Solving Problems by Formula Manipulation in Logic and Linear Inequalities*

Louis Hodes

Heuristics Laboratory, Division of Computer Research and Technology, National Institutes of Health, Department of Health, Education and Welfare, Bethesda, Maryland 20014

Recommended by B. Raphael

ABSTRACT

Using formal logic, many problems from the general area of linear inequalities can be expressed in the elementary theory of addition on the real numbers (EAR). We describe a method for eliminating quantifiers in EAR which has been programmed and demonstrate its usefulness in solving some problems related to linear programming.

In the area of mechanical mathematics this kind of approach has been neglected in favor of more generalized methods based on Herbrand expansion. However, in a restricted area, such as linear inequalities, the use of these specialized methods can increase efficiency by several orders of magnitude over an axiomatic Herbrand approach, and make practical problems accessible.

As is common in artificial intelligence, the work reported here is of an interdisciplinary nature. It involves mathematical logic, linear inequalities, and symbolic mathematics on a computer.

For the sake of argument, let us distinguish two kinds of workers in the area of linear inequalities. There is the *theoretician*, who is developing new methods and discovering new theorems. Then there is the *user*, who is faced with a practical problem which can be expressed in some way at least piecewise linearly. As a simple-minded distinction between the theoretician and the user we can say that the latter is interested in questions involving a fixed number of variables, while the former is concerned with questions involving an arbitrary number of variables. Using terminology from logic to be made more precise below this means that the user is generally working within the *elementary theory of addition on the reals* while the theoretician is generally working on a higher level.

* This is a revised version of a paper presented at the International Joint Conference on Artificial Intelligence, London, September 1971.

Artificial Intelligence 3 (1972), 165–174

In this paper we are concerned mainly with the user at the stage where he has formulated his problem in symbolic terms. This may or may not be the first stage in formulating the problem. For example, he may first have developed a model as a mathematical idealization of his problem. Indeed, if he is very lucky, the model may be of a standard type, for example, flow networks, for which there are known efficient solution methods. In that case he would bypass a symbolic formulation.

Suppose, now, in one way or another the user has arrived at a symbolic formulation of the problem. We assume that the size of the problem is such that he would want to use a computer. Again, he may have a perfectly standard problem such as to find a solution to a set of simultaneous inequalities. Then he can use the relaxation method or another numerical approach. Or, if he can formulate his problem as an optimization problem, he may conveniently be able to express it in linear programming format and use the famous simplex algorithm.

Problems can arise, however, which do not fit too easily into the standard molds. Furthermore, one may require a solution in symbolic form. For example in n-person game theory the set of solutions can be described as a union of convex polyhedra, these polyhedra as an intersection of half-spaces, each of which is represented by a linear inequality.

Using formal logic one can often represent one's problem conveniently in the lower predicate calculus under the interpretation of addition on the real numbers. In that case, there are simple methods for eliminating quantifiers and simplifying expressions, often resulting in the solution of various problems. Such methods were first programmed by the author at IBM in 1962 [4] and further elaborated with a game theory application [5]. Since then some modifications and improvements have been made. The complete methods are described here since the earlier papers were never formally published.

Although the author has been addressing himself to workers in linear inequalities, the work reported here may also be categorized as mechanical mathematics or equivalently applied logic on computers. In this field such work has been neglected in favor of more generalized methods based on Herbrand expansion. The effort on these general methods is, of course, worthwhile and productive, but the neglect is unfortunate since restricted specific theories (like addition on the real numbers) often allow the use of direct, specialized methods which increase efficiency by several orders of magnitude over an axiomatic Herbrand approach.

We now describe the elementary theory of addition on the reals (EAR), or, more precisely, the elementary theory of ordered dense Abelian groups without endpoints. "Elementary" means that all formulas in EAR belong to the lower predicate calculus, i.e., quantification occurs only over variables for elements of the group and not, for example, over variables for sets of

elements. The formulas in EAR, to be defined more precisely below, are just those formulas of the lower predicate calculus with two binary predicates, $<$ and $=$, the operator $+$, and at least two constant symbols, 0 and 1.

We refer the uninitiated reader to Rogers' [7] expository paper which, without presupposing any knowledge of formal logic, develops the elementary theory of dense linear order together with a method due to Langford for deciding sentences within that theory. This method extends to EAR, as pointed out to me by A. Robinson, and is described below. EAR has an unlimited set of *variable symbols*, $x, y, z, x_1, y_1, z_1, \ldots$, and real numbers as *constant symbols*, of which we will need only 0 and 1.

A *term* in EAR is either a constant symbol or a variable symbol or of the form $t_1 + \cdots + t_n$, where the t_i, $1 \leqslant i \leqslant n$, are terms. These are the only terms.

We write $x + \cdots + x$, n times, as nx and $1 + \cdots + 1$, n times, as n, so the following are examples of terms.

(1) $2x + 3y + x$

(2) $4 + x_9$

Using the commutativity of addition we see that there is a canonical form for terms where each variable symbol appears once. Thus, example (1) above could be written $3x + 3y$.

An *atomic formula*, or *atom*, is any expression of the form $t_1 = t_2$ or of the form $t_1 < t_2$, where t_1 and t_2 are terms. There is a canonical form for atoms where each variable symbol appears on at most one side of any atom. For example $3x + 3y < 2x$ would be written $x + 3y < 0$. We also allow two constant atoms: TRUE and FALSE.

Atoms are the simplest *formulas*. Non-atomic formulas are built from atoms by means of the propositional connectives; negation, $\neg$, "and", $\&$, "or", $\vee$, "implies", $\rightarrow$, "if and only if", $\leftrightarrow$, as well as the universal and existential quantifiers, "for all x", $(\forall x)$, and "there is an x such that," $(\exists x)$, where x is any variable symbol.

Examples of formulas are:

(1) $x < y \vee x = y \vee y < x$

(2) $(\exists x)(x < y \,\&\, 2y < x + 5)$

(3) $(\forall x)(\exists y)\, \neg 2x + y < 0$

Formally, we say that an atom is a formula and if f_1 and f_2 are formulas then $\neg f_1$, $(f_1 \,\&\, f_2)$, $(f_1 \vee f_2)$, $(f_1 \rightarrow f_2)$, $(f_1 \leftrightarrow f_2)$, $(\forall x) f_1$, $(\exists x) f_1$ are all formulas. Every formula is so derived. We delete parentheses to improve readability, being careful not to introduce undesirable ambiguities. Example (1) above illustrates desirable ambiguity.

For convenience, we use $t_1 \geqslant t_2$ as an abbreviation of $\neg t_1 < t_2$.

As a further example of a formula in EAR we represent the statement

that the real numbers $x_1, \ldots, x_n$ minimize the sum $\Sigma_1^n c_i x_i$ subject to the conditions

$$\sum_{i=1}^{n} a_{ij} x_i = b_j \quad \text{for} \quad j = 1, \ldots, m$$

$$x_i \geqslant 0 \quad \text{for} \quad i = 1, \ldots, n$$

This is a statement of the linear programming problem, and says that $x_1, \ldots, x_n$ yield an optimal solution. We assume, for simplicity, that the a_{ij}, b_j and c_i are all integers.

We allow the use of the minus sign by considering any occurrence of $-t$ in an atom as a convenient notation for $+t$ on the opposite side of the atom.

The linear programming problem is stated in EAR as follows:

$$\begin{aligned} &\mathop{\&}_{j=1}^{m} \sum_{i=1}^{n} a_{ij} x_i = b_j \\ \& \quad &\mathop{\&}_{i=1}^{n} x_i \geqslant 0 \\ \& \quad &(\forall y_1) \cdots (\forall y_n)\Big(\mathop{\&}_{j=1}^{m} \sum_{i=1}^{n} a_{ij} y_i = b_j \quad \& \quad \mathop{\&}_{i=1}^{n} y_i \geqslant 0 \\ &\qquad \rightarrow \sum_{i=1}^{n} c_i y_i \geqslant \sum_{i=1}^{n} c_i x_i \Big) \end{aligned} \tag{1}$$

From the point of view of people working in linear inequalities the most novel feature of EAR is the formal use of quantifiers. In fact, the elimination of quantifiers, together with their bound variables, is the main content of the algorithm described below.

Suppose $F(x_1, \ldots, x_n)$ is a formula in EAR in which $x_1, \ldots, x_n$ are the only free (not quantified) variables. Then the following algorithm yields a quantifier-free formula $G(x_1, \ldots, x_n)$ in which $x_1, \ldots, x_n$ are the only variables, with the following property. Let $a_1, \ldots, a_n$ be any n real numbers. Then $F(a_1, \ldots, a_n)$ and $G(a_1, \ldots, a_n)$ are equivalent in the sense that they are both true or are both false. For example, if $F(x)$ is the formula $(\exists y)$ $(y < x + 4 \ \& \ 2x < y + 5)$ then $G(x)$ would be the formula $x < 9$.

Since EAR is a subset of elementary algebra, Tarski's algorithm [8] provides a method for eliminating quantifiers. When applied to formulas in EAR, however, Tarski's method is exceedingly cumbersome. Furthermore, no one has reported a computer program for any algorithm to eliminate quantifiers in elementary algebra except for some beginnings in that direction by Collins [2]. On the other hand, EAR has a simple method and seems to be in the unique position of having such a simple algorithm along with a wide range of application.

The elimination of quantifiers is achieved by systematically replacing

selected subformulas by equivalent formulas. The equivalence may be purely logical or may be a property of $+$, $=$, $<$ in the reals, as we used, for example, in establishing a canonical form for atoms.

Step 1. If $F(x_1, \ldots, x_n)$ is quantifier-free, let $G = F$. Otherwise, choose a subformula of the form $(\exists y)f(x_1, \ldots, x_n, y)$ where f is quantifier-free, i.e., $(\exists y)$ is an innermost existential quantifier. If all innermost quantifiers are universal quantifiers then use the identity $(\forall x)g \Leftrightarrow \neg(\exists x)\neg g$, so that $(\exists x)\neg g$ yields the required subformula $(\exists x)f$, with $f = \neg g$.

Step 2. We use steps 3–8 to find a quantifier-free formula $g(x_1, \ldots, x_n)$ equivalent to $(\exists y)f(x_1, \ldots, x_n, y)$. We replace $(\exists y)f$ by g in F, then return to Step 1.

Step 3. Eliminate all occurrences of equivalence, $(\leftrightarrow)$, and implication, $(\rightarrow)$ in the formula $(\exists y)f$ by the use of identities

$$f_1 \leftrightarrow f_2 \equiv (f_1 \rightarrow f_2)\ \&\ (f_2 \rightarrow f_1)$$
$$f_1 \rightarrow f_2 \equiv \neg f_1 \vee f_2$$

Step 4. The only connectives in f are $\&$, $\vee$, $\neg$. Use the identities

$$\neg(f_1\ \&\ f_2) \equiv (\neg f_1 \vee \neg f_2)$$
$$\neg(f_1 \vee f_2) \equiv (\neg f_1\ \&\ \neg f_2)$$
$$\neg\neg f_1 \equiv f_1$$

to eliminate all occurrences of negation save those appearing directly before atoms.

Step 5. The remaining negations are eliminated by using the mathematical identities:

$$\neg t_1 = t_2 \equiv t_1 < t_2 \vee t_2 < t_1$$
$$\neg t_1 < t_2 \equiv t_1 = t_2 \vee t_2 < t_1$$

Call the new formula $(\exists y)f$.

Step 6. Now f has connectives $\&$, $\vee$ only. Use the distributive law $f_1\ \&(f_2 \vee f_3) \equiv (f_1\ \&f_2) \vee (f_1\ \&f_3)$ to exapand f into disjunctive normal form

$$f \equiv f_1 \vee f_2 \cdots \vee f_n$$

where each f_i is a conjunction of atoms.

Step 7. The existential quantifier distributes over disjunction. So

$$(\exists y)f \equiv (\exists y)(f_1 \vee \cdots \vee f_n) \equiv (\exists y)f_1 \vee \cdots \vee (\exists y)f_n$$

Each f_i can be written $f_i'\ \&\ f_i''$ where f_i' is a (possibly empty) conjunction of all the atoms which contain y. Since f_i'' does not contain y, we have

$$(\exists y)(f_i'\ \&\ f_i'') \equiv f_i''\ \&\ (\exists y)f_i'$$

or $(\exists y)f_i \equiv f_i$ if f_i' is empty.

Step 8. We must now eliminate the quantifier from each formula $(\exists y)f_i'$. Each such formula must be of the form

$$(\exists y)(t_1^{(i)} = s_1^{(i)}\ \&\cdots\&\ t_{j_i}^{(i)} = s_{j_i}^{(i)}\ \&\ q_1^{(i)} < r_1^{(i)}\ \&\cdots\&\ q_{k_i}^{(i)} < r_{k_i}^{(i)})$$

For $1 \leqslant i \leqslant n$, $j_i \geqslant 0$ is the number of atoms of the type $t_1 = t_2$ and k_i is the number of atoms of the type $t_1 < t_2$. We can assume $j_i + k_i > 0$, i.e., f_i' is non-empty. In each atom y appears on the right or on the left, but not both.

We now consider the case $j_i > 0$, i.e., there is at least one equality. If $j_i = 1$ and $k_i = 0$ then we replace $(\exists y)(t_1^{(i)} = s_1^{(i)})$ by the constant atom TRUE. Otherwise, we solve $t_1^{(i)} = s_1^{(i)}$ for y and substitute this expression for y in the remaining $j_i + k_i - 1$ atoms. Let g_i be the conjunction of these $j_i + k_i - 1$ atoms, restored to canonical form. We replace $(\exists y)f_i'$ by g_i, thereby eliminating the quantifier.

Now we consider the case $j_i = 0$. We have $(\exists y)f_i'$ of the form

$$(\exists y)(q_1^{(i)} < r_1^{(i)} \,\&\, \cdots \,\&\, q_{k_i}^{(i)} < r_{k_i}^{(i)})$$

Suppose y appears always on the right, i.e., in every $r_p^{(i)}$, $1 \leqslant p \leqslant k_i$. Then we replace $(\exists y)f_i'$ by TRUE. Similarly, if y appears always on the left, i.e., in every $q_p^{(i)}$, $1 \leqslant p \leqslant k_i$, we replace $(\exists y)f_i'$ by TRUE.

In the remaining case, for each m, p such that y occurs on opposite sides in the atoms $q_m^{(i)} < r_m^{(i)}$ and $q_p^{(i)} < r_p^{(i)}$ we eliminate y from the pair of inequalities by forming the weighted sum, $\oplus$. Each such m, p pair forms a new inequality and finally we get

$$(\exists y)f_i' \equiv \mathop{\&}_{m,p} q_m^{(i)} < r_m^{(i)} \oplus q_p^{(i)} < r_p^{(i)}$$

Note: Let C_i be the coefficient of y in the ith inequality, $i = 1, 2$, Then the weighted sum is C_2 times the first inequality added to C_1 times the second inequality. In the resultant inequality y has coefficient C_1C_2 on both sides and thus vanishes.

From Step 8 we return to Step 2 to complete the algorithm.

We can make two observations before giving some examples of the algorithm.

1. If the original formula were closed, i.e., all its variables were quantified then the algorithm produces a quantifier-free formula with no variables. In that case the only atoms are of the form $0 < 0$, $0 = 0$, $0 < n$, $n < 0$, $0 = n$, $n = 0$. These can be replaced by the constant atoms FALSE, TRUE, TRUE, FALSE, FALSE, FALSE, respectively. Then the propositional identities TRUE & FALSE ≡ FALSE, etc., can be repeatedly used as the last step of the algorithm to complete the reduction of the formula to a single atom, either TRUE or FALSE. Such an extended algorithm provides a *decision procedure* for EAR, i.e., a method for determining the truth or falsity of any sentence expressible in EAR.

2. With slight modifications of the algorithm we can consider $t_1 \geqslant t_2$ and even $t_1 \neq t_2$ as atoms instead of as abbreviations of $\neg t_1 < t_2$ and $\neg t_1 = t_2$. In this case the only steps of the algorithm which can increase the

length of a formula are expansion into disjunctive normal form (Steps 3 and 6) and Step 8 as we will see from further consideration of the linear programming example.

Let us now see how one may apply the algorithm to linear programming. We rewrite a portion of the formula (1) but instead of the cost $\Sigma c_i x_i$ we use the variable z.

$$(\forall y_1)\cdots(\forall y_n)\left(\mathop{\&}_{j=1}^{m}\sum_{i=1}^{n} a_{ij}y_i = b_j \;\&\; \mathop{\&}_{1}^{n} y_i \geqslant 0\right) \rightarrow \sum_{1}^{n} c_i y_i \geqslant z \qquad (2)$$

The algorithm can be used on formula (2), eliminating all the variables y_i, $1 \leqslant i \leqslant n$. The formula equivalent to (2), with z as the only variables must express the same mathematical statement expressed in (2), namely that z is less than or equal to the minimum cost, c, if it exists. If no solution exists there are 2 possibilities: The constraints may be inconsistent, or else they may include such values of y_i so that the sum $\Sigma c_i y_i$ is unbounded below. In the former case, formula (2) will reduce to TRUE and in the latter case to FALSE.

Let us perform some of the algorithm on formula (2). Eliminating $\rightarrow$, we get

$$(\forall y_1)\cdots(\forall y_n)\neg\left(\mathop{\&}_{j=1}^{m}\sum_{i=1}^{n} a_{ij}y_i = b_j \;\&\; \mathop{\&}_{1}^{n} y_i \geqslant 0\right) \vee \sum_{1}^{n} c_i y_i \geqslant z$$

Changing universal quantifiers to existential quantifiers and eliminating negations we then get

$$\neg(\exists y_1)\cdots(\exists y_n)\left(\mathop{\&}_{j=1}^{m}\sum_{i=1}^{n} a_{ij}y_i = b_j \;\&\; \mathop{\&}_{1}^{n} y_i \geqslant 0 \;\&\; \sum_{1}^{n} c_i y_i < z\right) \qquad (3)$$

Formula (3) has the form: negation followed by a string of n existential quantifiers followed by a kernel consisting of a conjunction of m equalities and $n + 1$ inequalities. Allowing $y_i \geqslant 0$ as an atom according to observation (2) above, we see that the kernel is trivially in disjunctive normal form as a pure conjunction. Also, when one quantifier, $(\exists y_n)$ is eliminated according to the modified algorithm, the resulting formula is still a pure conjunction. Thus, there is never any need to expand into disjunctive normal form, as each quantifier is eliminated in turn.

We give an example, which is given as an exercise on the simplex algorithm in Gass [3].

A. Minimize $2x_1 - 3x_2 + 6x_3$ subject to the constraints

$$\begin{aligned} 3x_1 - 4x_2 - 6x_3 &\leqslant 2 \\ 2x_1 + x_2 + 2x_3 &\geqslant 11 \\ x_1 + 3x_2 - 2x_3 &\leqslant 5 \\ x_j &\geqslant 0 \quad j = 1, 2, 3 \end{aligned}$$

To set up this example for the simplex algorithm, according to the reasonably standard version given in Gass, we convert A to the following equivalent problem.

B. Minimize $2x_1 - 3x_2 + 6x_3 + wx_6$ subject to the constraints

$$3x_1 - 4x_2 - 6x_3 + x_4 = 2$$
$$2x_1 + x_2 + 2x_3 - x_5 + x_6 = 11$$
$$x_1 + 3x_2 - 2x_3 + x_7 = 5$$
$$x_j \geqslant 0 \quad j = 1, 2, 3, 4, 5, 6, 7.$$

Here we require two slack variables, x_4 and x_7, an excess variable, x_5, and an auxiliary variable, x_6. Also, the weight w in the objective function, $2x_1 - 3x_2 + 6x_3 + wx_6$, must be larger than any fixed number so that x_6 will vanish in the so-called phase 1 of the simplex algorithm. We observe here that it is unnecessary to go from statement A of the example to form B when using our algorithm to determine the minimum cost. Statement A can be used directly in formula (3) as follows:

$$\neg(\exists y_1)(\exists y_2)(\exists y_3)(3y_1 - 4y_2 - 6y_3 \leqslant 2 \,\&\, 2y_1 + y_2 + 2y_3 \geqslant 11 \,\&\, y_1 + 3y_2 - 2y_3 \leqslant 5 \,\&\, y_1 \geqslant 0 \,\&\, y_2 \geqslant 0 \,\&\, y_3 \geqslant 0 \,\&\, 2y_1 - 3y_2 + 6y_3 < z)$$

Note that the order of the three quantifiers is irrelevant so any of the variables y_1, y_2, y_3 can be eliminated first. We can therefore choose a y_i that will lead to the minimum number of resulting atoms. We do this by counting the number of times each y_i appears positively on either side of an inequality. We thus get Table 1.

TABLE 1

	LEFT	RIGHT
y_1	3	2
y_2	1	4
y_3	1	4

We see that the choice of y_1 will yield $3 \times 2 = 6$ atoms from $3 + 2 = 5$ atoms. Either y_2 or y_3 will yield only 4 atoms from 5 atoms. So, at least at the first elimination, the formula will expand less if we choose y_2 or y_3.

Let us pick y_2. The only inequality in which y_2 appears positively on the left is $y_1 + 3y_2 - 2y_3 \leqslant 5$. We eliminate y_2 between this inequality and the others containing y_2 to obtain the new formula:

$$\neg(\exists y_1)(\exists y_3)(y_1 - 2y_3 \leqslant 2 \,\&\, 5y_1 + 8y_3 \geqslant 28 \,\&\, y_1 \geqslant 0 \,\&\, y_1 - 2y_3 \leqslant 5 \,\&\, y_3 \geqslant 0 \,\&\, 3y_1 + 4y_3 < z + 5)$$

The same considerations as above lead to the choice of y_3 to be eliminated next:

$$\neg(\exists y_1)(5y_1 < z + 9 \,\&\, y_1 < 2z - 18 \,\&\, y_1 \geqslant 0 \,\&\, 5y_1 < z + 15 \,\&\, 3y_1 < z + 5)$$

Finally, eliminating y_1, we get

$$\neg(0 < z + 9 \,\&\, 0 < 2z - 18 \,\&\, 0 < z + 15 \,\&\, 0 < z + 5)$$

which reduces to

$$\neg(z > -9 \,\&\, z > 9 \,\&\, z > -15 \,\&\, z > -5)$$

or $\neg(z > 9)$ or $z \leqslant 9$. Therefore the minimum cost exists and is equal to 9.

We emphasize here that the linear programming example is used as a familiar problem for illustrative purposes. There are two drawbacks in using this method on typical linear programming problems.

1. The formulation given above is used to determine the minimum cost but not the complete solution. Of course, the minimum cost can be used to verify an optimal solution or to determine exactly how far from optimal any feasible solution is.

2. The main drawback is the lack of efficiency especially compared to the high efficiency of the simplex algorithm. The low efficiency of our algorithm is due to the expansion of the formula upon eliminating quantifiers.

Let us consider formula (3) above. The kernel is a conjunction of m equalities and $n + 1$ inequalities. As seen in step 8 of the algorithm, when a quantifier is eliminated such that one of the conjuncts is an equality containing the quantified variable, there is no increase in the number of conjuncts. On the contrary there is a decrease of one. So, in general, m quantifiers can be eliminated using the equalities and resulting in a formula with $n - m$ quantifiers and $n + 1$ inequalities. Now each successive elimination of a quantifier from a formula with k inequalities can lead to a formula with $(k/2)(k/2) = k^2/4$ inequalities in the worst case, when the occurrences on the left and right are balanced. It appears that the worst case is also the most likely case and deviations of a few inequalities do not help much.

So, if $n = 8$ and $m = 4$ then we could end up with 1,562,500 inequalities. This seems the largest size that can be handled with this method in its present form.

We tried the algorithm on a dairy-feed diet problem due to Waugh [9] and presented in Gass [3]. Here there were 10 variables and four inequalities; i.e. 10 feeds were combined to meet 4 minimum requirements at a minimum cost. The problem as initially stated proved too big for the program. However, conversion to the dual problem, which had 4 variables and 14 inequalities, allowed determination of the minimum cost. Even so, it took 12 hours on the PDP-10 under a LISP compiled version of the algorithm.

The use of a linear programming example, thus, is not to advocate use of

this algorithm for linear programming, but to show that it can be used on practical problems.

The point we would like to emphasize to skeptical readers is that although the EAR algorithm may not be efficient in given areas, its generality allows application to various problems with no programs specific for each problem area.

For example, we can do non-linear programming whenever it is actually piecewise linear. Let us take the case of a piecewise linear objective function. These problems are not direct applications of the simplex algorithm. The optimal solutions may not even occur at the extreme points of the constraints. For example, suppose we wish to minimize the maximum of x_1 and x_2 under the constaints $x_1 + x_2 = 1$, $x_1 \geqslant 0$, $x_2 \geqslant 0$. Here, the optimum value of the objective function is $\frac{1}{2}$ as opposed to a value of 1 at the extreme points.

Piecewise linear functions can not only be expressed in terms of maximum and minimum as above, but also, equivalently, in terms of & and $\vee$. The above example can be stated in EAR as follows:

$$(\forall y_1)(\forall y_2)(y_1 + y_2 = 1 \;\&\; y_1 \geqslant 0 \;\&\; y_2 \geqslant 0 \rightarrow y_1 \leqslant z \vee y_2 \leqslant z)$$

This is of course equivalent to $z \geqslant \frac{1}{2}$ which can be verified by the algorithm.

General non-linear programming can also be approximated by using Chang [1]. He shows how to do piecewise linear curve-fitting in n-dimensions. After using Chang's method to determine piecewise linear approximations, one can express the problem in EAR.

REFERENCES

1. Chang, C. L. Fuzzy Algebras, Fuzzy Functions, and Their Application to Function Approximation. Heuristics Laboratory Report, Division of Computer Research and Technology, N.I.H., July, 1970.
2. Collins, G. E. Tarski's Decision Method for Elementary Algebra. Summer Institute for Symbolic Logic. Cornell University, Communications Research Div., Inst. for Defense Anal., 1957.
3. Gass, S. I. Linear Programming. Third Edition, McGraw-Hill, 1969.
4. Hodes, L. Dense Linear Order and Linear Inequalities. IBM Research Report RC-908, 1963.
5. Hodes, L. Programming Languagues, Logic and Cooperative Games. Presented at Symposium for Symbolic and Algebraic Manipulation, March, 1966. Abstract in Comm. of ACM, August, 1966.
6. McCarthy, J., et al. LISP 1.5 Programmer's Manual, M.I.T., 1963.
7. Rogers, H. An Example in Mathematical Logic. *Amer. Math. Monthly*, **20** (November, 1963), 929–945.
8. Tarski, A. *A Decision Method for Elementary Algebra and Geometry*. Second Edition, University of California, 1951.
9. Waugh, F. V. The Minimum-Cost Dairy Feed. *J. Farm Econ.* (August, 1951).

Received June 1961

Artificial Intelligence 3 (1972), 165–174