A Computer Application to Study Engineering Projects at the Early Stages of Development

M. H. Gedig
M.A.Sc Structural Engineer AGRA Coast Inc.
1515 KingswayPort Coquitlam, B.C., Canada V3C 1S2

and

Dr. S. F. Stiemer
Dr.-Ing, P.Eng. Professor, Department of Civil Engineering
University of British Columbia 2324 Main Mall Vancouver, B.C., Canada V6T 1Z4

ABSTRACT

This paper describes a computer software application, the Qualitative Engineering System (QES), which the engineer can use to perform qualitative and semi-quantitative analysis of preliminary engineering designs. In engineering practice, many situations arise in which the engineer wishes to perform a logical, objective comparison between conceptual or preliminary design options. Although there exist many applications which can be used to perform detailed numerical analysis to justify detailed final designs, relatively few useful programs are available to validate designs at the preliminary stages. The early stages of design are characterized by higher levels of uncertainty than the latter stages. Established qualitative and semi-quantitative reasoning techniques may be used to detail with uncertainty and incomplete information in a sound, logical manner. The QES application utilizes a unified framework, which is used to implement a number of qualitative and semi-quantitative reasoning techniques. This paper gives an explanation of qualitative and semi-quantitative analysis in the context of the QES application. In addition, the paper gives some practical examples of how the QES program can be used in the engineering environment.

KEYWORDS

Computer application, design uncertainty, conceptual design, decision evaluation, qualitative reasoning, semi-quantitative reasoning, interval technique.

1 Introduction

An accurate evaluation of conceptual design alternatives holds many potential benefits for the engineering practice. Unworkable designs can be eliminated early, and more resources can be dedicated to the concepts, which are most practical and efficient. It is accepted that the earlier a decision is made in a project the greater the financial implications of the decision. Often a large amount of capital and time is expended at the latter stages of a project in order to make an unworkable design workable. A better understanding of the feasibility of a conceptual design allows the engineer to be more competitive, because the inherent risk of working in new territory can be decreased.

A fundamental problem in evaluating conceptual design alternatives is the relatively high level of uncertainty present in design concepts. Many of the currently available analytical tools are not effective in coping with the type of uncertainty, which exists in design concepts. Numerical analysis tools are indispensable in the practice of engineering today. The finite element method, for example, is a widely used systematic approach to predicting the behaviour of systems in a wide range of applications, including thermal analysis, structural analysis, mechanical analysis and electromagnetics. In most of the available numerical analysis packages, input must be specified in a rigid, structured format. If these tools are used in the conceptual stages, many of the required input fields must be estimated, and there is usually no systematic way of

determining how the accuracy of the program output relates to uncertainty in the input parameters.

Probability methods are useful in evaluating the level of risk in a completed design. At the conceptual design stages however, these methods are cumbersome and suspect because of the large amount of input required. Much of the input used for describing the probability distribution functions for various design parameters would have to be estimated at the early stages of a design. The time and effort required preparing the input, and the value of the results given the uncertainty in the output both detract from the worth of probability methods in such an application.

Expert systems have been successfully used in the past for reasoning with information that is characterized by a high level of uncertainty. The problem with expert systems is that they are notoriously "brittle" - they are domain specific and tend to be of limited use outside their narrow range of expertise. Expert systems do not generally reason with established physical principles, which apply over a number of domains. There is a great deal of incentive in engineering to develop new, innovative, and cost-effective solutions to problems. Innovation often involves interdisciplinary approaches to problem solving which are incompatible with the domain-specific nature of expert systems.

An alternative approach to evaluating conceptual designs is to use the techniques of qualitative and semi-quantitative analysis. The systematic study of qualitative techniques developed out of artificial intelligence research in the early eighties. Qualitative and semi-quantitative techniques may be used to perform sound reasoning about physical systems, which are characterized by a high degree of uncertainty.

2 The QES Program

The goal of this research is to develop a tool to evaluate conceptual designs using proven, logical, and sound reasoning techniques. The tool must be able to cope with a high level of uncertainty, and flexible enough to accept more information and adapt to the changing level of uncertainty as a project progresses. In order to fulfil the goals of this research, the QES program was developed. The program uses a number of qualitative and semi-quantitative analysis techniques. These techniques were described in detail in a previous paper and will be only briefly covered here.

In developing a practical engineering tool for reasoning with partially-specified information, it would be desirable to incorporate both the power of abstraction inherent in qualitative methods, and the elegance of semi-quantitative methods such as interval analysis [10]. The *Qualitative Engineering System* (QES), developed as a part of this research, features a coherent framework, which integrates both qualitative and semi-quantitative reasoning techniques.

The level of uncertainty present in an engineering project is not static. In the early, conceptual stages, there is more uncertainty than in the final stages. A flexible reasoning framework is needed which is capable of using new information, as it becomes available, and updating previous conclusions. QES uses the powerful notion of constraints as a means of expressing engineering problems.

2.1 Constraints

Much of an engineer's knowledge is best expressed in terms of what is, or conversely, what is not allowed. Functional requirements, material limitations, labour considerations, environmental characteristics and the laws of physics can all be expressed in the language of constraints. Many of the computational tools currently used in engineering demand a rigid dichotomy between input and output. Furthermore, the input format is often inflexible, requiring that input data be completely specified before computation can proceed. In contrast, QES is able to accept as input

only the information that is known at a particular time. Information, both qualitative and quantitative, can be added any stage of a project.

2.2 Program Structure

QES is effectively a quantity database [1], which is implemented as a constraint network. The database contains *expressions* and *relations*, where the expressions represent the nodes of the constraint network and the relations represent the constraints. The relations may be either arithmetic in nature, or may represent inequalities (*ordinal* relations). Expressions are capable of representing qualitative information, such as whether a parameter is less than, greater than, or equal to zero. Expressions also can store semi-quantitative information, which is expressed in the form of intervals. A number of techniques, both qualitative and quantitative, are used to reason with the quantity database. These techniques include label propagation using both intervals and signs, interval arithmetic, graph search, and constraint inference.

This section discusses the main components of the QES representation: intervals, expressions, variables, constants, equations, inequalities and systems of equations. In QES, constraints are represented as equations and inequalities. Both equations and inequalities are composed of expressions, which consist of variables and constants. Expressions evaluate to quantities. The interval representation is used to express quantities to varying degrees of precision. QES was implemented in the programming language C++, an object-oriented language. In the object-oriented approach, data and the procedures, which use this data, are encapsulated as entities called objects. In the following discussion, the components of QES are formulated as objects.

2.3 Intervals

Interval methods are well-developed techniques, which enable one to reason on the range of values of quantities. Intervals are flexible enough to represent vague qualitative values as well as precise numeric values. For this reason, the interval representation was chosen as to represent quantities in QES. In QES an interval is represented by a data structure which contains two members, the upper and lower bounds (Figure 1). The interval structure includes the procedures add, multiply, negate, and reciprocal.

interval		
data members	***name***	***type***
	`negativeBound`	*bound*
	`positiveBound`	*bound*
methods	***name***	***arguments***
	`add`	*interval*
	`multiply`	*interval*
	`negate`	
	`reciprocal`	
	`isLess`	*interval*
	`isGreater`	*interval*
	`isEqual`	*interval*

Figure 1: Interval data structure.

2.4 Bounds

A bound is a structure which consists of three fields: the `value`, which is either a floating-point number or one of the symbolic constants -∞, +∞, or *NaN* (Not a Number); the `inclusion` field, which is TRUE if the bound includes the value, and FALSE otherwise; and the `type` member, which indicates whether the bound is at the lower or upper extent of the interval. Using

this representation, QES can reason with open, closed, or half-open intervals. As shown in Figure 2, the bound structure also encapsulates procedures which define bound operators (`add,multiply,negate,reciprocal`). Since operators are defined on the bounds, the implementation of the interval operators is simplified. For example, the interval `add` procedure takes the following convenient form in C++:

```
add(Interval toInterval)
{
      lowerBound.Add(toInterval.lowerBound)
      upperBound.Add(toInterval.upperBound)
}
```

bound		
data members	***name***	***type***
	`value`	*number, ±∞, NaN*
	`inclusion`	*TRUE, FALSE*
	`type`	*UPPER, LOWER*
methods	***name***	***arguments***
	`add`	*bound*
	`multiply`	*bound*
	`negate`	
	`reciprocal`	
	`isLess`	*bound*
	`isGreater`	*bound*
	`isEqual`	*bound*

Figure 2: Bound data structure.

Additional processing is required for the `multiply` and `reciprocal` functions. For these operators the ordinal relations `isLess`, `isGreater`, and `isEqual` are used to order the bounds correctly.

expression		
data members	***name***	***type***
	`domain`	*list of intervals*
	`constraint`	*interval*
	`type`	*UNARY, BINARY*
	`operator`	*NEGATE, RECIPROCAL, ADD, MULTIPLY, SUBTRACT, DIVIDE*
	`leftChild`	*expression*
	`rightChild`	*expression*
	`parents`	*list of expressions*
	`relations`	*list of relationLinks*
methods	***name***	***arguments***
	`constrain`	
	`relativeConstrain`	*relation, interval, width*
	`add`	*expression*
	`multiply`	*expression*
	`subtract`	*expression*
	`divide`	*expression*
	`negate`	
	`reciprocal`	

Figure 3: Expression data structure.

2.5 Expressions

Just as algebraic expressions may take on a number of seemingly different forms, so may expressions in QES. Expressions represent a fundamental component of QES, as several other components of the system derive their behaviour from expressions. The expression data structure is shown schematically in Figure 3. The definition of the expression is recursive in nature, as the data structure contains a reference to itself. For illustration, consider the simple algebraic expression $A + (B \times C)$. This expression may be represented as a binary tree, as shown in Figure 4. The node labeled '×' has two *children*, the variables *B* and *C*. The node labeled '+' also has two children, the variable *A* and the expression $B \times C$. Since variables may be considered simple expressions, variables and expressions are essentially the same in QES. Representing expressions as trees proves to be a convenient construct for many of the techniques, which are used to reason with the quantity database.

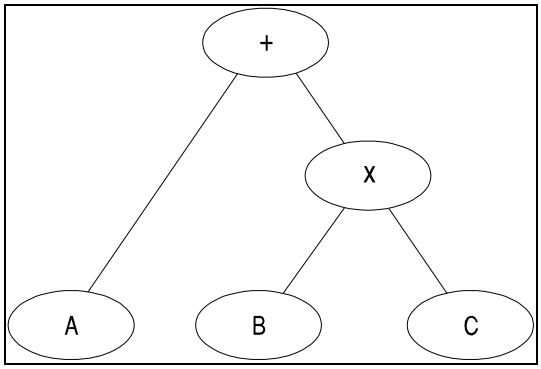

Figure 4: Graphical representation of expressions

Two of the main components of the expression data structure, shown in Figure 3, are the `constraint`, and the `domain`. The `constraint` is a single interval, while the `domain` is a set of intervals. By default, the constraint interval holds the value [-∞, +∞], which expresses a complete ignorance of the value of the expression. As more information about various quantities and relations is entered into the system, the constraint intervals become narrower. The domain set is interpreted slightly differently for variables than for expressions. In QES the domain of a variable is analogous to the quantity space representation in other qualitative reasoning frameworks, such as Kuipers' QSIM [6]. Kuipers defined the quantity space as a finite, ordered set of landmark values:

$$l_1 < l_2 < \ldots < l_k \,,$$

which can be represented as the set of alternating closed and open intervals:

$$\{[-\infty, -\infty], (-\infty, l_1), [l_1, l_1], (l_1, l_2), [l_2, l_2], \ldots, [l_k, l_k], (l_k, +\infty), [+\infty, +\infty]\}.$$

In QES the landmarks are replaced by valid assignments to the value field of the bound structure. For example, the simple quantity space {-, 0, +} may be described by the domain set {[-∞, 0), [0, 0], (0, +∞]} in QES. Expressions are also used to represent numeric constants in the quantity database. The numeric constant zero is a default member of the database. The domain of the constant zero contains the single closed interval, which identified with the real number zero, {[0, 0]}.

An expression may have two, one or no children. A binary expression, such as $A + B$, has two children, while a unary expression such as $-X$ has only one child. In the expression structure shown in Figure 3, the `type` field indicates whether the expression is unary or binary. The `operator` member indicates what type of operator is applied to the child expressions. The

available unary operators are *negate* and *reciprocal*, while the binary operators are *add*, *subtract*, *multiply* and *divide*. The `leftChild` and `rightChild` fields point to the child expressions. The `rightChild` field is unused for unary expressions, while both are unused when the expression represents a variable. The variables are the 'leaves' of the expression tree, while each expression, which has no parent, is a 'root'. The child data members act as the hierarchical or arithmetic links between expressions. These links are bi-directional, as each child has a corresponding link with their 'parent'. The `parents` data field holds a list of pointers to the parents of the expression. The network of expressions cannot properly be called a tree, since an expression may have multiple parents, for example, $A + B \times A$. Having noted this, the tree analogy is still a convenient descriptive construct which reflects the hierarchical nature of the interconnections between expressions.

The expression tree representation is similar to what Forbus terms a 'tree of functional dependencies', which represents the relationship between qualitative proportionalities. In the tree of functional dependencies, quantities are linked by qualitative proportionalities, independent quantities are at the leaves, and the dependent quantity is at the root. When the expression tree is used for qualitative analysis, the relationships between expressions feature a similar dependence, however, for other operations such as interval propagation, the links are essentially bi-directional.

2.6 Relations

Along with arithmetic links, relational links represent the means by which constraints may propagate through the quantity database. Relational links (called *relationLinks* in QES) serve two purposes: they constrain expressions to form equations and inequalities, and they enable search techniques to be used to infer additional information about ordinal relationships. Inequalities are an important means of capturing fundamental qualitative distinctions, without sacrificing the expressive capabilities of algebraic expressions. Relation links represent the ordinal relation between two expressions. The *relationLink* data structure, shown in Figure 6, contains three fields: one for each expression and another for the ordinal relation.

relation		
data members	*name*	*type*
	`value`	=, <, >, ≤, ≥, ≠, ?

Figure 5: Relation data structure.

Ordinal relations are represented by the *relation* data structure, presented in Figure 5. Relations can take on one of seven values: =, <, >, ≤, ≥, ≠, and the *unknown* relation, represented by the symbol ?.

relationLink		
data members	*name*	*type*
	`leftExpression`	*expression*
	`rightExpression`	*expression*
	`ordinalRelation`	*relation*

Figure 6: Relation link structure.

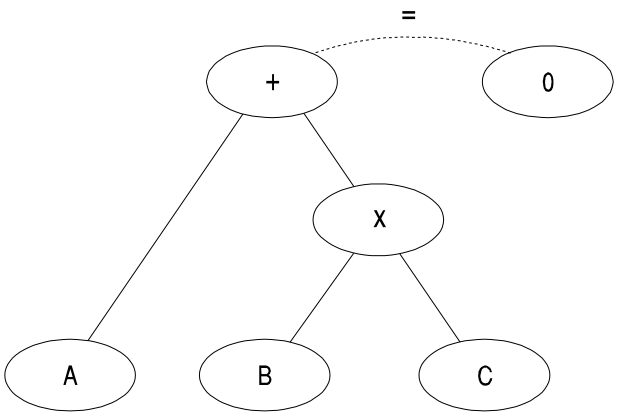

Figure 7: Graphical representation of an equation.

As an example of relation links, consider the equation A + (B × C) = 0. This equation is depicted in Figure 7, where the relation link is shown as the dotted arc between the number zero and the expression node labeled '+'.

2.7 Systems of Equations

When there is more than one equation present in the system, a data structure is used to maintain a list of the applicable equations. The main components of the *equationSystem* structure are shown in Figure 8. The *equationSystem* structure is required in the constraint satisfaction procedure. This procedure is explained in detail in the next section.

equationSystem		
data members	*name*	*type*
	`expressions`	*list of expressions*
	`solutions`	*list of interval sets*
	`arguments`	*list of variable names*

Figure 8: Equation system structure.

3 Qualitative Analysis

In QES, the domains of qualitative variables consist of a finite set of intervals. The most common set represents the domain of signs {[-∞, 0), [0, 0], (0, +∞]}, which may be abbreviated by {-, 0, +}. The domain of signs will be used to look at how QES solves constraint equations which are expressed as qualitative variables. The tree structure of expressions is exploited in the solution procedure, which uses constraint-satisfaction methods.

Each of the nodes of the constraint network may be considered a constraint. This representation of a constraint network is different to that given by Mackworth [8]. In Mackworth's network, the variables in the constraint-satisfaction problem are the nodes. Loops represent unary constraints on a variable and directed arcs indicate binary constraints. In solving constraints consisting of equations, this representation is unsuitable, because equations can consist of more than two variables. The simple example given earlier, $A + (B \times C) = 0$, involves three variables: *A*, *B*, and *C*. For this reason, the alternate representation of Freuder [3] is used. In this system, nodes represent constraints, while arcs are used to link constraints having common variables. QES uses a constraint synthesis process to build up successively higher levels of consistency in the constraint network.

The structure of the expression tree provides a convenient framework in which to construct and solve constraints. Constraints are built up from variables in an incremental fashion, following

the expression tree from leaf to root. The following simple example will be used to illustrate the constraint satisfaction process:

$$F_1 + F_2 = 0 \quad (1)$$

$$F_1 - F_3 = 0. \quad (2)$$

The problem is to determine which assignments to the qualitative variables F_1, F_2 and F_3 satisfy the two constraints (1) and (2). We begin building the expression tree by entering three nodes into the constraint network; one for each variable. This network is shown in Figure 9. Each variable is initially unconstrained, so the constraint interval of each is [-∞, +∞]. The contents of the domain are shown below each expression.

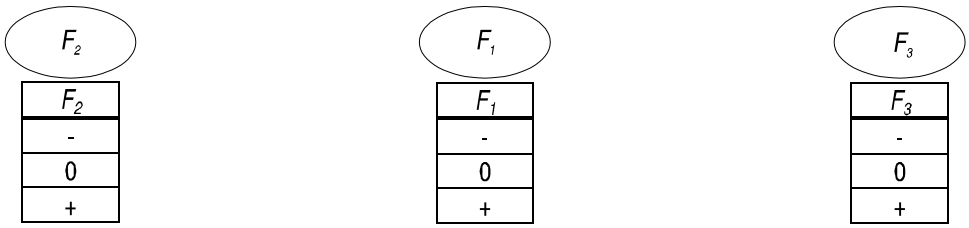

Figure 9: Initial constraint network.

Table 1: Qualitative addition [X] + [Y].

+		[X]			
		+	0	-	?
[Y]	+	+	+	?	?
	0	+	0	-	?
	-	?	-	-	?
	?	?	?	?	?

If we add the additional constraint $F_2 > 0$, the values - and 0 can be removed from the domain of F_2. The expression $F_1 + F_2$ will be constructed first. Addition is a constraint, which is satisfied if the values assigned to the arguments are consistent with the result shown in the qualitative addition table, Table 1. Note that in Table 1, the character '?' represents a complete lack of information about the result. Multiplication, subtraction, division, negation and reciprocal functions can all be defined as constraints, similar to addition. To find the variable assignments, which satisfy the addition constraint, the Cartesian product of the domains of the child expressions is generated, and the corresponding value of the sum is determined according to Table 1.

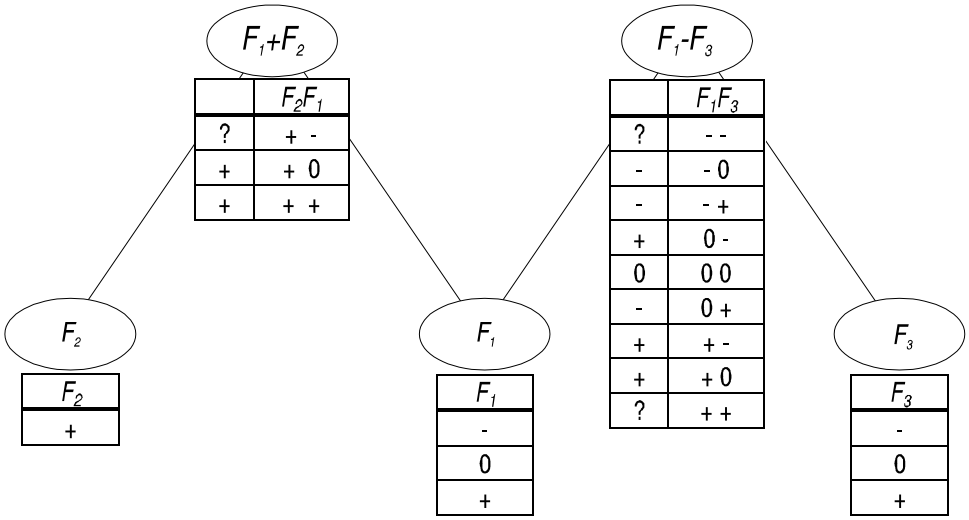

Figure 10: Constraint network with expression links added.

In Figure 10, all possible combinations of the domains of F_1 and F_2 are shown in the table under the node for expression $F_1 + F_2$. The sum of the two values is shown in the left column of the table. A similar table is generated for the expression $F_1 - F_3$. The left-hand column of each table contains the elements of the domain set for the constraint expression, while the right-hand column holds the values of the arguments of the expression.

The next step in solving the system of constraints is to enforce the equality constraint, which will constrict the domains of the two expressions $F_1 + F_2$ and $F_1 - F_3$. As discussed earlier, relation links represent ordinal relations between expressions. Two relational links are added to the constraint network, as shown in Figure 11, linking the two expressions to the constant expression zero. The constraint interval of a constant expression is always equal to the domain, which is [0, 0] in this case. The interval [0, 0] is intersected with the constraint interval of the expressions $F_1 + F_2$ and $F_1 - F_3$. Since the constraint interval of each is initially $[-\infty, +\infty]$, their new values are [0, 0]. To enforce the constraint at the two expressions, the constraint interval is further intersected with each element of their domains. When intersection results in the empty set, the domain element is deleted.

The updated domains are shown in Figure 11, which shows that $F_1 + F_2$ has one legal instantiation while $F_1 - F_3$ has three. At this point, node consistency has been achieved at the two expression nodes $F_1 + F_2$ and $F_1 - F_3$. Note that a relational link also exists between the variable F_2 and the constant zero. For clarity, this link is not shown in Figure 11.

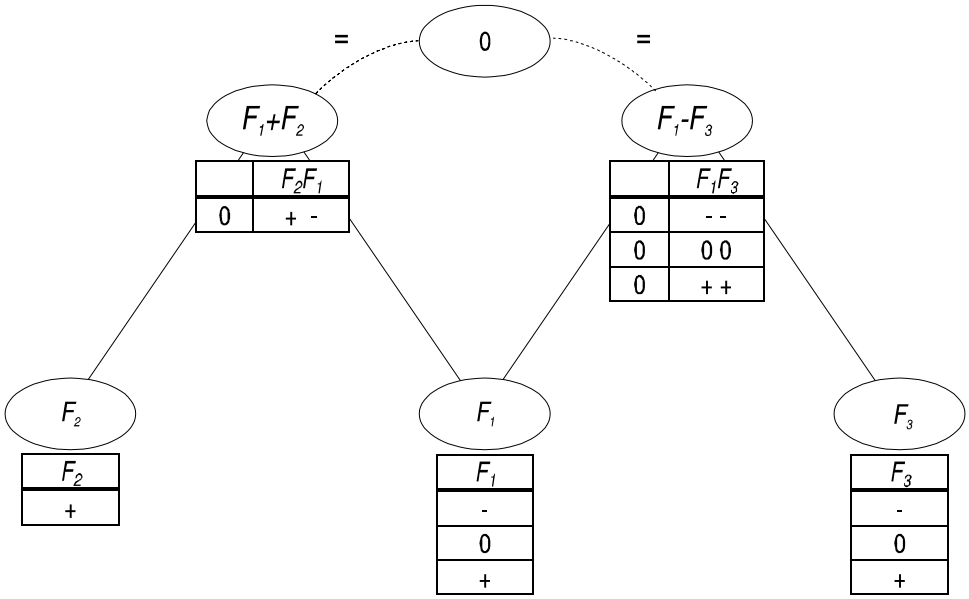

Figure 11: Constraint network after node consistency procedure.

The two expressions $F_1 + F_2$ and $F_1 + F_3$ now contain partial solutions to the problem. These partial instantiations must be combined to find the complete solution. The simple approach is to find all combinations of the domains of the two expressions, which correspond to consistent assignments of variables. A more efficient method is employed by QES, which uses the concept of *arc consistency*. In QES arc consistency is enforced on pairs of equations. The simple arc consistency AC-1 [8], is used here. QES maintains a list of all equations in the system using the *equationSystem* data structure (Figure 8). Each time an equation is created, it is made consistent with each of the existing equations. If elements in the domain of an expression are deleted when arc consistency enforced between two expressions, the procedure is repeated until no further deletions are necessary. In the example problem, the expression $F_1 + F_2$ is first added to the *equationSystem* structure. Since it is the first equation to be added, no deletions are performed. The second expression is then added, and the two equations are made arc consistent. The uppermost box in Figure 12 represents the *equationSystem* structure with its links to each of the equations.

Looking at expression F_1 - F_3, one of the instantiations involves the assignment $F_1 = 0$ and another involves $F_1 = +$. The domains corresponding to these two assignments are deleted, because the only valid assignment for F_1 in the expression $F_1 + F_2$ is $F_1 = -$. Because deletions have occurred, the two expressions are checked for consistency again. Since no further deletions are possible, the arc consistency procedure is halted. The complete solution to the problem is found by simply merging together the domains of the two binary expressions. This results in the single legal instantiation:

$F_1 = -$

$F_2 = +$

$F_3 = -$.

Although this problem has one solution, this is not generally the case when systems of qualitative constraints are concerned. If the additional constraint on F_2 ($F_2 > 0$) had not been added, there would be three legal instantiations instead of one.

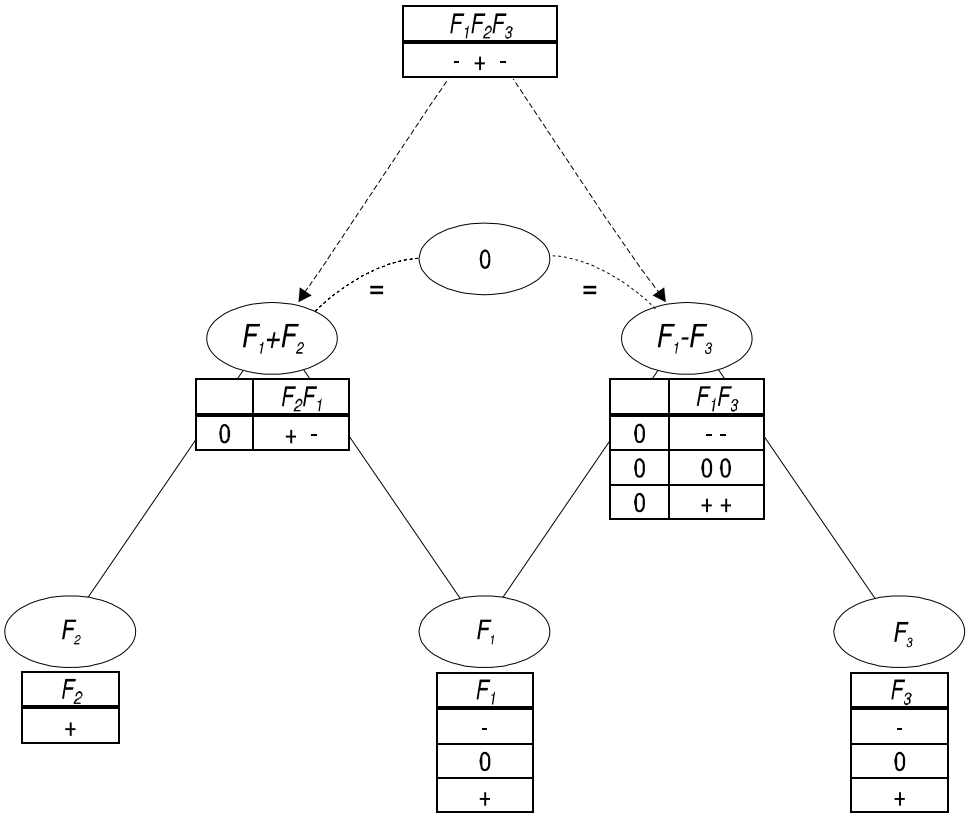

Figure 12: Final consistency network.

3.1 Example from Structural Analysis

An example of qualitative analysis from the field of structural mechanics is given in this section. This example illustrates some of the practical implications of qualitative analysis, as well as some refinements that may be made.

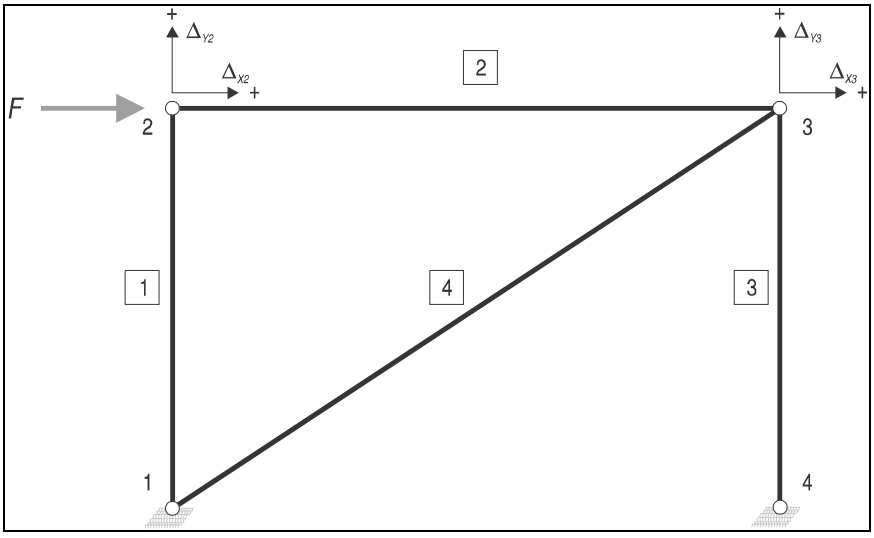

Figure 13: Pin-jointed structure.

The pin-jointed plane structure model shown in Figure 13 will be studied using qualitative analysis techniques. In the figure, the boxed numbers are the element labels and the other

numbers are the node labels. The parameters of this model are the lengths (L_1, L_2, L_3, L_4) and axial stiffnesses (EA_1, EA_2, EA_3, EA_4) of each of the four members, as well as the nodal displacements at the upper left node (Δ_{X2}, Δ_{Y2}), and the upper right node (Δ_{X3}, Δ_{Y3}). The horizontal displacements Δ_{X2} and Δ_{X3} are considered positive if to the right, and the vertical displacements Δ_{Y2} and Δ_{Y3} are positive if upward. A horizontal force, which acts to the right, is applied at the upper left node. One of the uses for such a model would be to estimate the deflected shape of the structure. This is the objective of the qualitative analysis.

The *qualitative* equations, which relate the nodal displacements to the material properties and geometry of the structure, are given as follows:

$$\frac{E_2 A_2}{L_2}\left[\Delta_{X2} - \Delta_{X3}\right] - F = 0 \tag{3}$$

$$\frac{E_1 A_1}{L_1}\Delta_{Y2} = 0 \tag{4}$$

$$\frac{E_2 A_2}{L_2}\left[\Delta_{X3} - \Delta_{X2}\right] + \frac{E_4 A_4}{L_4}\left[\Delta_{X3} + \Delta_{Y3}\right] = 0 \tag{5}$$

$$\frac{E_3 A_3}{L_3}\Delta_{Y3} + \frac{E_4 A_4}{L_4}\left[\Delta_{X3} + \Delta_{Y3}\right] = 0 \tag{6}$$

These qualitative equations are very similar to the usual quantitative equations. The difference is that positive numeric constants have been omitted. Positive constants may be eliminated from products, because the positive sign value acts as the identity for multiplication in the domain of signs.

The input for QES, which is used to solve this problem, is shown in Figure 14. The QES program accepts input in a format that is quite similar in form to the problem specification that has been given here. The variables in the problem are defined as `dx2`, `dx3`, `dy2`, and `dy2`, which have their obvious counterparts in the notation used here. The output produced by QES is shown in Figure 15. The analysis produces four solutions to the problem, the first of which is the correct solution:

Δ_{X2} = + Δ_{Y2} = 0

Δ_{X3} = + Δ_{Y3} = -.

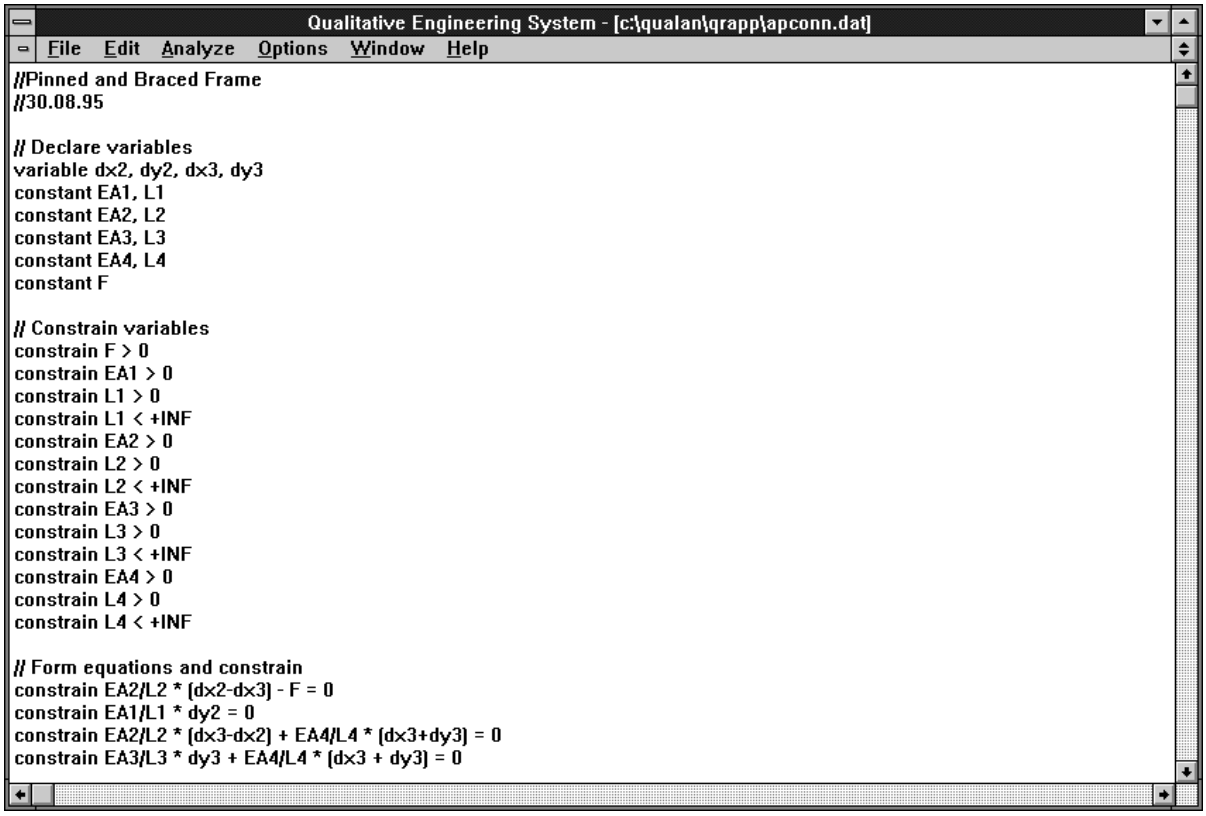

Figure 14. QES input for pin-jointed structure.

The result of this analysis is a typical result in qualitative analysis: multiple solutions are generated but the 'correct' solution to the problem is always contained in the set of solutions. The other solutions are not, strictly speaking, incorrect, because they are correct solutions to the *qualitative* model. A qualitative model is usually a generalization of an associated quantitative model. The process of generalizing a quantitative model allows solutions, which do not satisfy the quantitative equations. One of the prime sources of weakness in qualitative predictions derives from the weakness of the qualitative addition function. Looking at Equation 3, the assignment Δ_{X2} = -, Δ_{X3} = - causes the term (Δ_{X2} - Δ_{X3}) to evaluate to ?. Since the sign ? represents the interval [-∞, +∞], this combination of assignments satisfies Equation 3, even though such values would not satisfy the corresponding qualitative equation. As mentioned previously, the qualitative addition function introduces uncertainty, which tends to propagate through systems of qualitative equations.

One way of strengthening the conclusions drawn by qualitative analysis is to make some use of ordinal relations between qualitative variables [2]. Ordinal relations may be used to reduce the uncertainty caused by the qualitative addition operator. Whenever qualitative addition results in the sign ?, three new cases may be generated to reflect different possible ordinal relations between the arguments involved in the addition operation (Table 2).

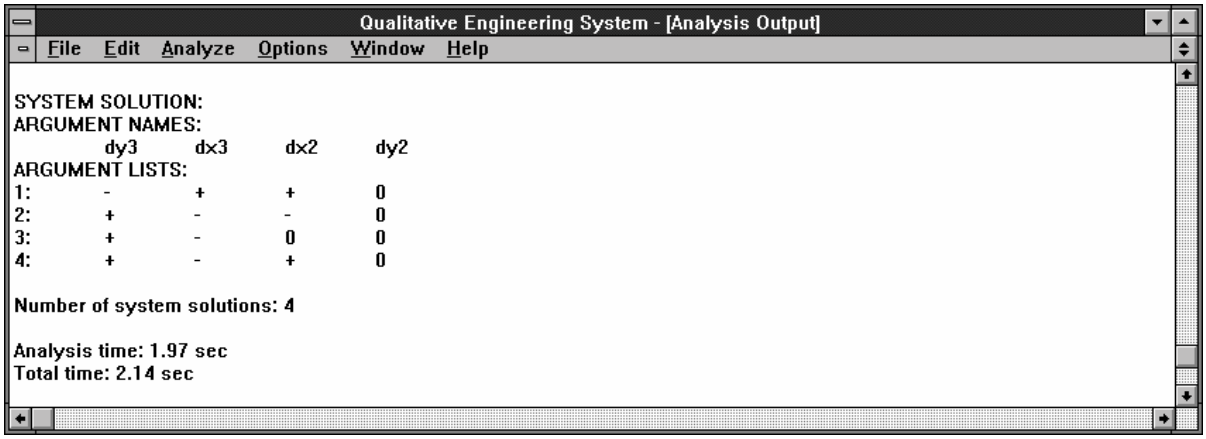

Figure 15. QES output for pin-jointed structure.

Table 2: Qualitative addition with ordinal relations.

+		[X]		
		+	0	-
[Y]	+	+	+	- |X| > |Y| 0 |X| = |Y| + |X| < |Y|
	0	+	0	-
	-	- |X| < |Y| 0 |X| = |Y| + |X| > |Y|	-	-

The advantage in using ordinal relations is that the relations provide additional information that may be used to filter inconsistent variable assignments. The ordinal relations are assertions about the relationship between two variables. This assertion may cause a contradiction at some stage of the solution process, which allows us to delete elements from the partial solution.

In QES, the user has the option of including ordinal relations in the solution procedure. The structural model shown in Figure 13 was analyzed using information about ordinal relations. The result of this new analysis shows that instead of four possible solutions, there is only one valid solution, which corresponds to the expected behaviour for such a structure. In addition to information about the values of the variables, a solution obtained using ordinal relations provides insight into the relationship between the magnitudes of variables. The analysis yields the following additional information:

$|\Delta_{X3}| > |\Delta_{Y3}|$

$|\Delta_{X2}| > |\Delta_{X3}|$

In the previous problem, qualitative analysis lead to a fairly clean, informative solution, considering the minimal amount of specific information that was furnished as input. It is important to note that, in general, a qualitative analysis produces more than one solution. Increasing the number of variables in a problem leads to a rapid increase in the number of combinations, which must be considered, and also, an increase in the number of solutions. The following problem illustrates this concept.

Figure 16 shows the graphical representation of a model of a structure. The model contains three members, each member *i* with associated length L_i, axial stiffness EA_i, and bending stiffness EI_i. As in the previous structural analysis example, a lateral force *F* directed towards the right acts at the upper left joint. The qualitative equations which may be used to derive the deflections from the loading and member properties given here:

$$\frac{E_1 I_1}{L_1^2}\left[\frac{\Delta_{X2}}{L_1} - \theta_2\right] + \frac{E_2 A_2}{L_2}\left[\Delta_{X2} - \Delta_{X3}\right] - F = 0 \tag{7}$$

$$\frac{E_1 A_1}{L_1}\Delta_{Y2} + \frac{E_2 I_2}{L_2^2}\left[\frac{1}{L_2}\left(\Delta_{Y2} - \Delta_{Y3}\right) - \frac{\Delta_{Y3}}{L_2} - \theta_2 - \theta_3\right] = 0 \tag{8}$$

$$\frac{E_1 I_1}{L_1}\left[\theta_2 - \frac{\Delta_{X2}}{L_1}\right] + \frac{E_2 I_2}{L_2}\left[\frac{1}{L_2}\left(\Delta_{Y3} - \Delta_{Y2}\right) + \theta_2 + \theta_3\right] = 0 \tag{9}$$

$$\frac{E_2 A_2}{L_2}\left[\Delta_{X3} - \Delta_{X2}\right] + \frac{E_3 I_3}{L_3^2}\left[\frac{\Delta_{X3}}{L_3} - \theta_3\right] = 0 \tag{10}$$

$$\frac{E_2 I_2}{L_2^2}\left[\frac{1}{L_2}\left(\Delta_{Y3} - \Delta_{Y2}\right) + \theta_2 + \theta_3\right] + \frac{E_3 A_3}{L_3}\Delta_{Y3} = 0 \tag{11}$$

$$\frac{E_2 I_2}{L_2}\left[\frac{1}{L_2}\left(\Delta_{Y3} - \Delta_{Y2}\right) + \theta_2 + \theta_3\right] - \frac{E_3 I_3}{L_3}\left[\theta_3 - \frac{\Delta_{X3}}{L_3}\right] = 0\,, \tag{12}$$

where again the qualitative equations have been obtained from the quantitative equations by deleting constant numeric terms from products. In this problem, the variables are the four translational degrees of freedom Δ_{X2}, Δ_{Y2}, Δ_{X3}, and Δ_{Y3}, and the two rotational degrees of freedom θ_2 and θ_3, where the numeral 'two' in the subscript denotes the upper left joint and the numeral 'three' the upper right joint.

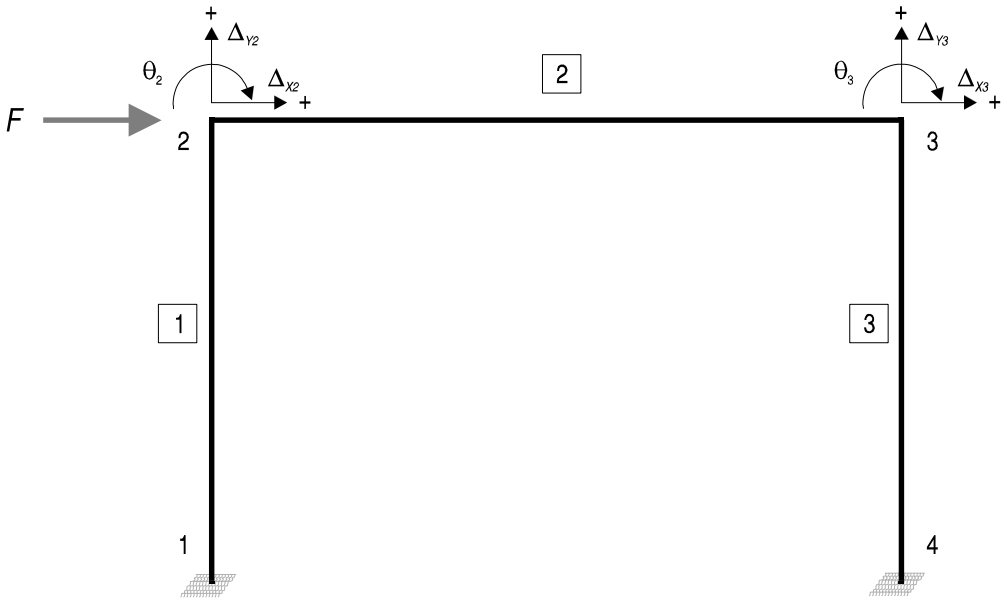

Figure 16: Structural frame model.

An analysis of the system of Equations 7 through 12 using QES resulted in a total of 189 different solutions. This result shows that increasing the number of qualitative variables in the problem leads to considerably weaker qualitative predictions. In this case, the added complexity

of the equations certainly has a detrimental impact on the analysis. In particular, the equations in this example include more complex sums than in the previous example. Even when ordinal relations are considered in the analysis, the number of solutions *increases* to 266, in contrast with the previous problem where the number decreased. The use of ordinal relations increases the number of combinations, which must be considered, but the additional information simply does not cause enough of the assertions to be refuted. Although the 189 solutions resulting from simple qualitative analysis is less than the total number of possible combinations of six variables, each having three values ($3^6 = 729$), we gain little practical insight into the problem. Even though more sophisticated improvements than the use of ordinal relations may be applied to qualitative analysis, it is worthwhile to consider the application of pure qualitative analysis to engineering practice.

In the previous two examples from structural analysis, very little information was specified about the various quantities in the problems. In most problems in engineering, partial knowledge about quantities takes the form of partial numerical values. This observation suggests that, for an engineering application at least, it is more beneficial to pursue a formulation involving partial numeric information, rather than to seek refinements to the pure qualitative representation. This was the direction taken with the QES program. The implications of reasoning with partial numeric information are discussed in the next section.

4 Semi-Quantitative Analysis

In QES, partial quantitative information is incorporated using the interval representation. Integers are a simple, compact and flexible means of representing uncertainty or partially specified numeric information. Since intervals may also be used to represent qualitative information, it is possible to develop a unified framework for representing and reasoning with qualitative and semi-quantitative knowledge. This was the approach used in the QES program, which is meant to support engineering decision-making at different stages of a project: from the initial stages, where qualitative information is more prevalent, to the later stages, which are characterized by primarily quantitative information.

4.1 Numeric Constraint-Satisfaction

The QES program is able to accept constraints in a number of forms, both qualitative and quantitative. Systems of numeric constraints are formulated as numeric constraint-satisfaction problems. The bounds consistency techniques developed by Lhomme [7], were implemented in QES to solve numeric constraint-satisfaction problems. The details of this implementation within the QES framework will be discussed here.

Consistency techniques are applied to numeric constraint-satisfaction problems by associating a dynamic domain with each variable and by propagating this domain through the constraints. The domains of variables are represented by intervals, so that as constraints are propagated from one variable to another, the bounds of the intervals are updated dynamically. This procedure can be examined more closely using the following example. Consider the system of constraints:

$$C_1: Y = X + 1 \qquad C_2: Y = 2 \times X$$

Let the initial domains for the variables X and Y be

$$D_X = [-\infty, +\infty] \qquad D_Y = [-\infty, +\infty].$$

In the constraint network representation of QES, the constraints and variables may be described graphically as shown in Figure 17. The constraint network contains six expressions: X, Y, 1, 2, $X + 1$, and $2 \times X$. In the figure, solid arcs represent arithmetic links, while broken arcs symbolize ordinal relational links.

In the constraint network, changes propagate along relational and arithmetic links. The variable X has an arithmetic link to the expressions $X + 1$ and $2 \times X$ recorded in its *parents* field, while the numeric constants 1 and 2 have links to $X + 1$ and $2 \times X$, respectively. The expressions $X + 1$ and $2 \times X$ both maintain a relation link to the variable Y. Suppose the domain D_X is updated to [0, 10]. The following sequence of events transpires:

- The change in D_X propagates from X to $X + 1$ through a parent link of X.
- The constraint interval of $X + 1$ is updated to [1, 11], using the current values of its children.
- The change in $X + 1$ is propagated through a relation link to Y.
- Variable Y is updated to [1, 11].
- The change in Y causes $2 \times X$ to be updated to [1, 11], because $2 \times X$ shares a relation link with Y.
- X is recalculated using the new value of $2 \times X$ and the right child, the numeric constant 2. The constraint interval of X is updated to [0.5, 5.5]
- The change in X propagates to $X + 1$ by way of the arithmetic link.

The domain changes propagate cyclically through the constraint network, and asymptotic convergence toward the solution ($X = 1$, $Y = 2$) occurs.

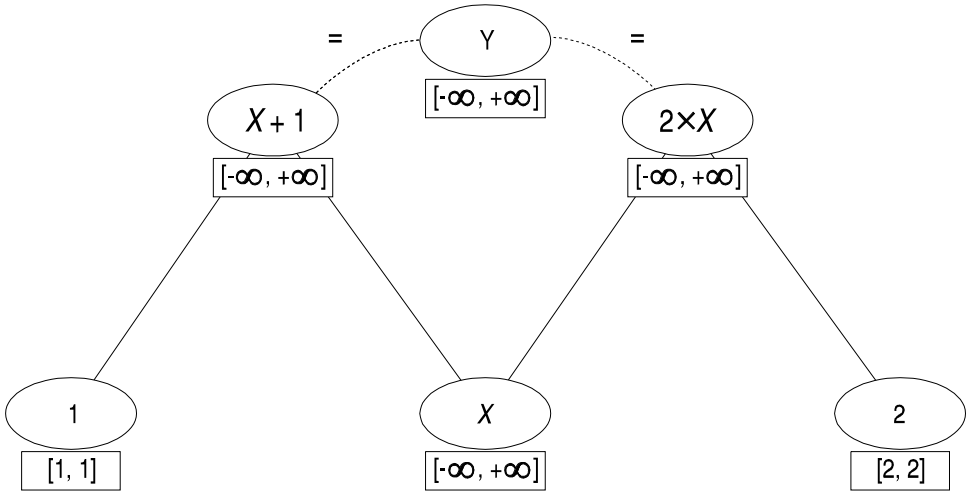

Figure 17: Constraint network representation for numeric constraint problem.

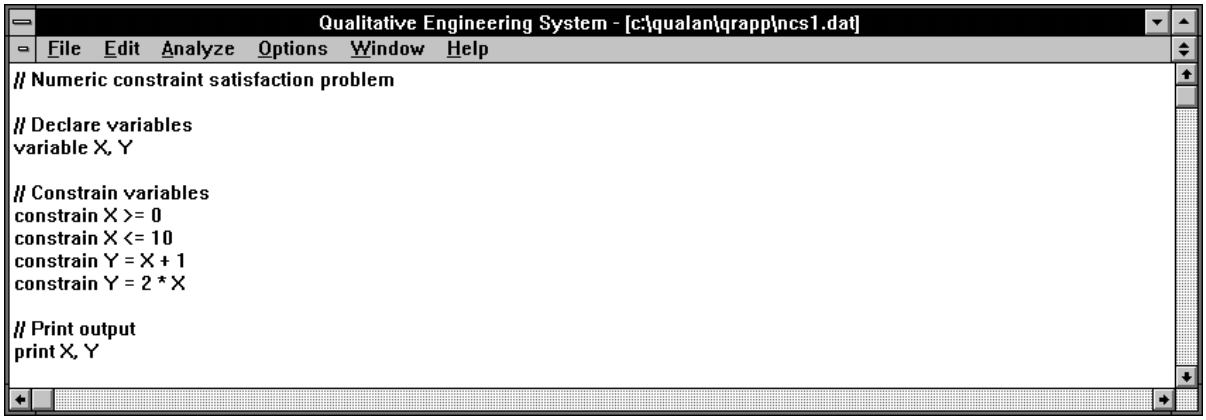

Figure 18: QES input for numeric constraint problem.

The input for QES for this constraint-satisfaction problem is displayed in Figure 18. As shown, interval constraints are represented by inequality constraints, so that the assignment $X = [0, 10]$ is entered as two constraints; $X \geq 0$ and $X \leq 10$. The QES program uses the technique of arc B(w)-consistency [7] to solve numeric constraint-satisfaction problems. The solution progresses according to the sequence discussed above, as shown in Figure 19. The consistency algorithm terminates after about 20 cycles through the network, using specified imprecision w of 1.0×10^{-6}. As shown the final values for the intervals of X and Y are:

$X = [0.999998, 1.000001]$

$Y = [1.999998, 2.000001]$

```
Numeric constraint satisfaction:
Modify X to [0.5,5.5]
Modify (X+1) to [1.5,6.5]
Modify Y to [1.5,6.5]
Modify 2*X to [1.5,6.5]
Modify X to [0.75,3.25]
Modify (X+1) to [1.75,4.25]
Modify Y to [1.75,4.25]
Modify 2*X to [1.75,4.25]
Modify X to [0.875,2.125]
...
Modify X to [0.999998,1.000001]
Modify (X+1) to [1.999998,2.000001]
Modify Y to [1.999998,2.000001]
Modify 2*X to [1.999998,2.000001]
Normal termination
```

Figure 19: Solution sequence for numeric constraint problem.

4.2 Constraint Propagation

In the QES constraint network, three distinct types of constraint propagation are used:

- From an expression to a parent through an arithmetic relation
- From an expression to a child through an arithmetic relation
- From one expression to another through an ordinal relation

Table 3: Constraint propagation to parents

+	$[\underline{X}, \overline{X}] + [\underline{Y}, \overline{Y}] = [(\underline{X} + \underline{Y}), (\overline{X} + \overline{Y})]$
-	$[\underline{X}, \overline{X}] - [\underline{Y}, \overline{Y}] = [(\underline{X} - \overline{Y}), (\overline{X} - \underline{Y})]$
×	$[\underline{X}, \overline{X}] \times [\underline{Y}, \overline{Y}] = [\min(\underline{X}\times\underline{Y}, \underline{X}\times\overline{Y}, \overline{X}\times\underline{Y}, \overline{X}\times\overline{Y}),$ $\max(\underline{X}\times\underline{Y}, \underline{X}\times\overline{Y}, \overline{X}\times\underline{Y}, \overline{X}\times\overline{Y})]$
/	$[\underline{X}, \overline{X}] / [\underline{Y}, \overline{Y}] = (-\infty, +\infty)$ if $(\underline{Y} < 0)$ and $(\overline{Y} > 0)$ $= [\min(\underline{X}/\underline{Y}, \underline{X}/\overline{Y}, \overline{X}/\underline{Y}, \overline{X}/\overline{Y}),$ $\max(\underline{X}/\underline{Y}, \underline{X}/\overline{Y}, \overline{X}/\underline{Y}, \overline{X}/\overline{Y})]$

4.3 Propagation to parents

A change in one of the children of an expression requires the expression to be recalculated. Propagation in this sense is straightforward, given the convenient representation in QES of

expressions as trees. The details of this type of propagation follow directly from interval arithmetic [10]. In the following discussion, the notation of Moore is used for interval arithmetic; an interval is denoted by a capital letter, say X, where $X = [\underline{X}, \overline{X}]$. Given the set of basic constraints available in the QES system, Table 3 summarizes the details of the arithmetic. Note that Table 3 covers only closed intervals. Open and half-open intervals are handled in a similar way.

4.4 Propagation to children

Given a binary expression Z = X ∘ Y, where the symbol '∘' indicates an operator, equality requires that the arguments X and Y be updated when Z is modified. Note that a distinction must be made between the right and left arguments in a binary expression, since, in general, X ∘ Y × Y ∘ X. Table 4 lists the arithmetic which must be performed to update the children of an expression. In Table 4, X denotes the left hand operand and Y the right operand.

Table 4: Constraint propagation to children.

	Expression	*Left*	*Right*
+	$Z = X + Y$	$X = Z - Y$	$Y = Z - X$
-	$Z = X - Y$	$X = Z + Y$	$Y = X - Z$
×	$Z = X \times Y$	$X = Z / Y$	$Y = Z / X$
/	$Z = X / Y$	$X = Z \times Y$	$Y = X / Z$

4.5 Propagation through ordinal relations

The propagation of constraints through relation links is useful for both qualitative and quantitative analysis. This technique becomes even more useful when additional relational constraints may be inferred from existing ones, a procedure called constraint inference. Constraint inference will be described in a following section. Relational constraints, which take the form $X \circ Y$, where X and Y are expressions, are detailed in Table 5. The equality constraint is identical with the intersection operation on intervals. A constraint operation that does not meet the condition listed in Table 5 results in an assignment of the empty set (∅) to each variable involved in the constraint. This result indicates that an inconsistency exists in the constraint network. This information is indicated to the user of the system.

Table 5: Constraint propagation through ordinal relations.

	Relation	*Condition*	*Procedure*
≤	$X \le Y$	$\underline{X} < \overline{Y}$	if $\overline{X} > \overline{Y}$ then $\overline{X} = \overline{Y}$ if $\underline{Y} < \underline{X}$ then $\underline{Y} = \underline{X}$
≥	$X \ge Y$	$\overline{X} > \underline{Y}$	if $\overline{\overline{Y}} > \overline{\overline{X}}$ then $\overline{\overline{Y}} = \overline{\overline{X}}$ if $\underline{X} < \underline{Y}$ then $\underline{X} = \underline{Y}$
=	$X = Y$	$\underline{X} < \overline{Y}$ and $\underline{Y} < \overline{X}$	if $\overline{\overline{X}} > \overline{\overline{Y}}$ then $\overline{\overline{X}} = \overline{\overline{Y}}$ else $\overline{\overline{Y}} = \overline{\overline{X}}$ if $\underline{X} < \underline{Y}$ then $\underline{X} = \underline{Y}$ else $\underline{Y} = \underline{X}$

A noteworthy property of propagation over relation links is that modifications to interval bounds always result in an interval of decreased width. This means that the quantitative precision of a variable can never decrease. This property has significant impact on the convergence of numeric consistency algorithms, because intervals shrink monotonically as the algorithm progresses.

4.6 Soundness of Interval Methods

By using established techniques for dealing with intervals, numeric consistency techniques are able to provide the important guarantee that the correct result will be bounded by the domains of each variable. This is a guarantee, which other methods such as Monte Carlo simulation and hill-climbing techniques cannot provide.

One of the strengths of numeric constraint satisfaction algorithms is they are able to solve interval equations even when a closed-form formula cannot be obtained. A number of numerical simulation techniques may also be applied to solve these types of equations, including Monte Carlo simulation. In Monte Carlo simulation, complex equations are solved by picking at random a value for each variable, which lies within the range of acceptable values for the particular variable. Each term in the equation is evaluated on these particular values, giving one possible value for each term. By iterating random choices, a range of possible values of the terms is found. In this way complex terms can be evaluated even though the value of the variables are not known with certainty. Another method, which may be used to evaluate such expressions, is to use hill-climbing techniques, which seek the maximum and minimum values of each quantity. Neither Monte Carlo simulation nor hill climbing is a sound inferential technique [1]. They generally return a subset of the true range, and thus arbitrarily exclude legitimate possibilities. Experience with hill-climbers has found them to be slow and unreliable [9].

The assurance that the correct result is bounded by the domains of variables comes from a basic property of interval arithmetic [10]. Let $y = f(x)$ define a function where x is within an interval X, and where $F(X)$ is also an interval which has as a lower bound the minimum value of any $f(x)$, and upper bound the maximum value of any $f(x)$. This property is stated as:

$$\forall\, x \in X,\ f(x) \in F(X).$$

This result is significant in the context of the soundness of interval analysis techniques. While any other floating-point calculation provides simply an estimate of the correct result of a computation, interval methods guarantee bounds for the correct result.

5 Constraint Inference

A number of additional techniques, both qualitative and quantitative, are used to QES to complement constraint propagation. Most of these techniques fall under the category of constraint inference methods, which have been used in systems such as Quantity Lattice [11]. Constraint inference is a way of deriving new constraints from existing ones. Three types of constraint inference used in QES are relational arithmetic, constant elimination arithmetic, and graph search.

5.1 Relational Arithmetic

Interval arithmetic sometimes leads to results, which are weaker than would be expected. One such limitation of interval arithmetic is the *selection problem*. Given the relation $X > Y$, and the assignments $X = (0, 1]$, $Y = [0, 1)$, we should be able to infer that $(X - Y) = (0, 1]$, since $X > Y$. In general, interval arithmetic does not even allow one to determine that $X - X = 0$. Using interval arithmetic, if $X = [1, 2]$ then $X - X = [-1, 1]$. Only by knowing that both intervals refer to the same quantity can we infer that the result is $[0, 0]$. Another limitation of interval arithmetic is that sometimes it cannot increase our knowledge at all. If we are given two quantities X and Y, and all we know is that $X = Y + 5$ and $Y = [-\infty, +\infty]$, interval arithmetic leads to the result $X = [-\infty, +\infty]$. Although we know that $X > Y$, interval arithmetic is not able to capture this fact. An arithmetic technique called *relational arithmetic* may be used to compensate for both these deficiencies. Relational arithmetic maintains constraints on the qualitative relationship of an arithmetic expression to its arguments. The axioms of relational arithmetic are given in Table 6.

5.2 Constant Elimination Arithmetic

Another type of arithmetic which may be used to reason about relationships between two arithmetic expressions is *constant elimination arithmetic* [11]. This type of arithmetic provides inference rules for a simple form of algebraic simplification. Constant elimination axioms allow us to infer that if $A = B + X$, $C = D + X$, and $B > D$, then $A > C$. Axioms for constant elimination are shown in Table 7.

Table 6: Relational arithmetic axioms.

+	$X \circ 0$	$\Rightarrow$	$(X + Y) \circ Y$
	$Y \circ 0$	$\Rightarrow$	$(X + Y) \circ X$
-	$X \circ Y$	$\Rightarrow$	$(X - Y) \circ 0$
×	$X > 0 \;\&\; Y > 0$	$\Rightarrow$	$(X \circ 1 \Rightarrow (X \times Y) \circ Y)$ & $(Y \circ 1 \Rightarrow (X \times Y) \circ X)$
	$X > 0 \;\&\; Y < 0$	$\Rightarrow$	$(X \circ 1 \Rightarrow Y \circ (X \times Y)$ & $(Y \circ -1 \Rightarrow (X \times Y) \circ -X)$
	$X < 0 \;\&\; Y > 0$	$\Rightarrow$	$(X \circ -1 \Rightarrow (X \times Y) \circ -Y)$ & $(Y \circ 1 \Rightarrow X \circ (X \times Y))$
	$X < 0 \;\&\; Y < 0$	$\Rightarrow$	$(X \circ -1 \Rightarrow -Y \circ (X \times Y))$ & $(Y \circ -1 \Rightarrow -X \circ (X \times Y))$
/	$X > 0 \;\&\; Y > 0$	$\Rightarrow$	$X \circ Y \Rightarrow (X / Y) \circ 1$
	$X > 0 \;\&\; Y < 0$	$\Rightarrow$	$X \circ -Y \Rightarrow -1 \circ (X / Y)$
	$X < 0 \;\&\; Y > 0$	$\Rightarrow$	$X \circ -Y \Rightarrow (X / Y) \circ -1$
	$X < 0 \;\&\; Y < 0$	$\Rightarrow$	$X \circ Y \Rightarrow 1 \circ (X / Y)$
For $\circ \in \{ <, \leq, >, \geq, =, \neq \}$			

Table 7: Constant elimination axioms.

+	$X \circ Y$	$\Rightarrow$	$(X + Z) \circ (Y + Z)$
-	$X \circ Y$	$\Rightarrow$	$(X - Z) \circ (Y - Z)$
	$X \circ Y$	$\Rightarrow$	$(Z - Y) \circ (Z - X)$
×	$X \geq 0 \;\&\; X \circ Y$	$\Rightarrow$	$(X \times Z) \circ (Y \times Z)$
	$X \leq 0 \;\&\; X \circ Y$	$\Rightarrow$	$(Y \times Z) \circ (X \times Z)$
/	$X \geq 0 \;\&\; X \circ Y$	$\Rightarrow$	$(X / Z) \circ (Y / Z)$
	$X \leq 0 \;\&\; X \circ Y$	$\Rightarrow$	$(Y / Z) \circ (X / Z)$
For $\circ \in \{ <, \leq, >, \geq, =, \neq \}$			

5.3 Graph Search

In order to invoke the axioms discussed above, information about ordinal relations between expressions is required. In some cases it is possible to infer new ordinal relations from a given set of ordinal relations on expressions using a technique called graph search. Graph search is conveniently implemented in a constraint network. Given the relations $A \leq D$ and $D = E$, it is possible to deduce that $A \leq E$. The transitivity table (Table 8) enumerates the relationships between two inequalities which share a common expression. Entering the transitivity table at the intersection of the row labeled ≤ and the column labeled =, we find the relation ≤. In a constraint network, a simple *breadth-first search* [i.e. 5] may be used to find a path between two expressions. In the constraint graph, we wish to find the relationship between the expressions A and C, even though they are not directly connected by a relation link.

The search starts at the expression A and proceeds until the expression C is reached. As each expression is visited, the relationship of that particular expression with the variable A is recorded. In the figure, the notation [*n*: *rel*] is used to indicate that the relation *rel* is written at

the expression node at the n^{th} step of the algorithm. For example, since the sign '≤' was written at node *D* in step 1, this information along with the relation '=' on the arc linking nodes *D* and *E*, is used with the transitivity table to determine that '≤' must be entered at expression *E*.

Table 8: Transitivity table for ordinal relations.

	<	≤	>	≥	=	≠
<	<	<	?	?	<	?
≤	<	≤	?	?	≤	?
>	?	?	>	>	>	?
≥	?	?	>	≥	≥	?
=	<	≤	>	≥	=	≠
≠	?	?	?	?	≠	?

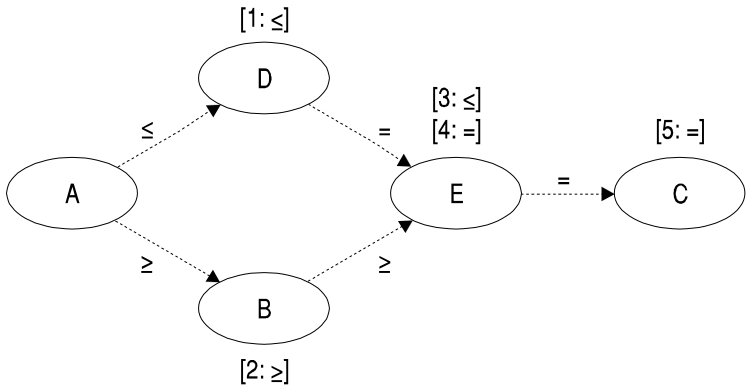

Figure 20: Constraint network for graph search example.

In some cases, more than one path may exist between two expressions in a constraint network. If this occurs, the results of each path may be combined to produce a 'tighter' result. In Figure20, two paths lead from expression *A* to expression *C*. On the third step of the algorithm, the sign '≤' is written at the node *E*. Traversing the path *A-B-E* we find that $A \geq E$. Combining the relations '≤' and '≥' we may deduce that $A = E$. If the graph search is able to determine a constraining relation between two expressions, the relation may be cached by adding a new relation link between the two expressions. A subsequent query to the constraint network can use this information to quickly determine the relationship between two expressions.

Constraint inference furnishes techniques, which complement the constraint-satisfaction methods discussed earlier. Systems, which rely only on constraint influence, have a number of limitations however. One of the problems with constraint inference is that it is difficult to control and it often falls into infinite loops. The constraint propagation techniques discussed in connection with constraint satisfaction algorithms are easier to control than constraint inference. In addition, as new constraints are inferred, it may be difficult to ensure that they will be useful in answering queries to the network.

6 Conclusions

This paper has described a computer software application, QES, which may be used to evaluate conceptual and preliminary engineering designs. The application employs a number of qualitative and semi-quantitative techniques to handle physical systems, which are characterized by a high level of uncertainty. QES exploits the constraint network paradigm to perform both qualitative and quantitative reasoning within an integrated framework. Qualitative reasoning, quantitative interval arithmetic, arithmetic reasoning and graph search are complementary techniques which are used to extract a considerable amount of information from uncertain or

incompletely specified input data. These techniques show considerable advantages over other, more commonly used approaches for handling uncertainty in engineering practice.

An important feature of qualitative and semi-quantitative methods is that they allow the engineer the reason with a high level of abstraction. Qualitative equations are capable of representing a class of physical systems rather than one specific system. In the structural example, the corresponding quantitative equations would represent a very specific system, with one set of member lengths and section properties. On the other hand, the qualitative equations represent a large number of physical structures, with a range of materials, section properties, and member geometry.

Qualitative and semi-quantitative methods are useful for evaluating conceptual and preliminary designs, because by reasoning with an abstract model, the engineer may perform a sound preliminary evaluation of a physical system without committing to details. All system parameters can be represented using qualitative variables or interval variables that are capable of covering a wide range of possible system configurations. The overall integrity of the proposed system can be confirmed at an early stage in the design, so that sound conceptual design alternatives are retained while unsound alternatives are eliminated.

A wide range of tools is available which are able to perform detailed numerical analysis. These tools generally require input that is specific and complete. The user must commit to a large number of detailed assumptions about the physical system. At the earliest stages of design, the validity of these detailed assumptions is questionable. A considerable amount of time may be required to generate the assumptions and create the detailed model of the system.

Qualitative and semi-quantitative methods have distinct advantages over other approaches for dealing with uncertainty in engineering systems, such as probability methods, Monte Carlo simulation, hill-climbing techniques and expert systems. Numerical simulation techniques based on probability, such as Monte Carlo methods, and hill-climbing techniques, which seek the maximum and minimum values of each quantity, are not sound inferential techniques. and thus may exclude legitimate solutions. They generally return a subset of the true range, and thus arbitrarily exclude legitimate possibilities. The techniques discussed in this paper are logically sound. The interval analysis methods employed are able to guarantee bounds on the correct solution.

Expert systems represent another approach to coping with uncertainty in engineering practice. These systems have been applied successfully in engineering domains, partly because they are able to reason with valuable experiential and heuristic knowledge. Most expert systems suffer, however, because of an inability to reason using fundamental domain knowledge. On the other hand, qualitative reasoning uses a detailed domain model, an explicit representation of the first principles of the domain that underlie heuristic knowledge.

Purely qualitative reasoning methods developed in artificial intelligence research were applied to a practical problem in the domain of structural engineering. It was found that the conclusions that may be drawn from such methods could be quite powerful when the number of variables in the problem is limited. Unfortunately the conclusions that may be drawn become rapidly weaker as the number of unknowns in the problem is increased. Although a limited number of problems were tested, the findings are not out of line with previous research. A number of attempts have been made by artificial intelligence researchers to resolve the ambiguity of predictions made using qualitative analysis. The approach used in this research was to augment qualitative analysis with partial quantitative information. Semi-quantitative techniques show considerable promise in the practice of engineering for their ability to model design abstraction while providing bounds for the behaviour of physical systems.

6.1 Further Research

In order to fully evaluate the techniques presented in this paper, the techniques must be used in real engineering applications. The goal of developing the QES software was to provide an accessible tool, which could be used, for the analysis of conceptual and preliminary engineering designs. Further research is required to ascertain which types of reasoning, qualitative or quantitative, are best suited various types of engineering problems.

7 References

1. Davis, E. (1987) Constraint propagation with interval labels. *Artificial Intelligence* **32**:281-331.
2. Forbus, K.D. (1984) Qualitative process theory. *Artificial Intelligence* **24**:85-168.
3. Freuder, E.C. (1978) Synthesizing constraint expressions. *Communications of the ACM*, **21**:958-966.
4. Gedig, M.H. (1995) A Framework for Qualitative and Semi-Quantitative Analysis in Engineering Design and Evaluation. Masters Thesis, University of British Columbia.
5. Knuth, D.E. (1968) *The Art of Computer Programming.* Addison-Wesley, Reading, MA.
6. Kuipers, B.J. (1994) Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge. MIT Press, Cambridge, MA.
7. Lhomme, O. (1993) Consistency techniques for numeric CSPs. In *Proc. Thirteenth Joint Conference on Artificial Intelligence, IJCAII 1993*, Morgan Kaufmann, San Mateo, CA. 232-238.
8. Mackworth, A.K. (1977) Consistency in networks of relations. *Artificial Intelligence* **8**:99-118.
9. McDermott, D.V. and Davis, E. (1984) Planning routes through uncertain territory. *Artificial Intelligence* **22**:107-156.
10. Moore, R.E. (1966) *Interval Analysis*. Prentice-Hall, Inc., Englewood Cliffs, NJ.
11. Simmons, R. (1986) Commonsense arithmetic reasoning. *Proc. Fifth National Conference on Artificial Intelligence* 118-124.

