Defence Science Journal, Vol 43, No 2, April 1993, pp 111-119

An Object-oriented Knowledge-based System for Basic Problem-solving in Science and Engineering

M.V. Krishnamurthy* and F.J. Smith

Department of Computer Science, The Queens University of Belfast

Belfast BT7 INN, Northern Ireland, United Kingdom

ABSTRACT

The traditional database systems and knowledge-based systems are inadequate to handle the knowledge required to solve computational problems in science and engineering. The conventional handbooks and textbooks of science and engineering contain data on scientific quantities in the form of tables and a large number of formulae relating these quantities in the form of algebraic equations. This paper describes a knowledge-based system, developed using object-oriented approach, to store and manipulate this kind of knowledge. The formulae are input in the same form as they are written, in terms of the well known symbols used by an engineer or a scientist. The system interprets the symbols as representing scientific quantities and links them with the underlying data and methods in the knowledge base. By linking data on scientific quantities with one or more appropriate formulae, it is shown that the knowledge base can be used for basic problem solving in science and engineering. The system has been developed on a Sun Sparc Station using object-oriented enviornment, objectworks C++

1. INTRODUCTION

Most of the present-day database systems are designed primarily to store and retrieve data in the form of tables, whereas the conventional knowledge base systems are intended to handle knowledge in the form of rules. It has been argued frequently that these systems are not adequate to store and manipulate the wide range of data and knowledge encountered in science and engineering[1,2]. For example, the conventional handbooks and textbooks of science and engineering contain numerous algebraic formulae representing scientific laws describing relationships among scientific quantities, besides data on these quantities[3]. As a frequent requirement of many scientists and engineers is to solve problems which involve the application of formulae to data, we believe that storage of formulae is as important as storage of data. We describe in this paper a knowledge-based system developed by us at the Queens's University of Belfast to store and manipulate knowledge in the form of data and scientific laws represented by formulae. The system has been developed as a part of a large cooperative project involving cooperation with the Belfast Aircraft Manufactures, Shorts, part of the Bombardire group.

We first describe the nature of the basic knowledge required for solving problems in science and engineering. We give an overview of the system and later describe the two important components: Knowledge base and Problem solver. We also illustrate the problem solving strategy with examples involving mechanical properties of materials.

2. RELATED WORK

There have been some attempts in the past to build systems which can store formulae. Smith and Hughes[4] developed a system called 'Belfast Scientific Database System' suitable for storing a wide range of data on science and technology. A special feature of this system was the storage of formulae as functions and subroutines. Another parallel system was developed by

Received 20 September 1992

*Presently on leave of absence from Defence Research and Development Laboratory (DRDL), Hyderabad

the same team to process more general queries using the relational data model and the Query-by-Example interface[5]. Bandyopadhyay *et al*[6] developed a system called "Symbolic Information Management System (SIS)" for storage and manipulation of structured data and algebraic formulae. They used a model called the "Symbolic Relational Model" which was a derivative of the relational model. In SIS, formulae were represented as special attributes of a symbolic relation. All these systems facilitate storage of both data and formulae.

Early intelligent programs to solve physics problems were MECHO by Bundy[7] and ISSAC by Novak[8]. MECHO was developed to solve problems in mechanics. In MECHO, formulae were represented using predicate logic. In ISSAC, formulae were represented as procedures attached to a canonical object frame. Though these systems were powerful in exploiting the domain knowledge and facilitated specification of the problem in natural language, their usage was restricted to highly specific domains. Recently, Tyugu *et al*[9, 10] developed a system called PRIZ for general problem solving. Tyugu suggested the use of 'Computational models' which are special kinds of semantic networks for representing quantities in mathematics and physics and formulae relating these quantities. However, these systems do not provide any means to store data.

Our project is an attempt to develop the above ideas further, to represent data and formulae together in an object-oriented framework aimed at problem solving. Besides data and formulae, we also intend to store knowledge on geometry and units. We believe that the object-oriented methodology makes possible a uniform paradigm to represent such heterogeneous data and knowledge, which was lacking in our earlier work[4, 5]. Besides supporting complex data modelling, the object-oriented paradigm also offers abstraction, encapsulation and the capability to capture semantics through inheritance.

3. NATURE OF THE KNOWLEDGE

The two most basic entities involved in problem solving are quantities and formulae.

3. Quantities

The notion of quantity is central to problem solving in science and engineering. A quantity can be a geometrical quantity like area, volume, etc or a physical quantity like mass, force, etc. The value of a geometrical quantity depends on the geometrical shape under consideration and its dimensions. Physical quantities can be categorized into constants, variables and properties. Physical constants are the universal constants of nature, such as the velocity of light *(c)*, $2.9979 \times 10^8 ms^{-1}$, or the gas constant *(R)*, 8.314 J $mole^{-1}$ $Kelvin^{-1}$. Physical variables (sometimes called state variables) are independent variables which describe the state of a physical system, such as temperature *(T)* or pressure *(P)*. Physical properties are quantities which hold different values for different materials (or elements) in different states, for example, elastic modulus *(E)*, which has a vlue of 0.70×10^{11} Pascal for aluminium and 0.91×10^{11} Pascal for brass at room temperature. The physical constants and physical properties are held in the knowledge base and the variables (including geometric values) are specified by a user during problem solving.

Each quantity can be described (as above) by the attributes such as name, symbol, units and data values. The data values, which are often fuzzy[11], are normally expressed in the form of tables. Since there are different systems of units like SI (International System), FPS (foot-pound-second), CGS (centimetre-gram-second), etc., which different users might wish to use, conversion of a vlue from one system of units to another is vital. The dimensions of a quantity are normally determined from the units and are expressed as symbolic expressions in terms of dimensions of fundamental quantities like mass *(M)*, length *(L)*, time *(T)* and temperature *(K)*. Examples of dimensions are force *(F)*: MLT^{-2} or viscosity *(N)*: $M^{-1}LT^{-1}$. These can be used to check the correctness of the physical laws or formulae.

Quantities are normally identified by symbols, well known to every practising engineer or scientist. In certain cases, a single symbol is used to represent two or more different quantities; for example, the symbol E is used to represent energy, electric field or elastic modulus. Normally, an engineer interprets the symbol appropriately based on the context in which it is used. Our system resolves these ambiguities using simple dimensional arguments.

3.2 Formulae

Formulae describe the relationships among quantities and are expressed in the form of mathematical equations. A formula can be a simple definition of a

physical quantity or a physical law describing a phenomenon or fact. For example, the quantity tensile stress is defined by the formula

$$\sigma = F/A \quad (1)$$

where F is the load applied and A is the cross-sectional area. An example of a physical law is

$$K/\sigma = LT \quad (2)$$

which states that the ratio of thermal conductivity K to electrical conductivity σ is proportional to the absolute temperature T, where L is a constant known as the Lorentz constant.

Physical laws are often valid only under certain conditions, which are expressed as constraints on the law. Two examples illustrate this. In example 1, if E is the elastic modulus (or Young's modulus), K is the bulk modulus, G is the tortion modulus, and σ is the Poisson's ratio, then K and G are related to E and σ through the following laws:

$$K = E/3\,(1 - 2\sigma) \quad (3)$$

$$G = E/2\,(1 + \sigma) \quad (4)$$

However, these relations are valid only for isotropic solid materials. Therefore, before using them, the system would need to know that they apply to some solid material and that the material is isotropic.

In the second example, the resistivity of a metal over a temperature range that is close to T_o is expressed as

$$\rho = \rho_o\,[1 + \alpha\,(T - T_o)] \quad (5)$$

where ρ_o is the resistivity as temperature T_o and α is the resistivity temperature coefficient.

Formulae are used mainly for solving problems by computing the value of a quantity such as the change in length of a steel wire under an applied force. This would be computed from the law

$$\delta L = FL/AE \quad (6)$$

where (δL) is the change in length, E is the elastic modulus, L is the length, A is the cross-section, and F is the force applied.

4. COMPONENTS OF THE SYSTEM

The system consists of three main components (Fig. 1): knowledge base, problem solver and user interface.

The knowledge base consists of the essential knowledge required for problem solving, that is, the physical quantities including data values, formulae, units of measurement and geometry. The problem solver obtains the specifications of the problem to be solved from the user interface and interacts with the knowledge-base for retrieving the value of a quantity, or retrieving a formula or converting a quantity value from one system of units to another. The search engine in the problem solver is responsible for the selection of appropriate formulae for solving a problem. The formula interpreter interprets a formula to compute the value of a quantity. The user interface consists of two components: graphic front-end, and formula translator. The graphic front-end creates menus, pop up windows, etc. required for user interaction. The formula translator translates the formula in symbolic form at input to an unambiguous form. This is required because, as mentioned earlier, there are symbols which correspond to more than one quantity. For example, symbolic form and the actual form of Eqn.(2) are given below.

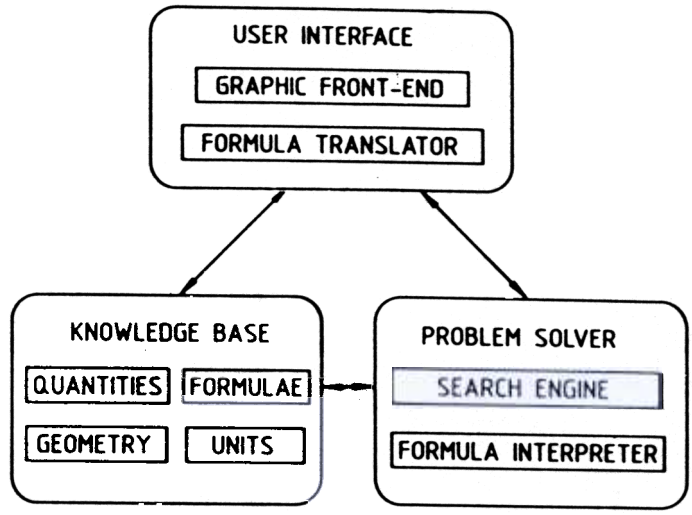

Figure 1. Components of the system.

Symbolic form:	$K = E/3(1 - 2\sigma)$
Actual form:	Bulk modulus = elastic modulus/3 (1 − 2 Poisson's ratio).
Constraint:	Isotropic solid materials

The two important components of the system knowledge base and problem solver are discussed below.

5. KNOWLEDGE BASE

A conceptual model of the knowledge base showing the interrelationship between quantities and formulae is given in Fig. 2. For the sake of completeness, we have

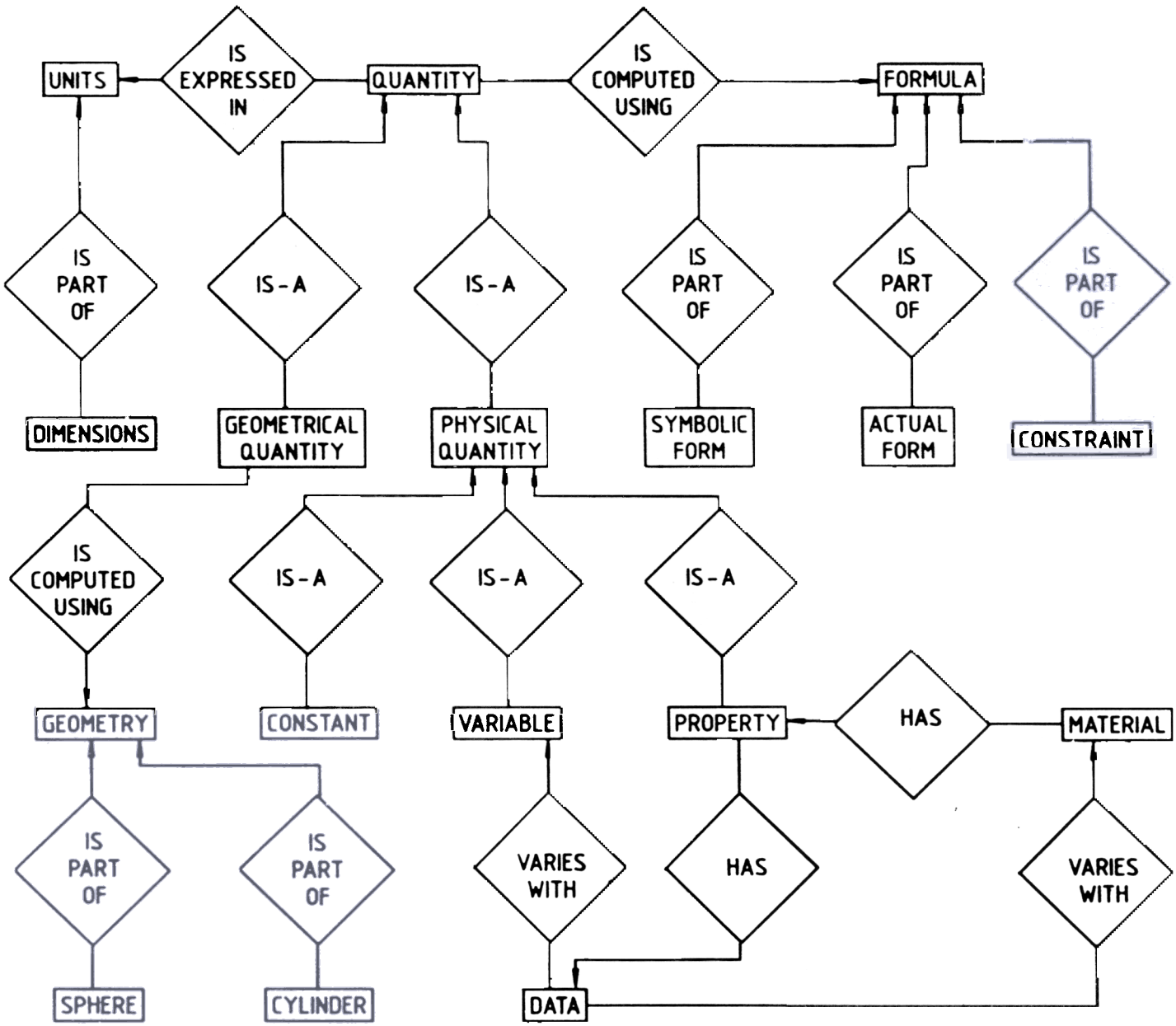

Figure 2. Conceptual model of the knowledge base

also included the entities units, geometry and materials. We believe that complete information on these entities is essential in a full-fledged scientific database or knowledge base system with intended capabilities of problem solving. As it is not often possible to store the data values in all systems of units, knowledge of units is essential for converting values among different systems of units. The entity geometry contains knowledge of different geometrical shapes (only sphere and cylinder are shown in the figure). This information will be useful in computing the values of certain geometrical quantities during the process of solving a problem. The entity material contains the descriptions of materials (or substances or elements). This entity can be further classified or elaborated based on the application. We do not claim that this model is absolutely complete for solving all problems in science and engineering. But we believe that it can be easily extended to include data and knowledge required to solve more general problems.

The conceptual model has been mapped onto an object-oriented framework which has been built by us[12,13]. Each entity in the diagram is implemented as a class in C++[14], and for the sake of illustration, the

```
CLASS QUANTITY
{

        SET(STRING)          NAMES;
        SET(STRING)          SYMBOLS;
        SET(UNIT)            UNITS;
        SET(FORMULA)         FORMULAE;

        PUBLIC:
           DOUBLE CONVERT-VALUE (UNIT SOURCE-UNIT, UNIT TARGET-UNIT, DOUBLE VALUE);
                              //CONVERTS VALUE FROM ONE SYSTEM OF UNIT TO ANOTHER
           SET (FORMULA) GET FORMULAE();
                              //RETRIEVES THE FORMULAE RELATED TO A QUANTITY
                              .....
        };
```

Figure 3. Definition of class quantity.

```
CLASS FORMULA
{

    STRING              SYMBOLIC-FORM;
    LIST                ACTUAL-FORM;
    STRING              CONSTRAINT;
    SET(QUANTITY)       QUANTITIES-RELATED

    PUBLIC:
        LIST            REWRITE (STRING QUANTITY-NAME);
                        //REWRITES THE FORMULA FOR A SPECIFIC QUANTITY
    DOUBLE              INTERPRET(STRING QUANTITY-NAME,ASSOC-LIST(DOUBLE) PARAMETERS);
                        //INTERPRETS THE FORMULA FOR COMPUTATION OF A SPECIFIC QUANTITY
                        .....
    };
```

Figure 4. Definition of class formula.

definition of the classes quantity and formula are given in Fig. 3 and 4 respectively, in their C++ syntax. Each class contains a set of attributes (for example, names, symbols, units, data, and formulae in the case of class quantity) and a set of methods specified under the label public. For the sake of simplicity only important attributes and methods are given. Each instance of a class represents a real world object. For example, mass is an instance of the class physical quantity and the unit Newton is an instance of the class units. A class can inherit data and methods from a more general class through inheritance. For example, the classes geometrical quantity and physical quantity can inherit data methods from the class quantity. Each object of a class is uniquely indentified by a system generated identifier which is transparent to the user.

Procedures for computing a value of a quantity from a formula or procedures for converting a value from one system of units to another are expressed as methods in our object-oriented design. This approach has also been useful in representing knowledge of geometry, because the geometrical shapes can be represented as objects (for example, sphere, cylinder, etc) with their properties (for example, radius) as attributes and formulae for computing the geometrical quantities as methods. It would have been more difficult to have represented this knowledge in any other than an object-oriented paradigm.

6. PROBLEM SOLVER

The main purpose of the problem solver is to compute the value of a quantity by applying formulae. This is simple when there is a single formula to apply and the values of all other quantities in the formula are known, either specified in the problem or available in the knowledge base. The process becomes complex when there are more than one formulae to apply and when the values of some of the quantities in the formulae are unknown (neither specified in the problem nor available in the knowledge base), but have to be computed from other formulae.

We used the problem decomposition strategy in artificial intelligence[15,16] to tackle this problem. We divide the problem of computation of a quantity into a number of sub-problems, each involving computation of a sub-quantity in the formula. When there are more than one formulae to apply, we try them one by one. The entire problem space can then be represented as an AND/OR tree and we can employ the depth-first recursive search to traverse the tree. The leaf-nodes represent quantities whose values are known. The search terminates at this level and returns the value to its previous level. When a dead-end is reached, the system backtracks to a previous level and selects another formula, if available. If the complete search space is exhausted, the system reports that the problem is unsolvable and prompts for more information. This search strategy is illustrated later with some examples.

7. AN APPLICATION

We illustrate the advantages of storing data and formulae together with a practical application. We tested our prototype system with some sample data and formulae related to properties of materials. Table 1 shows a portion of a data table with data on different elastic moduli stored in our system. Table 2 shows a sample subset of formulae stored in the knowledge base. The problem solving strategy is illustrated in Section 7.2 with some examples.

Table 1. Sample data of moduli of elasticity of some materials (as given in a handbook)

Material	Young's modulus (*Y*)		Shear modulus (*G*)		Bulk modulus (*K*)		Poisson's ratio (*v*)
	Pa 10^{11}	lb $in^{-2}10^6$	Pa 10^{11}	lb $in^{-2}10^6$	Pa 10^{11}	lb $in^{-2}10^6$	
Aluminium	0.70	10	0.30	3.4	0.70	10	0.16
Brass	0.91	13	0.36	5.1	0.61	8.5	0.26
Copper	1.1	16	0.42	6.0	1.4	20	0.32
Glass	0.55	7.8	0.23	3.33	0.37	5.2	0.19
Iron	1.9	26	0.70	10	1.0	14	0.27
Lead	0.16	2.3	0.056	0.8	0.077	1.1	0.43
Nickel	2.1	30	0.77	11	2.6	34	0.36
Steel	2.0	29	0.84	12	1.6	23	0.19
Tungsten	3.6	51	1.5	21	2.0	29	0.20

Table 2. Sample formulae relating elastic properties of materials

Symbolic form	Actual form	Constraint
$\sigma = F/A$	Tensile stress = Load/cross-sectional area	
$\epsilon = \Delta L/L_o$	Tensile strain = Elongation/original length	
$\Delta L = L_i - L_o$	Elongation = Instantaneous length - original length	
$T = F/A$	Shear stress = Load/cross - sectional area	
$\gamma = \text{Tan}\,\theta$	shear strain = Tan (Shear angle)	
$\sigma = E\epsilon$	Tensile stress = Elastic modulus* tensile strain	Perfect deformation
$T = G\gamma$	Shear stress = Shear modulus* shear strain	Perfect deformation
$B = E/3(1-2\nu)$	Bulk modulus = Elastic modulus / [3*(1 − 2 *Poisson's ratio)]	Isotropic materials
$G = E/2(1+\nu)$	Shear modulus = Elastic modulus / [2* (1 + Poisson's ratio)]	Isotropic materials

7.1 Reduction of Storage

Besides the automatic solution of problems which we discuss next, the system reduces storage requirements by eliminating the need to store data in different systems of units and to store data on interdependent quantities. For example, the actual datastored in the system is shown in Table 3. We eliminated the storage of values in FPS units (which can be computed using the knowledge of units) and the storage of values of bulk modulus and shear modulus (which can be computed using formulae 3 and 4). The actual reduction in storage is considerable, since the data for each material may vary with two or more independent variables like temperature, pressure, etc. The computational overhead arising out of this reduction in space is minimal, as the formulae contain mostly simple arithmetic operators.

Table 3. Reduced table (as stored in the system)

Material	Young's modulus *(E)* Pa 10^{11}	Poisson's ratio (v)
Aluminium	0.70	0.16
Brass	0.91	0.26
Copper		0.32
Glass	0.55	0.19
Iron	1.9	0.27
Lead	0.16	0.43
Nickel	2.1.	0.36
Steel	2.0	0.19

7.2 Problem Solving

The problem solving strategy is illustrated below with two examples.

Problem 1: A piece of copper orginally 12 in. long is pulled down in tension with a stress of 40,000 psi. If the deformation is entirely elastic, what will be the resultant elongation?

This problem can be specified as follows

$L_o = 12$ in.
$\sigma = 40{,}000$ psi
Material = copper
geometry = bar
$\Delta L = ?$

A solution tree for the above problem which involves the use of three formulae from Table 3 is shown in Fig. 5.

Problem 2: A tensile strees is to be applied along the long axis of a cylindrical rod that has a diameter of 0.4 in. Determine the magnitude of the load required to produce a 10^{-4} in change in diameter if the deformation is entirely elastic.

To compute the load *(F)*, the formula $\sigma = F/A$ is selected. Since A (cross-sectional area of the cylinder) and σ (tensile stress) are not specified in the problem, they have to be computed before computing F; A is computed using the knowledge of geometry and σ is computed using other formulas and data available in the knowledge base. A solution tree similar to Fig. 5 can be drawn for this problem. The tree will be much larger, as it involves computation of several intermediate quantities.

8. CONCLUSION

The prototype of the system has been implemented on a Sun Sparc system using the environment object works C++ and has been tested with sample data and formulae on properties of materials. We have selected the C++ language because of its efficiency and portability[17,18]. As C++ does not support persistence, we had to build our own object manager to store and manipulate objects. Although this was aided by our earlier work on database design[4] and by the object-works environment[19], an object-oriented database system, such as objectstore[20] might have speeded up the

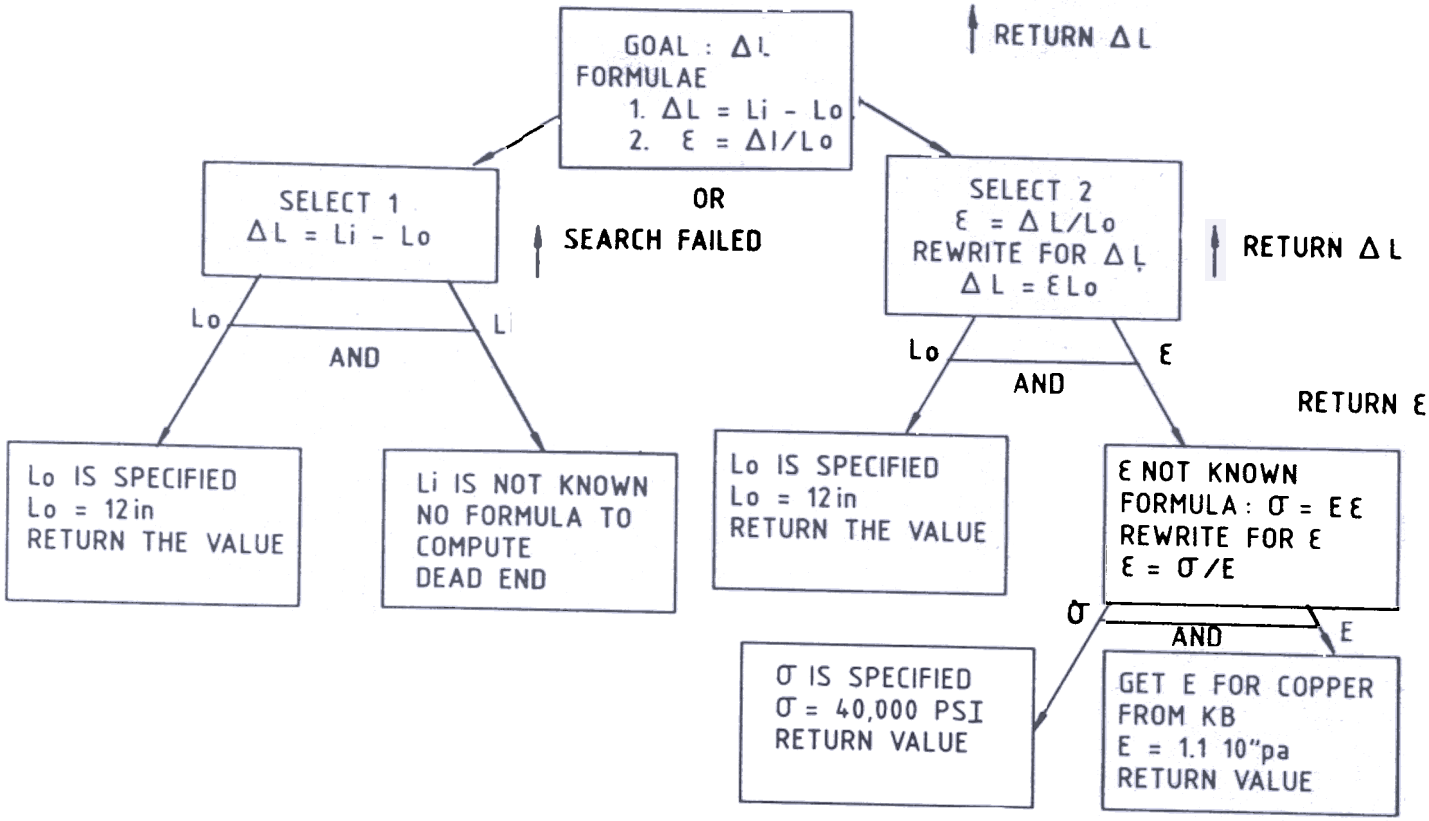

Figure 5. Solution tree for problem 1

work if it had been available when the project started. There appears to be no undue difficulty in building a system of this kind using object-oriented technology to store and manipulate simple formulae found in many text books of science and engineering. However, automatic manipulation of more complex formulae than those described in this paper, such as those involving differentials and integrals, will be much more difficult.

ACKNOWLEDGEMENTS

One of the authors (MVK) expresses his gratitude to the Ministry of Education, Government of India, for awarding Nehru Centenary British Fellowship administered by the Bristish Council to carry out this research. He also expresses his gratitude to Brig RK Bagga AVSM, Director, CIC, DRDL, for encouragement and DRDO authorities for granting leave to do this research.

REFERENCES

Rumble, J.R. & Smith, F.J. Database systems in science and engineering. Adam Hilger, Bristol, United Kingdom, 1990. p. 134.

2. Smith, F.J. & Hughes, J.G. Some special features of a materials database system. *In* Proceedings of the Conference on Data and Knowledge Systems for Manufacturing and Engineering, Hartfortd Ct, October 1987. pp. 76-82.
3. Hughes, J.G.; Smith, F.J. & Tripathy, S.R. A knowledge-base for the properties of materials. *In* Proceedings of the 11th International CODATA Conference edited by P.S. Glaeser. Hemishphere Press, 1990. pp. 122-26.
4. Smith, F.J. & Hughes, J.G. The Belfast scientific database system. In the role of data in scientific progress, edited by P.S. Glaeser. North Holland, Amsterdam, 1985. pp. 435-37.
5. Bandyopadhyay, S; Hughes, J. G.; Smith, F.J. & Sen, K. A Generalised scientific information system. *Comput. Phys. Cmmun.*, 1984, **33**, 49-53.
6. Bandyopadhyay, S. & Devitt, J.S. A symbolic information management system. *J. Symb. Comput.* 1987, **4**, 397-408.
7. Bundy, A. Solving mechanics problems using meta-level interface. *In* Expert systems in micro

electronic age, edited by D. Michie. Edinburgh University Press, Edinburgh, Scotland, 1979. pp. 50-64.

8. Novak, G.S. Representation of knowledge in a program for solving physics problems. *In* Proceedings of the 5th International Joint Conference on Artificial Intelligence, 1977. pp. 286-91.

9. Mints, G. & Tyugu, E. The programming system PRIZ. *J. Symb. Comput.*, 1988, **5**, 286-91.

10. Tyugu, E.N. Knowledge-based programming environments. *Knowledge Based Syst.*, 1991, **4** (1), 4-15.

Smith, F.J. & Emerson, L. Indexing technical data in a material database. *In* Proceedings of the Conference on Data and Knowledge Systems for Manufacturing and Engineering Gaithersburg, MD, October 1989. pp. 11-18.

12. Smith F.J. & Krishnamurthy, M.V. Integration of scientific data and formulae in an object-oriented system. *In* Proceedings of the 6th International Conference on Scientific and Statistical Database Management, Zurich, Switzerland, June 1992. pp. 110-22.

13. Krishnamurthy, M.V. & Smith, F.J. Representation of scientific and engineering knowledge for problem solving-An object oriented approach. *In* Proceedings of the 2nd Golden West International Conference on Intelligent Systems, Reno, June 1992, pp. 104-109.

14. Stroustrup, Bjarne. The C++ programming language. Prentice-Hall, London, UK, 1987.

15. Nilsson, J.N. Principles of artificial intelligence, Chapter 3. Springer International, Berlin, Germany, 1980.

16. Rich, E. Artificial intelligence, Chapter 3. McGraw-Hill, New York, 1983.

17. Marc, R. *et al.* Object-oriented programming in AI-new choices. *AI Expert*, 1989, 53-69.

18. Jordan, D. Implementation benefits of C++ language mechanisms. *CACM*, September 1990, 61-64.

19. AT&T objectwork C++ selected readings Release 2.1. USA, 1989.

20. Lamb, Charles *et al.* The objectstore database systems. *Communications of the ACM*, 1991, **34** (10), 50-63.