Propagating Qualitative Values through Quantitative Equations

DEEPAK KULKARNI
STERLING SOFTWARE
ARTIFICIAL INTELLIGENCE RESEARCH BRANCH
MS 269-2
NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035-1000

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA-92-38

Propagating Qualitative Values through Quantitative Equations

Deepak Kulkarni
Sterling Software
MS 269-2, AI Research Branch
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

In most practical problems where traditional numeric simulation is not adequate, one need to reason about a system with both qualitative and quantitative equations. In this paper, we address the problem of propagating qualitative values represented as interval values through quantitative equations. Previous research has produced exponential-time algorithms for approximate solution of the problem. These may not meet the stringent requirements of many real time applications. This paper advances the state of art by producing *a linear-time algorithm that can propagate a qualitative value through a class of complex quantitative equations exactly and through arbitrary algebraic expressions approximately.* The algorithm was found applicable to Space Shuttle Reaction Control System model.

Content Areas: Qualitative Reasoning, Constraint Based Reasoning

1. Introduction

This paper presents results on combining traditional numeric models with qualitative models. A variety of AI programs use qualitative knowledge. Expert-defined rules often reason about variables with 'low', 'medium' and 'high' values. Qualitative reasoning techniques such as QSIM [Kuipers, 1984, 1986], and QP theory [Forbus, 1984] infer a behavior of a device from its qualitative models defined using qualitative variables, equalities and inequalities. *A qualitative variable can take a qualitative value, i.e., an interval separated by landmark values.* Knowledge of many physical systems like Space Shuttle subsystems takes the form of qualitative models of some components and quantitative models of others [Robinson, 1992]. In general, a qualitative version of a quantitative model (e.g. y *increases monotonically with* x for $y = 3x + 4$) has less information, and may not be adequate to solve a problem at hand. For example, the equation in model fragment M2 in figure 1 can not be converted into any meaningful qualitative model. The model in the figure is an example of a system that has two qualitative model fragments (M1 and M3) and one quantitative model fragment (M2). The methods for combining quantitative knowledge with qualitative knowledge are of fundamental importance to be able to use artificial intelligence techniques in applications involving models like that described in figure 1.

The system whose model is shown in figure 1 could be assumed to be working if particular values of S, T, U and O (e.g., S= very low, T = very low, U = low, and O = high) are consistent with the model. We can reason qualitatively about M1 to find that P = high will be consistent with O = high. Similarly, we can reason about M3 to find that R will be low if U is low. One would still need to test if the set of values { S= very low, T= very low, P = high, R = low} are consistent with M2. We can check this consistency if we have an algorithm to solve the following problem:

P1. Given a quantitative expression relating y with $x_1, x_2, ..., x_n$ and interval values[1] of $x_1, x_2, ... x_n$, what is the interval value of y within a specified precision?

The required precision would vary from one problem to another. In our example, we may need to know if P is low or high for checking it against the qualitative

[1] A variable has an interval value [a b] if it can take any value in the interval but can never take a value outside the interval.

model M1. If P has a sensor associated with it, the precision needed may be the least count of measurements of the sensor. If this sensor measured P to be [16 17], we would need to know if the set of values { S= very low, T= very low, P = [16 17], R = low} is consistent with M2. In general, expressions like M2 may have more than just a few variables. This paper presents results of the research associated with the above described problem (P1).

O, P, R, S are positive real interval variables.
very low = [1 2], low = [2 8], nominal = [8 16], high= [16 24]

Model Fragment M1
if P is high, then O is high.
if P is low or nominal, then O is nominal.

Model Fragment M2

$$P = \left\{ \frac{R^2 - S}{R(S^3 + S)} \right\}^2 - \left\{ \frac{R^2 - S}{R(S^3 + S)} \right\} + T^2$$

Model Fragment M3
If U is low or nominal, R is low.
If U is high, R is nominal.

Problem: Are the values S= very low, T= very low, U= low and O= high consistent with the above model?

Figure 1: A model with qualitative and quantitative equations.

In the next section, we overview related research. Third section defines the terminology used. Forth section presents results on computational intractability of the general interval propagation problem. Fifth section discusses conditions under which interval propagation can be done in linear time, and present an algorithm to

do so. Sixth section discusses utility of this algorithm in practical applications, and the final section is the conclusion.

2. Background

In this section, we will briefly describe important AI work on problem P1. (The reader is referred to [Davis, 1987] for a detailed survey of use of interval labels in constraint propagation systems.) There are two approaches to infer values of variables from a set of equations: *symbolic manipulation* approach and *direct solution* approach. In symbolic manipulation approach, the equations are simplified to answer a question. For example, if $y = x^3 - 3x^2 + 3x - 1$ and $x = 3$, one could simplify y to be $(x - 1)^3$ by manipulating the symbolic form of the equation and then find y to be $2^3 = 8$. MINIMA (Williams, 1988) work falls in this category. In the direct solution approach, an algorithm is used to compute the result directly from an equation without symbolic manipulation. Thus $y = 3^3 - 3.3^2 + 3.3 - 1 = 8$ using real arithmetic.

Interval algebra lacks the properties such as existence of additive and multiplicative inverses and distributive property [Struss, 1988]. As a result, it is not possible to have a powerful symbolic manipulation system like MACSYMA for interval variables. Williams [1988] presents a hybrid algebra that allows both signs and reals, but not arbitrary intervals and shows that it has strong mathematical properties. As a result, he was able to develop a system called MINIMA that can do symbolic manipulation on a set of equations defined over reals and signs (+ - 0).

Given the lack of a symbolic manipulation system for arbitrary intervals, a straightforward algorithm often used to evaluate an arithmetic expression approximately is to evaluate an operand at a time. This evaluation overestimates true results, e.g., consider $y = x^3 - 3x^2 + 3x - 1$. *If $x = [0, 1]$, what is the value of y?* The set of values of y for $x \in [0\ 1]$ is $[-1\ 0]$, as the expression for y is an expansion of $(x - 1)^3$. Evaluation of one operand at a time would give y to be [0 1] + [-3 0] + [0 3] + [-1 -1] = [-4 3]. As another example, consider the evaluation of P in figure 1 for the values $S = [1\ 2]$, $R = [2\ 8]$ and $T = [1\ 2]$. The set of values of P based on all possible combinations of values of S, R and T is $[0.75\ 3.9525]$. Evaluation based on one operand at a time is $[-14.75\ 252.0375]$. Notice that the range of actual evaluation is not significantly larger that of any of inputs, but the particular scheme of approximate evaluation overestimates the result widely. In general, the one-operand-at-a-time evaluation may be undefined due to division by zero or may

give intervals with arbitrarily large width even for simple functions with finite real evaluation.

Additional inference rules can reduce this large overestimation in some cases where one-operand-at-a-time evaluation is used. Quantity Lattice [Simmons, 1986] deduces comparative relationships from a set of relationships and arithmetic expressions. For example: if $A = [2\ 4]$, $B = [1\ 4]$, $D = A - B$, *is* $D > A$? Here, we have $D = A - B = [-2\ 3]$ and $A = [2\ 4]$. Thus, with one-operand-at-a-time evaluation alone, the system will estimate D to be more than A in some cases. To address the problem of overestimation in one operand-at-a-time scheme used, Quantity Lattice uses a relational arithmetic to infer relationships from arithmetic expressions. In the above example, it uses an inference rule "$a > 0 \Rightarrow X - a < X$" to infer that $D = A - B$ will imply $D < A$. *i. e. D is never more than A.* The scheme was developed to handle simple relational inference encountered in common-sense reasoning, and is not meant to be used with equations encountered in science applications. (Is *p high i.e.,* $p >$ *16 for given interval values of R, S, and T?* in figure 1.) Literature reports other work which does not use this one-operand-at-a-time scheme and uses polynomial time algorithms to do exact interval propagation [Davis, 1988]. However, these algorithms are limited to constraints of the form of unary predicates, order relations, and linear equations. The focus of this paper is on tmore complex equations like that in figure 1.

An approach to evaluate such expressions approximately has been taken by researchers in interval computations [Alefeld and Herzberger, 1983]. They have developed a method of evaluation based on mean value theorem that can evaluate $f(x = [a\ b])$ so that the evaluation error is at the most $[b - a]^2$. However, the average case complexity of execution of this method is exponential with the number of variables in an expression. Monte Carlo simulation is another approach that can produce a subset of the result along with a probability distribution, but it is computationally even more expensive for multi-variable functions. Sachs [1988] describes a system that uses computationally less expensive methods such as one-operand-at-a-time first before applying more expensive methods such as the iterative approximation method.

In summary, previous work has used efficient one-operand-at-a-time scheme that does not produce good approximations on typical engineering expressions and iterative approximation methods that produce good approximation, but are computationally expensive. (Section 6 gives concrete details of a Space Shuttle

application where we found previously reported methods not to meet its requirements.) Thus, these are not efficient for processing multi-variable models like that in figure 1 in real time.

3. Terminology

In this section, we define real intervals and expressions based on the treatment by Neumaier[1990]. An interval is a set of the form $[a\ b] = \{x \varepsilon R \mid a <= x <= b\}$. We will use the convention that interval variables be represented by capital letters, and real variables be represented by small letters.

Elementary operations {+, -, *, /, **} are defined on the set of intervals as $A\ o\ B = \{ x\ o\ y \mid x\ \varepsilon\ A$ and $y\ \varepsilon\ B\}$ for all intervals A, B such that $A\ o\ B$ is defined for all $x\ \varepsilon\ A$ and $y\ \varepsilon\ B$.

We extend any real function to interval arguments as $f(A) = \{ f(x)\ \mid x\ \varepsilon\ A\}$ for all intervals A such that $f(x)$ is defined for all $x\ \varepsilon\ A$.

An algebraic arithmetic expression E is defined as below:
E = *variable, real constant*
$E = E\ o\ E$
$E = f(E)$
Real evaluation of expression $f(X_1,X_2,\ldots X_n)$ is defined as $\{f(val_1, .. val_i .. val_n)$ */val_i belonging to Xi).*

4.Intractability of algebraic interval expression evaluation

The problem for solving interval expressions exactly is known be NP-Hard for algebraic expressions [Yemini, 1979], and undecidable for expressions with both arithmetic operators and transcendental functions [Richardson, 1968]. The author [Kulkarni, 1992] has shown that the problem for solving these with limited approximation (problem P1) formulated in one of the forms below is also NP-Hard for algebraic expressions.

Absolute Approximation: Given an expression E defined over X_1, X_2, ... X_n with the interval values of val_1, val_2, ... val_n, find an interval that approximates E with error less than $\delta = 2^{cn}$, where c is a positive integer constant.

Relative Approximation: Given an expression E defined over X_1, X_2, X_n with

the interval values of val_1, val_2, val_n, find an interval that overestimates E = [a b] to be a proper subset of [a- |a|, b + |b|], but a superset of [a b].

5. An Efficient Algorithm for Propagating Qualitative Values

Given the intractability of **P1** for the class of algebraic expressions, it would be a good strategy to identify classes of expressions that typically occur in applications and for which there are efficient algorithms to propagate interval values. In this section, we will identify one such class of expressions for which propagation can be done exactly in linear time. Our philosphy is similar to that of Simmons (1986), but our algorithm is applicable to a class of complex expressions which can be defined precisely. First, we will describe certain properties of expressions that simplify propagation, and then present the algorithm.

5.1 Properties that Simply Value Propagation

P1. **Monotonicity**: Consider a function $f(s\ r) = \frac{R - \frac{S}{R}}{S + S^3}$ defined for positive S and R on the interval S= *[1 2]* and R = *[1 2]*. For this function we have a monotonic relation between f and the inputs R and S, i.e., $\frac{df}{dR}$ is always non-negative and $\frac{df}{dS}$ always negative. Thus, we conclude that maximum value of f will correspond to the maximum value of R and minimum value of S. Also, minimum value of f will correspond to the minimum value of R and maximum value of S.

f(1 2)= 0.75
f(2 1)= -0.1
Therefore f is [-0.1 0.75] for R = [1 2] and S = [1 2]

This property has been used in the BOUNDER program [Sachs, 1988], and is described formally in the following lemma has been proved in [Neumaier, 1990].

Lemma: If f has arguments w_1,..w_m, x_1,.. x_n, y_1,.. y_o z_1,.. z_p; with Wi (i = 1,.. m) = (w_{i1}, w_{i2}); Xi (i = 1,...n) = (x_{i1}, x_{i2}); Yi (i = 1,.. o) = (y_{i1} y_{i2}); and Zi (i = 1,.. p) = (z_{i1}, z_{i2}) and $\frac{df}{dx_i}$ is zero, $\frac{df}{dy_i}$ belongs to *[0 inf)*, $\frac{df}{dz_i}$ belongs to *(minf 0]*, then $f(W_1 ,..W_m, X_1,.. X_n, Y_{1,..} Y_o Z_1,.. Z_p) = f(W_1 ..W_m, x_{11},.. x_{m1}, y_{11},.. y_{o1}, z_{12},.. z_{p2}) \cup f(W_1 ..W_m, x_{11},.., x_{m1}, y_{12},.. y_{02}, z_{11}, ..z_{p1})$.

P2. **Simple Expressions**: Some expressions that do not exhibit the monotonicity property can be solved using their mathematical properties. Many functions (for example, the trigonometric and the Bessel functions) decrease and increase monotonically over a sequence of adjacent intervals. The points of separation of these intervals are known. Extrema of the function can only occur at these points. In some cases, one may use further knowledge about the function to simplify the computation, e.g., the values of the function at the local maxima may be identical.

E.g., $sin\ [0\ 20] = [-1\ 1]$

$f(x) = x^2 - x$, $f([0\ 3]) = (-.25\ \ 6)$ based on local extrema.

P3. **Substitution Principle**: If a solvable subexpression, S, occurs in an expression, and S is defined in terms of variables that do not occur in other parts of the expression, then S can be replaced by a variable that has the evaluated value of S. For example, $f = sin(x) * z + sin^2(x)$ with $x = [0\ 20]$, $z = [1\ 20]$ will reduce to: $f = s * z + s^2$ with $s = [-1\ 1]$, $z = [1\ 20]$. Now f is monotonic w.r.t s, whereas f is not monotonic w.r.t. x in the original expression. so we can use monotonicity property to evaluate the latter expression. In general,
$g(h(X_i...X_m), Y_1 .. Y_n)$ can be evaluated by computing first $S = h(X_1, .. Xm)$ and then $g(S, Y_1 .. Y_n)$. This rule allows us to decompose the evaluation for f into two functions with smaller number of arguments.

P4. **Singly Occuring Variables**: If the interval variables in an expression occur only once, then its real evaluation equals its one-operand-at-a-time evaluation.[Neumaier, 1990]

Computing an expression with unordered use of these properties can require time exponential in the number of variables in the expression. *The central contribution of this paper is an algorithm that uses the above properties to evaluate expressions in linear time.*

5.2. Algorithm for value propagation

We will now present a linear-time algorithm for propagation of interval values based on these properties.

ALGORITHM A

Input: An Expression E

Interval values of variables in E.

Output: Resultant value of E

1. Replace a subexpression S by a variable Vs, if S is a solvable subexpression[2] that is defined in terms of variables that do not occur in other parts of the expression E. Solve S recursively using algorithm A and assign the result to variable Vs.

2. Replace variables that occur only once by constants.

3. For variables that vary monotonically with E, substitute appropriate extrema to calculate maxima and minima of the expression.

4. If the overall expression is solvable by A, it would have been transformed into a constant or a simple solvable expression[3] by steps 1 to 3. Solve this expression using a standard method.

Figure 2: An Algorithm for Value Propagation

The steps 1, 2, 3, and 4 are justified by properties P3, P4, P1, and P2 respectively.

Evaluate $\{\frac{R^2-S}{R^2(S^3+S)}\}^2 - \frac{R^2-S}{R^2(S^3+S)} + T^2$ for S = [1 2], R= [2 8], and T= [1 2]

1. Decide to evaluate $\{\frac{R^2-S}{R^2(S^3+S)}\}$ and T^2

   Evaluate $\{\frac{R^2-S}{R^2(S^3+S)}\}$

   1. Not applicable
   2. Not applicable
   3. Maxima = Expression (R=8, S=1) = 0.49219

      Minima = Expression (R= 2, S=2) = .05

   Evaluate T^2

[2]For the clarity of description, we have omitted details about how solvability of a subexpression or an expression is determined and these are described in the appendix.

[3]We define a simple solvable expression as any expression that can solved in number of steps that is a linear function of number of variables e.g. a linear or a quadratic expression.

1. Not applicable

2. T^2 is [1 4]

Equation is transformed to v^2 - v + [1 4] with v = [0.05 0.49219]

2. Not applicable

3. Not applicable

4. The solution is [.75 3.9525] based on evaluation of local extrema.

Figure 3: Execution trace of Algorithm A on expression P

Now, we will examine how this algorithm computes the results of equation for P in figure 1 for S = [1 2], R = [2 8], and T = [1 2]. Figure 2 shows the execution trace of algorithm A on expression P. Step 1 will decide to execute the algorithm recursively on the solvable subexpression $V = \frac{R^2-S}{R^2(S^3+S)}$ and $W = T^2$. Next, V does not have any solvable subexpression or singly occurring variables and so will go to step 3. In step 3, the monotonic relation is used to compute maxima and minima for the expression. V increases monotonically with R and decreases monotonically with S. V will be maximum for R = 8 and S = 1; and minimum for R = 2 and S = 2. This turns out to be [.05 0.49219] to be the values of V. In the recursive call to evaluate W, step 2 is used to evaluate it to be [1 4]. Now the algorithm to evaluate V^2 - V + W. Next, this becomes V^2 - V + [1 4]. Now the algorithm evaluates V^2 - V using the fact that its only extrema is at .5. The result is that expression has the value [-.25 -.0475]. Adding [1 4] to it gives the overall result of [.75 3.9525].

Now we will prove that A is a linear time algorithm.

Lemma 1: Algorithm A can be executed in *16* c* (n-1)* number of steps where *n* is the number of occurrences of variables in an expression, and c is a constant such that the number of steps needed execute the step 4 in the algorithm A <= c* n and c >= 1 (Refer footnote 3).

Proof:

First, we will note that steps 2 and 3 in the algorithm A can be executed in less than 2n steps.

We will prove Lemma 1 by induction on the proposition that *S(n) <= c* (n-1)*, where *S* is the number of steps needed to execute algorithm A; *n* and *c* are as defined above.

*Initialization: S(2) <= 16*c .*

Induction Argument:

Now we will show if the proposition is true for $j < n$, then it will also be valid for $j = n$. Consider execution of an expression *S(n)* where n > 2.

if step 1 is used to solve *S(n)* then

let k be the number of occurrences of variables in the subexpression

S(n) = S(k) + S(n-k+1) for step 1

*S(n) <= 16c * (k- 1)* + 16 c * (n -k) <= 16 $c^*(n - 1)$

otherwise

S(n) <= *(4 + c) * n* that is needed to execute steps 2 to 4.

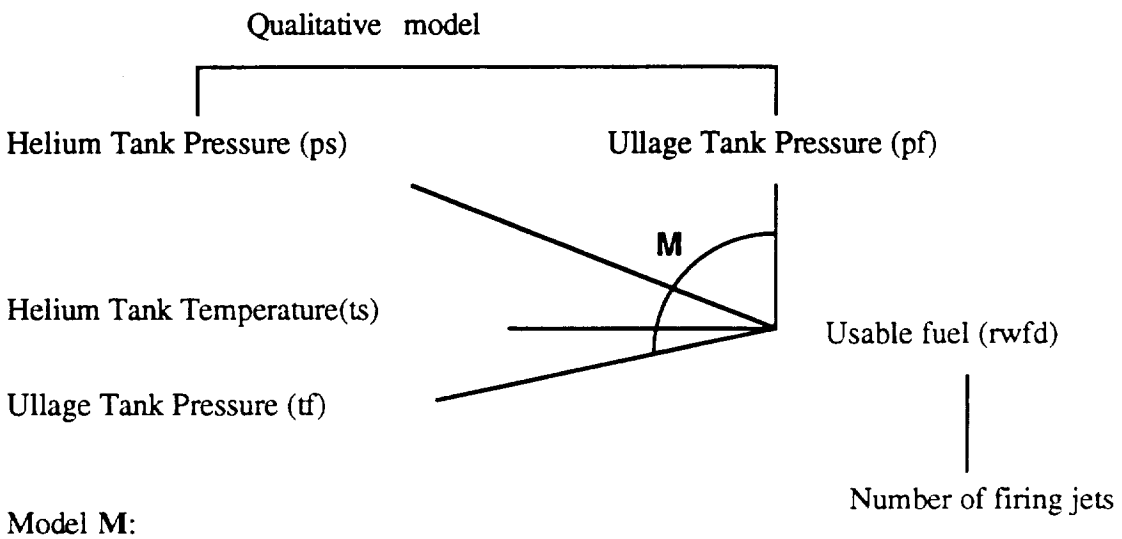
<= 16 c * *(n* - 1) as $n > 2$

Thus, we have proved if the proposition is valid for $j < n$, then it will also be valid for $j = n$.

6. Practical Utility of the algorithm:

We used the algorithm to develop a model-based diagnosis system for Space Shuttle Reaction Control System that has qualitative models of some components and quantitative models of others. Figure 4 shows the RCS pressurization system model. A qualitative model related Helium Tank Pressure and Ullage Tank Pressure. A set of quantitative equations relate Helium compressibility factor (zs), volume of Helium system(vhs), weight of Helium system (whs), ullage volume (vhu), density (rhof), and fuel (rwfd) to Helium and Ullage Tank Pressures. One can compose an expression for rwfd in terms of ps1, ps2, ts, pf1, pf2, and tf from all the equations described in the model M. By using algorithm A on this expression, one can compute an exact interval value for rwfd from interval values of ps1, ps2, pf1, pf2, ts, and tf using a few computations. This, in turn, can be used to check if the system is working normally or not. We found that one-operand-at-a-time and Quantity Lattice methods do not give acceptable approximations. Our theoretical analysis is that iterative approximation method would need 10^6 subroutine calls in the avarage case, as 6 variable intervals would be divided into 10 intervals to achieve desired accuracy. This would not meet the real-time requirements of the application. RCS example illustrates that Algorithm A can propagate interval values through expressions encountered in some applications. However, it may not be able to do so for expressions encountered in some real-time applications. In these applications, one can approximate expressions by ones belonging to the above-defined class off-line and then use algorithm A to produce good approximations in real time. In contrast to previous

known methods for approximate solutions such as Monte Carlo simulation or Mean value theorem based evaluation, this approach produces a solution *in linear time in real time* satisfying the processing time requirement of many applications. In contrast, in situations where computational time is not a concern, the Mean value theorem based method can produce arbitrarily accurate solution, where running algorithm A on an approximate functional fit would not.



Model **M**:
ps = (ps1 + ps2) /2
pf = (pf1 + pf2) /2
zs = 1 + b1 * ps * ts^(-b1)
vhs = vham * [1 + ps*a]^3 +vhli
whs = (ps*vhs) / (zs* R* ts)
pfv = e^ (d1 - (d2/tf) + (d3*tf))
vhu = (g * [whi - whs] *tf *R) / (pf - pfv)
rhof = c1 -c2*tf +c3*pf
rwfd = 100 * (rhof * [vp- vtp -vhu] -wtp) /wfda
b1, c1, c2, c3, d1, d2, d3, R, vhli, vham, a, vtp, vp, wtp, wfda, whi, pfv, g are constants

Figure 4: The model of Shuttle RCS pressurization system

---

7. Conclusion

This paper advances the state of art by producing *a linear-time algorithm that can propagate a qualitative value through a class of complex quantitative equations exactly and through arbitrary algebraic expressions approximately.* The algorithm was found applicable to Space Shuttle Reaction Control System model. It will be an important direction of future work to identify other classes of quantitative equations for which polynomial time propagation algorithms exist.

8. Appendix

This appendix describes an algorithm for finding if an expression is solvable by algorithm A.

Task: To check solvability of expression S at the height h given the list of all solvable expressions at height <= h-1 (SOLVABLE-EXPRESSION LIST) and all monotonic relations between subexpressions and variables.

Reprsentation: Expression S is represented as a tree with each operator as the father of the operands. Some leaf variables may be identical symbols.

Symbols:
L: variable list
S: Given Expression

1. Make list L of variables that occur more than once in the expression S and are not monotonic with S.
2. Store any subexpression that satisfies the following two conditions in SOLVABLE_SUBEXPRESSION LIST.
   a. It occurs in SOLVABLE-EXPRESSION LIST
   b. Variables occuring in this subexpression do not occur in the rest of the expression S.
3. Repeat (3a) until L is empty

(3a) IF L is non-empty
THEN
Pick a variable V from L
Use algorithm C to find a subexpression SE-V that contains all the occurances of variable V in S.
IF SE-V belongs to the SOLVABLE_SUBEXPRESSION LIST
THEN
Add a symbol denoting SE-V to the list L and redefine expression S in terms of SE-V.
Eliminate all variables occuring in this expression LS from L.
ELSE
Remove the variable V from the list L.

4. Drop from expression S all variables that are monotonic with S.
   If the new expression is of a type that is solvable directly, then add it to SOLABLE-EXPRESSION LIST.

Algorithm C:
1. Set i = 1
2. Find all the subexpressions of height i that contain variable V.
3. IF these are identical subexpressions or if there is only one such subexpression, THEN return this subexpression as the answer
   ELSE
   set i = i+1 and go to step 2.

References

Alefeld, G. and Herzberger, J. (1983) *Introduction to Interval Computations.*

Academic Press, New York.

Davis, E. (1987) Constraint Propagation with Interval Labels. *Artificial Intelligence*. 32:281-331.

Forbus, K. D. (1984) Qualitative Process theory. *Artificial Intelligence* 24:85-168.

Garey, M.R. and D.S. Johnson (1978) *Computers and Intractability: A Guide to Theory of NP-completeness,* H. Freeman. San Francisco.

de Kleer, J. and Brown, J. (1984) A qualitative physics based on confluences. Artificial Intelligence 24: 7-83.

Kozen, D. and Yap, C.K. (1985) Algebraic cell decomposition in NC, in: Proceedings of 26th Symposium on the Foundations of Computer Science. 515-521.

Kuipers, B. J. (1984). Commonsense reasoning about causality: deriving behavior from structure. *Artificial Intelligence* 24: 169-204.

Kuipers, B. J. (1986). Qualitative simulation. *Artificial Intelligence.* 29: 289-338.

Kuipers, B. J. and Berleant, D. (1980). A smooth integration of incomplete quantitative knowledge into qualitative simulation. Tech Report Al 90-122. Al Laboratory, University of Texas at Austin.

Kulkarni, D. (1992) Approximate Interval Expression Evaluation is NP-Hard, Artificial Intelligence Research Branch, Forthcoming *NASA Ames Technical Report.*

Neumaier, A. (1990) *Interval methods for systems of equations.* New York: Cambridge University Press.

Richardson, D. (1968) Some undecidable problems involving elementary functions of a real variable, J. *Symbolic logic.* 33: 514-520.

Robinson, P. (1992) A report on RCS diagnosis project. Forthcoming *NASA Ames Technical Report.*

Sachs, E. (1987) Hierarchical Reasoning about inequalities. pp 649-654, *AAAI-87*: Seattle, Washington.

Simmons, Reid (1986) Common-sense arithmetic reasoning, pp 118- 124. *AAAI-86,* Philadelphia, PA.

Struss, Peter (1988) Mathematical Aspects of Qualitative Reasoning. in *International Journal of AI in Engineering.*

Williams, Brian (1988) MINIMA: A symbolic approach to qualitative algebraic reasoning In *AAAI*-88: Los Altos, CA. pp 264-269.

Yemini, Y. (1979) Some theoretical aspects of position-location problems, in *Proceedings 20th Symposium on the Foundations of Computer Science* 1-7.

Academic Press, New York.

Davis, E. (1987) Constraint Propagation with Interval Labels. *Artificial Intelligence*. 32:281-331.

Forbus, K. D. (1984) Qualitative Process theory. *Artificial Intelligence* 24:85-168.

Garey, M.R. and D.S. Johnson (1978) *Computers and Intractability: A Guide to Theory of NP-completeness*, H. Freeman. San Francisco.

de Kleer, J. and Brown, J. (1984) A qualitative physics based on confluences. Artificial Intelligence 24: 7-83.

Kozen, D. and Yap, C.K. (1985) Algebraic cell decomposition in NC, in: Proceedings of 26th Symposium on the Foundations of Computer Science. 515-521.

Kuipers, B. J. (1984). Commonsense reasoning about causality: deriving behavior from structure. *Artificial Intelligence* 24: 169-204.

Kuipers, B. J. (1986). Qualitative simulation. *Artificial Intelligence*. 29: 289-338.

Kuipers, B. J. and Berleant, D. (1980). A smooth integration of incomplete quantitative knowledge into qualitative simulation. Tech Report Al 90-122. Al Laboratory, University of Texas at Austin.

Kulkarni, D. (1992) Approximate Interval Expression Evaluation is NP-Hard, Artificial Intelligence Research Branch, Forthcoming *NASA Ames Technical Report.*

Neumaier, A. (1990) *Interval methods for systems of equations.* New York: Cambridge University Press.

Richardson, D. (1968) Some undecidable problems involving elementary functions of a real variable, J. *Symbolic logic.* 33: 514-520.

Robinson, P. (1992) A report on RCS diagnosis project. Forthcoming *NASA Ames Technical Report.*

Sachs, E. (1987) Hierarchical Reasoning about inequalities. pp 649-654, *AAAI-87*: Seattle, Washington.

Simmons, Reid (1986) Common-sense arithmetic reasoning, pp 118- 124. *AAAI-86,* Philadelphia, PA.

Struss, Peter (1988) Mathematical Aspects of Qualitative Reasoning. in *International Journal of AI in Engineering.*

Williams, Brian (1988) MINIMA: A symbolic approach to qualitative algebraic reasoning In *AAAI*-88: Los Altos, CA. pp 264-269.

Yemini, Y. (1979) Some theoretical aspects of position-location problems, in *Proceedings 20th Symposium on the Foundations of Computer Science* 1-7.