helly's** theorem S can contain no more than $k+1$ constraints. Since no constraint contains more than two **variables, the sum, over** all variables u, of the number of constraints containing u, is **at** most **twice the** number of constraints; that is

$$\sum ik_i \leq 2k+2. \qquad (3)$$

Since $\sum k_i = k$ and $k_1 = 0$, $\sum_{i>2} k_i = k - k_2$, so from (3) we obtain

$$2k_2 + 3(k-k_2) \leq 2k+2$$

whence

$$k_2 \geq k-2. \qquad (4)$$

Thus we have proved that at most two variables appear in more than two constraints.

Now doubling (4) and subtracting it from (3) gives

$$\sum_{i>2} ik_i \leq 6.$$

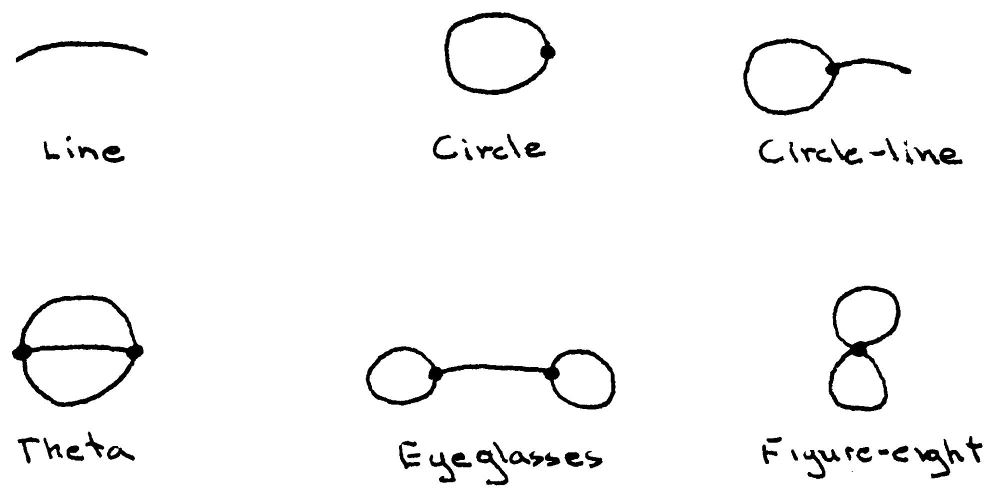
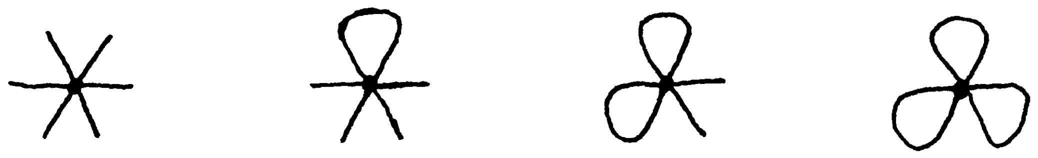


Figure 2. Dots represent variables and lines represent chains in these diagrams of the six types of minimally unsatisfiable sets of constraints. The lines with one undotted end represent non-circular chains with one end variable; the loops represent circular chains. Only end variables are represented by explicit dots; the chains may have any number of internal variables.

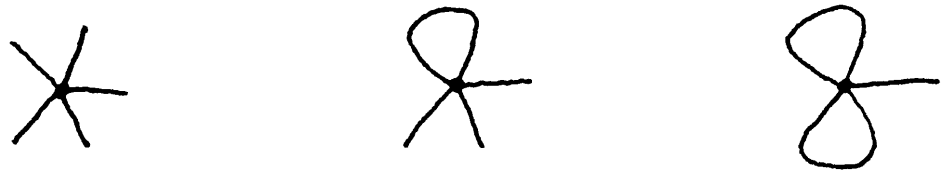
This inequality has only finitely many solutions in non-negative k_i; the remainder of the proof considers each solution separately.

If $k_6 > 0$, then $k_6 = 1$ and $k_i = 0$ for $i \neq 2, 6$. There are four cases, which can be diagrammed (using the conventions of Figure 2) as follows:



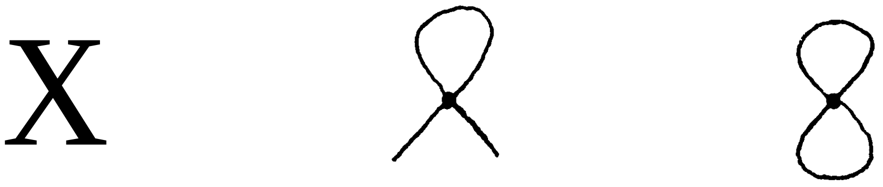
But each of these cases contradicts Helly's theorem, since the constraints **outnum**ber the variables by 5, 4, 3, and 2 (going from left to right).

If $k_5 > 0$, then $k_5 = 1$ and $k_i = 0$ for $i \neq 2, 5$. There are three cases:



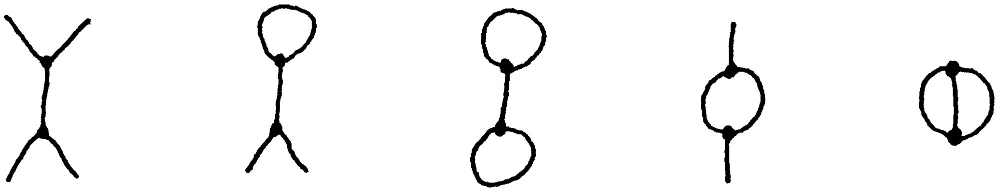
The constraints outnumber the variables by 4, 3, and 2; so none of these cases can occur.

If $k_4 > 0$, then $k_4 = 1$ and $k_i = 0$ for $i \neq 2, 4$. There are three cases:



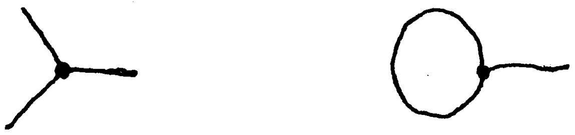
The first two cases cannot occur, since the constraints outnumber the variables by 3 and 2. The third case is the figure-eight case. Here is an example showing that this case can occur: $\{x \geq y, x \geq -y, x < z, x < -z\}$. The first two constraints imply that $x \geq 0$, the second two imply that $x < 0$.

If $k_3 > 1$, then $k_3 = 2$ and $k_i = 0$ for $i \neq 2, 3$. There are five cases:



In the first two cases the constraints outnumber the variables by 3 and 2, so these cases are impossible, The third case is the eyeglasses case. An example of this case is $\{x \geq y, x \geq -y, u \geq v, u \geq -v, x + u < 0\}$; the first four constraints imply that $x \geq 0$ and $u \geq 0$, contradicting the fifth constraint. The fourth case is impossible, since the constraints outnumber the variables by 2. The fifth case is the theta case, an example of which is $\{x \geq y, x \geq -y, y \geq 2x + 1)$.

If $k_3 = 1$, then $k_i = 0$ for $i \neq 2, 3$. There are two cases:



in the first case the constraints outnumber the variables by 2. The second case is the circle-line case, an example of which is $\{x \geq y, x \geq -y, x < 0\}$.

The only cases which remain are those in which $k_i = 0$ for $i \neq 2$; the only such cases (other than the trivial case) are the circle and line cases. Examples of these are $\{x \geq y, x < y\}$ and $\{x \geq 1, x \leq 0\}$.

This completes the proof of theorem 1. ▮

Actually, theorem 1 is a stronger result than we need to establish the correctness of the algorithm. All we really need is the inequality (4), which says that at most two variables appear in more than two constraints.

Lemma 1. Let S be an unsatisfiable *set* of two-variable constraints, every *proper* subset of which is satisfiable, V the set of all variables appearing *in* constraints of S, k the number of variables in V, and S' the set of *all* resultants of pairs of constraints in S. Then there exists a subset V' of V, containing no more than $\lfloor k/2 \rfloor + 1$ variables, such that $(S \cup S')_{V'}$ is unsatisfiable.

Proof. Suppose that $v_1, v_2, \ldots, v_n$ are the internal variables of some chain and that none of the u's appears in more than two constraints of S. If the Fourier-Motzkin method were used to eliminate $v_1, v_3, v_5, \ldots$, only one resultant would be added for each variable, and these resultants would involve only $v_2, v_4, \ldots$ and

the end variables of the chain. Thus if V' contains every variable of V except v_1, v_3, ..., then $(S \bigcup S')_{V'}$ must be unsatisfiable.

This argument shows that we can omit from V' at least half **the internal** variables of any chain with the property that each of its internal variables appears in only two constraints. Now each case of theorem 1 gives a division of S into a set of such chains which together contain at least $k - 2$ internal variables. Thus we **can** omit from V' at least $\lceil (k-2)/2 \rceil$ variables, leaving no more than

$$k - (\lceil k/2 \rceil - 1) = \lfloor k/2 \rfloor + 1$$

variables in V'. ▮

It is easy to show that if the map $x \mapsto \lfloor x/2 \rfloor + 1$ is applied $\lceil \lg n \rceil - 1$ times to the integer n, the result is less than three. Thus if the set of constraints given to the satisfiability algorithm is inconsistent, then in the last application of the general step some $S_{\{x,y\}}$ will become inconsistent. Thus the algorithm is correct.

We will now show that if the constraints are satisfiable, then at the conclusion of the algorithm the solution set of $S_{\{x,y\}}$ is exactly the projection of the solution set onto the xy-plane, for any two variables x and y. This in a consequence of the following stronger version of lemma 1, whose proof is omitted since it is essentially the same as the proof of the first version.

Lemma lb. ***Let*** S ***be*** a ***set*** of two-variable constraints, C a two-variable constraint such that $S \bigcup \{C\}$ is unsatisfiable, and suppose that $T \bigcup \{C\}$ is satisfiable for ***all proper*** subsets ***T of*** S. ***Let V be the set*** of variables appearing ***in*** constraints of S, ***k*** the number of variables in ***V,*** and S' the ***set*** of ***all*** resultants of pairs of elements of S. ***Then there*** exists a subset V' ***of V,*** containing ***no more*** than $\lfloor k/2 \rfloor + 1$ variables, such that $(S \bigcup S')_{V'} \bigcup \{C\}$ is unsatisfiable. ▮

Note that lemma 1 is a special case of lemma lb, since we can take C to be the constraint $0 \leq 0$. Lemma lb says that the size of a miminal set entailing some constraint C shrinks by a factor of two for each application of the general step, so we know that at the conclusion of the algorithm C will be entailed by some set of constraints involving only two variables.

Note that if the general step were repeated more than $\lceil \lg n \rceil - 1$ times, the extra steps would have no effect-every resultant generated would immediately be filtered out.

When the constraints are satisfiable it is sometimes required to find a point which satisfies them. Let x and y be any two variables; since the algorithm computes the projection of the solution set onto the xy plane, we can choose for x and y any pair of values which lies in this projection. By substituting these values for x and y and running the algorithm again, we can determine the set of values for the other variables consistent with the choices made for x and y. By continuing in this way we can find a satisfying point after $n/2$ applications of the algorithm.

3. The Proof of Correctness of the Time Bound

The correctness of the time bound for the satisfiability algorithm rests on the following theorem:

Theorem 2. Let S be a set containing n two-variable constraints, each of which contains both x and y; T a sef containing m two-variable constraints, each of which contains both y and z; and U *the* result of filtering the set of *all* y-resultants of constraints in S with constraints in T. *Then* U contains no more than $2(n+m)$ constraints.

This theorem is illustrated in Figure 3. In xyz space, the solution set of the xy constraints is a cylinder parallel to the z axis, with cross-section a polygon in the xy plane. Similarly the solution set of the yz constraints is a cylinder parallel to the x axis, with cross-section a polygon in the yz plane. The intersection of these cylinders is the solution set of all $n+$ m constraints. The projection of the intersection is a polygon in the xz plane, and each edge of this polygon corresponds to a resultant of some xy constraint with some yz constraint. Thus it suffices to bound the number of edges of the projection.

The shape of the intersection is between that of a ball and a lens, with an upper surface and lower surface meeting in an edge curved like the seam of a baseball. It is easy to see that each vertex of the intersection must lie on an edge

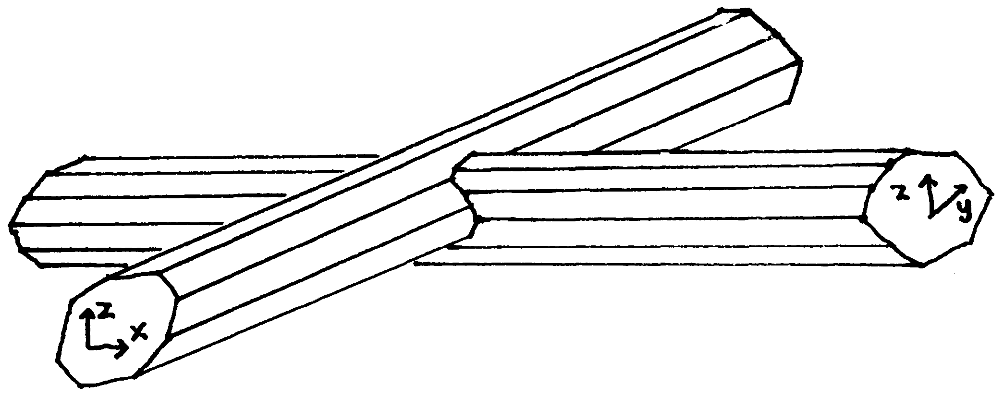


Figure 3

of one of the cylinders, and that no cylinder edge contains more than two vertices of the intersection. Thus the intersection has no more than $2(n+m)$ vertices; but then the same bound applies to the number of vertices-hence to the number of edges-of the projection of the intersection.

This argument neglects certain possibilities, such as the possibility that either the xy or the yz solution set was unbounded. The argument can be made perfectly rigorous; but rather than do so we will present, later in this section, an algebraic proof of theorem 2. The second proof may give more insight into the theorem, and moreover it provides an $O(n+m)$ algorithm for constructing U from S and T.

It is natural to expect that the Fourier-Motzkin method would run faster for the two-variable-per-constraint case if the set of constraints were filtered after every application of the general step. In fact, it follows easily from theorem 2 that the worst case of this method is no more than singly exponential.

In the remainder of this section we assume that none of the constraint6 are strict; the modifications necessary to handle strict constraints are straightforward.

The solution set of a conjunction of constraints on two variables is a polygonal region whose edges correspond to the constraints. It is convenient to view the boundary of the polygon as a single piece-wise linear function instead of as many linear functions, For example, consider the three constraints

$$\begin{aligned} 2y &\geq x-1 \\ y &\geq x \\ y &\geq 2x-1. \end{aligned} \tag{5}$$

These are graphed in figure 4. They are equivalent to the single constraint y $\geq$ f(x) where f is defined by

$$f(x) = \begin{cases} x/2-1/2 & \text{if } x \leq -1 \\ x & \text{if } -1 \leq x \leq 1 \\ 2x-1 & \text{if } x \geq 1. \end{cases}$$

In general, any set of constraints, each of which bounds x from above and y from below, is equivalent to the single constraint y $\geq$ f(x), for some increasing piece-wise linear function f. Although we are taking this point of view primarily in order to prove theorem 2, it happen6 that the general step of the satisfiability algorithm can be implemented efficiently if the set of constraints is represented using such piece-wise linear functions.

We will now be more precise: A *linear* function is a function whose graph is a straight line. A function f is piece-wise *linear* if either (a) it is a linear function or (b) there exist n real numbers $a_1 < \ldots < a_n$ and $n+1$ linear function6 $f_0, \ldots, f_n$ such that $f(x) = f_0(x)$ for $x \leq a_1$, $f(x) = f_n(x)$ for $x > a_n$,

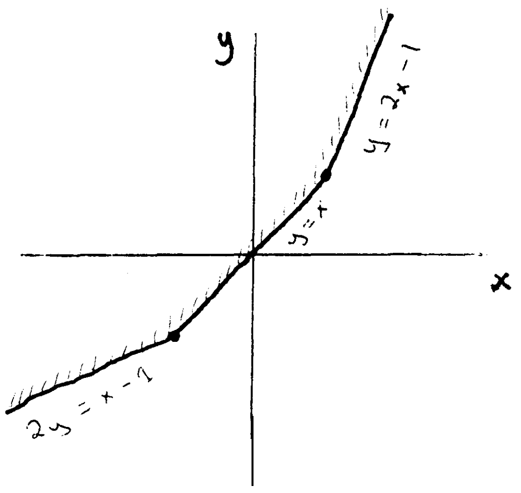


Figure 4. Solution set for the constraints (5).

and $f(x) = f_i(x)$ for $a_i < x \leq a_{i+1}$, $1 \leq i < n$. We call **the** f_i **the components of** f. **A** boundary function is a continuous piece-wise linear function which **is either strictly increasing** or strictly decreasing.

The next lemma is a collection of facts about boundary functions whose proofs -are trivial and will be omitted.

Lemma 2. Let f and g *be* two boundary functions with n and m components respectively. *Then:*

a) *The* composition gf off and g is a boundary function with ***no more*** **than** $n+m$ **components.** *If* one of/, g is increasing and the *other* **is decreasing,** then gf is decreasing; otherwise gf is increasing.

b) *If* f and g are increasing (decreasing) *then* both min(f, g) and **max**$(f,$ g) are increasing (decreasing) boundary functions with no more than $n+m$ components.

c) The inverse f^{-1} off is a boundary function, and it is increasing or **decreasing** according as f is.

Let S be **a** conjunction of k constraints involving x and y. Then:

d) If each constraint of S bounds x from above and y from *below,* S is equiv-**alent** to the inequality *h(x)* $\leq y,$ for *some* increasing boundary function h with at most k components.

e) *If* each constraint of S bounds both x and y from ***below,*** S is equivalent ***to the*** inequality *h(x)* $\leq$ y, **for** some decreasing boundary function h **with** at most k components.

f) If each constraint of S bounds both x and **y from** above, *S* is ***equivalent to the inequality h(x) $\geq y$, for*** some decreasing boundary function ***h with*** at most ***k*** components. ∎

Given a set *S* of constraints between x and y, it follows from the last three parts of this lemma that we can find four boundary functions *L, U, l,* and u, where *L* and *U* are increasing and *l* and u are decreasing, such that a point **(x, y) satisfies** S if and only if the following four conditions are satisfied:

$$\begin{aligned} L(x) &\leq y \\ l(x) &\leq y \\ U(x) &\geq y \\ u(x) &\geq y \end{aligned} \tag{6}$$

(To simplify the discussion, we ignore the possibility that S might contain no constraints of one of the four "types," for example, no constraint bounding x **from** below and y from above.) In this case we say that the quadruple

$$\begin{pmatrix} U & u \\ l & L \end{pmatrix}$$

represents S in ***the*** xy-plane. The functions are arranged in a square to suggest which edges of the solution set they correspond to, as shown in figure 5.

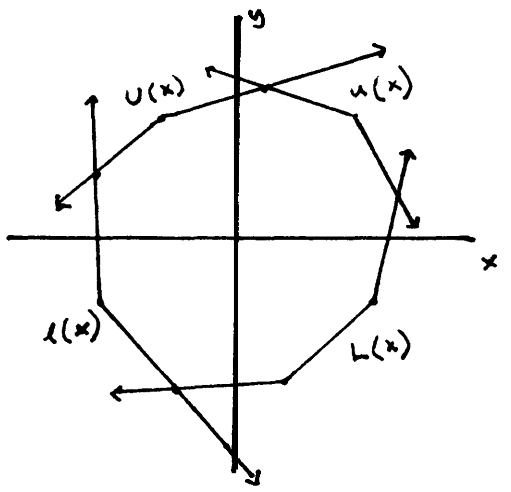


Figure 5. A polygon which contains no edges parallel to the axes can be represented as the solution set of four inequality constraints using piece-wise linear monotonic functions.

This definition is not symmetric between x and y. If L, l, U, **and u represent** S in the xy-plane, then we can rewrite the inequalities (6) (using **the fact that L and U are increasing and l** and u are decreasing):

$$\begin{aligned} x &\leq L^{-1}(y) \\ x &\geq l^{-1}(y) \\ x &\geq U^{-1}(y) \\ x &\leq u^{-1}(y) \end{aligned}$$

so the representation of S in the yx-plane is

$$\begin{pmatrix} L^{-1} & u^{-1} \\ l^{-1} & U^{-1} \end{pmatrix}.$$

Lemma 3. ***Let*** S be a set of constraints ***between*** x and y, S' a ***set of*** constraints ***between*** y and z, and Jet the representations of S and S' in ***the*** xy and yz **planes** be ***respectively***

$$\begin{pmatrix} U & u \\ l & L \end{pmatrix}, \qquad \begin{pmatrix} U' & u' \\ l' & L' \end{pmatrix}. \tag{7}$$

Then the set of y-resultants of constraints in S with constraints ***in S' is represented*** **in the *xz-plane* by**

$$\begin{pmatrix} \min(U'U, u'l), & \min(U'u, u'L) \\ \max(L'l,\ V\,U), & \max(L'L, l'u) \end{pmatrix}. \tag{8}$$

Proof. Let T be the conjunction of all constraints in S which bound x from above and y from below, T' the conjunction of all constraints in S' which bound y from above and z from below, and R the conjunction of all resultants of constraints in T with constraints in T'. Then (x, y) satisfies T if and only if L(x) $\leq$ y, and (y, z) satisfies T' if and only if $L'(y) \leq z$. We will show that (x, z) satisfies R **if and only if** $L'L(x) \leq z$.

By the correctness of the Fourier-Motzkin elimination method, a pair (x, z) satisfies R if and only if there exists a y such that (x, y) satisfies T and (y, z) satisfies T'; suppose that (x, z) satisfies R and choose such a y. Then ***L(x)*** $\leq$ y and $L'(y) \leq z$; since L' is increasing we have $L'L(x) \leq z$ as was to be proved. Now **suppose** $L'L(x) \leq z$, then ***L(x)*** $\leq L'^{-1}(z)$; choose y between ***L(x)*** and $L'^{-1}(z)$. With this choice of y, (x, y) satisfies T and (y, z) satisfies T', hence (x, z) satisfies R.

Thus we have shown that the inequality $L'L(x) \leq z$ is equivalent to the conjunction of all resultants which can be formed from **constraints** in S which bound **x from above** and y from below with constraints in S' which bound y from **above**

and z from below. By a similar argument it can be shown that the inequality $l'u(x) \leq z$ is equivalent to the conjunction of the resultants which can be formed from constraints in S which bound x and y from above with constraints in S' which bound y and z from below, Since this accounts for all resultants which bound x **from above** and z from below, we have proved that the lower right entry of (8) is correct. The correctness of the other entries can be established similarly. ∎

Each of the eight boundary functions appearing in (7) occurs twice in (8). By lemma 2, parts a and b, it follows that the sum of the numbers of components of the four functions in (8) is no more than twice the sum of the numbers of components of the eight functions in (7). **We** have therefore proved theorem 2.

We can now bound the growth of the set of constraints in the **satisfiability** algorithm. Instead of using the total number m of constraints, it is easier to consider ‘the maximum, over **all** pairs of variables (u, u), of the number of constraint6 involving both u and v. **We** will call this maximum the **degree of the set of constraints**. Let k be the degree of the set of constraints before some execution of the general step, and k' the degree after. **We** show that $k' \leq 2nk$, where n **is the** number of variables. Let x and z be any pair of variables. How many constraints involving both x and z can be added in this execution of the general step? Given y, the number of new xz constraints which are y-resultants is at most $2k$, by theorem 2. There are n — **2** other variables, so at most $2k(n-2)$ constraints can be added to the (at most) k constraints already present. ***So*** $k' \leq 2k(n-2) + k \leq 2kn$.

The original degree is bounded by m, the original number **of constraints. Hence we can** bound the degree of the set of constraints after successive executions of **the** general step by

$$m, 2nm, (2n)^2m, \ldots, (2n)^{\lceil \lg n \rceil - 1}m. \tag{10}$$

How much time is required to execute the general step of the algorithm, if there are n variables and the degree of the set of constraints is k? Using the obvious representation for boundary functions, the operations needed to form (8) from (7) can be performed in $O(k)$ time. For each of the $O(n^2)$ pairs of variables (x, z), there are O(n) variables y for which we must perform the operations; thus it requires $O(kn^3)$ time to form the resultants. In addition we must compute the intersection of the xz polygons obtained from the different choices of y; there are O(n) such polygons, each of which has $O(nk)$ sides. Using an obvious divide and conquer strategy, we can form their intersection in $O(kn^2 \log n)$ time. The amount **of time required to handle** the 2n constraints which involve only one variable **may be ignored, Thus the** total time required for one execution of **the general step is**

$$O(kn^3) + nO(kn^2 \log n) = O(kn^3 \log n). \tag{11}$$

From (10) and (11) we therefore find that the total running time of the algorithm is .

$$0\left(mn^3 \log n \sum_{0\leq i<\lg n} (2n)^i\right) = O(mn^{\lceil \lg n\rceil+3} \log n).$$

This completes the proof of correctness of the time bound,

Worst-case behavior by the algorithm seems very unlikely. Let k be the degree of the constraint set before some execution of the general step and x and y any two variables. For each variable z, distinct from x and y, a new polygon with as many as $2k$ sides is intersected into the sy-solution set during the execution of the general step; thus $n-2$ such polygons are intersected. The worst case growth occurs only if the resulting polygon has $2k(n-2)$ sides. This can only happen if each edge of the second truncates a vertex of the first, each edge of the third truncates a vertex of the intersection of the first two, and so forth.

The algorithm is fast in the special case that no coefficients other than plus or minus one occur. In this case, there are never more than four constraints involving any given pair of variables, so the algorithm requires $O(n^2)$ space and $O(n^3)$ time. This is comparable to the time bounds for algorithms which are designed specifically for this special case,

More generally, if the original coefficients lie in any multiplicative subgroup of the reals, then the coefficients of any resultants will lie in the same group. This follows from the expression (8) for the resultants, and the fact that the slope of the composition of two linear functions is the product of their individual slopes.

We can use this fact to prove that the algorithm is polynomial in many other cases. For example, suppose that the initial coefficients have absolute values in the set $\{1/2, 1, 2\}$. After the first execution of the general step, the absolute values of the coefficients must lie in the set $\{1/4, 1/2, 1, 2, 4\}$. After the ith execution of the general step, the absolute values of the base-two logarithms of the absolute values of the coefficients lie in the set $\{0, 1, 2, \ldots, 2^{i+1}\}$. Thus the total number of coefficients which can ever appear is bounded by $4 \cdot 2^{\lceil \lg n\rceil+1} \leq 16n$. Since a polygon can have at most two edges with a given slope, the degree of the set of constraints is bounded by $32n$, and the running time by $O(n^4 \log n)$.

In general, if initially every coefficient's absolute value can be written in the form $g_1^{i_1} g_2^{i_2} \ldots g_l^{i_l}$, where the g's are real numbers and the absolute values of the i's are bounded by B, then after one execution of the general step the absolute values of all new coefficients can be written in the same form, with the absolute values of the exponents bounded by $2B$. After j steps the bound becomes $2^j B$; hence at the conclusion of the algorithm every coefficient can be written in the given form with the exponents bounded by $O(nB)$. Thus no more than $O(nB)^l$ coefficients can occur. We have proved that if the coefficients are restricted to any finite set, **then** the behavior of the algorithm is polynomial in the number of variables. We have proved that if the coefficients are restricted to any finite set, then the **behavior of the** algorithm is polynomial in the number of variables.

References

1. **W. W. Bledsoe: The Sup-Inf method in presburger arithmetic. Univ. of Texas** Math Dept. **Memo ATP-18, December 1974.**
2. **G. Dantzig:** Linear Programming and Extensions. Princeton University **Press,** Princeton, New Jersey, 1963,
3. L. Danzer, B. **Grünbaum,** and V. Klee: Helly's theorem and **its relatives.** American Mathematical Symposium on Convexity, Seattle, ***Proc. Symp. Pure*** **Math, v. 7, 1963.**
4. **L. P. Deutsch: An *Interactive* Program Verifier. PhD Thesis, Univ. of** California, Berkeley, 1973.
5. **J.** King: A **Program** Verifier. **PhD Thesis, Carnegie-Mellon Univ., Pittsburg 1969.**
6. V. Pratt: Two easy theories whose combination is hard. Manuscript, September **1977.**
7. R. Shostak: Deciding linear inequalities by computing loop residues. **Manuscript, March 1978.**
8. **R.** Shostak: On the **Sup-Inf** method for proving presburger formulas. ***JACM,*** **October 1977.**
9. Y. Shiloach: An efficient algorithm for solving systems of linear **inequalities** with two variables per constraint. **In** preparation.
10. F. Valentine: Convex ***Sets.*** McGraw-Hill, New York, 1964.