Artif Intell Rev (2008) 29:93–122
DOI 10.1007/s10462-009-9110-0

A review of methods for automatic understanding of natural language mathematical problems

Anirban Mukherjee · Utpal Garain

Published online: 7 August 2009

Abstract This article addresses the problem of understanding mathematics described in natural language. Research in this area dates back to early 1960s. Several systems have so far been proposed to involve machines to solve mathematical problems of various domains like algebra, geometry, physics, mechanics, etc. This correspondence provides a state of the art technical review of these systems and approaches proposed by different research groups. A unified architecture that has been used in most of these approaches is identified and differences among the systems are highlighted. Significant achievements of each method are pointed out. Major strengths and weaknesses of the approaches are also discussed. Finally, present efforts and future trends in this research area are presented.

Keywords Artificial intelligence · Natural language processing · Mathematical problems · Automated reasoning · Knowledge engineering

1 Introduction

Right from the early 1960s or even earlier a distinct subject of research in computing had been artificial intelligence (AI)-based systems that can understand, analyze, solve or answer questions, cases, facts or problems in a natural language environment. Researches are still on in this direction, the ultimate objective being to explore the capability of a computer to act as a replacement of human intelligence and reasoning ability backed with multitude of domain knowledge. However, in comparison to the total research effort in the above direction

A. Mukherjee (✉)
Department of IT, RCC Institute of Information Technology, Canal South Road, Kolkata 700015, India
e-mail: anirban.mukherjee@rcciit.in; anirbanm.rcciit@gmail.com

U. Garain
Computer Vision and Pattern Recognition Unit, Indian Statistical Institute,
203, B.T. Road, Kolkata 700108, India
e-mail: utpal@isical.ac.in

little work has been seen in automating mathematical problem solving in a natural language environment in a way that simulates human problem solving approach.

By the term ‘mathematical problem’ we largely mean problem that involves some kind of scientific theory/rule or logic or formulae and basically requires mathematics to solve it—be it a problem of mechanics, physics, geometry, or algebra. Developing intelligent systems for understanding and solving natural language mathematical problems is not easy. It focuses on a problem which is different from one that is handled while developing menu or command driven software to solve computation-oriented mathematical problems. These typical softwares have a control over the user-input through pre-designed structured input mechanism whereas the intelligent systems are supposed to accept natural language description of a problem that can be stated in many different ways by different users. A natural language based system should use AI to understand the problem out of all lexical complexity, interpret it in terms of known concepts and then apply the most appropriate solution logic. On the other hand, software packages usually follow a processing pattern which is mostly mechanical and hence, do not require any language comprehension module or domain knowledge to solve a problem. The question of misinterpreting a problem does not arise for such programs whereas the natural language problem solvers run the risk of giving erroneous results for misinterpretation or lack of knowledge for a particular problem or case.

Ideally, a natural language mathematical problem solver should use the information presented in the problem itself to figure out the mathematical relationships among the elements and use some intelligence to propose the solution—the challenge lies in avoiding the rather easier approach of finding potential answers stored for matching cases. One more difficulty in developing such intelligent systems is to map an optimum set of rules out of a wide variety of available rules or techniques to solve all types of problems pertaining to any scientific domain and thus make the system robust for a given domain. This paper presents a comprehensive review of different natural language mathematical problem solvers developed from early systems such as STUDENT (Bobrow 1964a, b) through to recently developed ROBUST (Bakman 2007). Systems that deal with natural language other than English are kept out of the scope of this study. The approaches taken by different solver systems are summarized and presented with example problems. Attempt has been made to evaluate relative merits and general applicability of the approaches. The initial attempt of this study is to identify some common basic architecture used in the systems and also to trace the useful paradigms that have evolved to guide research efforts in this field. Generalizing from existing small-scale experimental systems to domain-specific complete problem solvers may entail a new order of complexity and require the modification of some of the known approaches or invention and development of entirely different approaches. The entire study embodied in this paper may provide a concise version of previous research as well as help to set a future direction of research in developing natural language mathematical problem solvers.

2 Description of systems

In late 1950s, Newell et al. (1959) developed the earliest predecessor of the automatic problem solver. This program was expected to solve all problems that humans can solve but ultimately it could only resolve a small set of logical problems (Parsaye and Chignell 1988). Gradually, it proved to be impossible to make a ‘general problem solver’ as nobody could find a small set of problem solving techniques or even a single semantic representation that would suit different domains. Thereafter, researchers concentrated on restricting the knowledge domain of the problem solvers and this eventually resulted in systems that could solve restricted

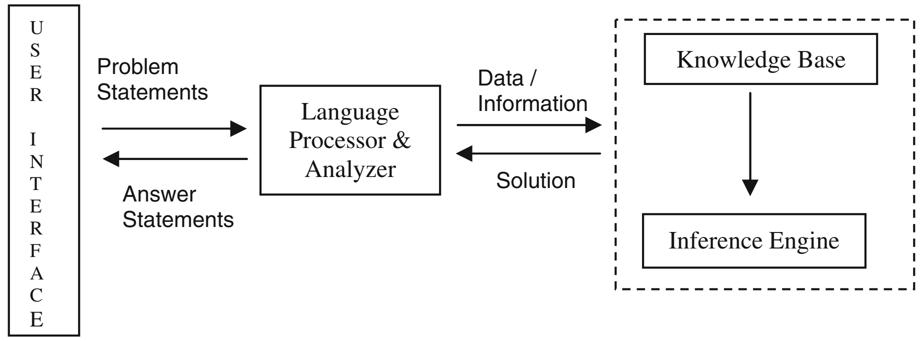

Fig. 1 Interaction of the logical parts of a natural language problem solver

sets of mathematical problems stated in natural language like English. Any natural language mathematical problem solver consists of four logical parts: the language processor and analyzer, the knowledge base, the inference engine and the user interface. The knowledge base is a combination of database and rule base that the inference engine uses to make inferences or solution. The language processor and analyzer extracts the data or information expressed through the text by syntactic and semantic processing of the problem statements. The user interface simply passes data or information back and forth between the user and the expert system.

Figure 1 is a simplified version of how most natural language problem solver works though all the systems covered here do not necessarily follow this model exactly; they do not all have inference engine or user interface. Figure 1 does not show some typical intermediate steps, for example, the formation of language independent internal representation of the input problem which is used by some systems before attempting to find the final solution or answer or output. The systems are studied here in two aspects—one is language comprehension and the other is the problem-solving scheme including much of what falls under the heading of 'knowledge engineering'.

2.1 Problems of mathematics

2.1.1 Algebra and arithmetic problem

The **STUDENT** program written by Bobrow (1964a, b) is the first program that reads, understands and solves an algebraic problem stated in a restricted set of English language and answers the questions pertinent to the problem in English language. For example a problem input to the program is—"`If the number of customer Tom gets is twice the square of 20% of the number of advertisement he runs, and the number of advertisements he runs is 45, what is the number of customer Tom gets?`" The output given by the program for this problem is—"`The number of customer Tom gets is 162.`"

The information storage structure used in STUDENT system follows a relational model. The objects in the model are variables, which are nothing but the words and phrases of the input sentences around the key words. For example "the number of customer Tom gets" is a variable where the key word is 'number'. The relationships between the objects are the mathematical relations of sum, difference, product, quotient, exponentiation and equality. The media for exhibiting such relationship are a set of equations expressed in a parenthesized prefix notation, e.g. `(EQUAL (NUMBER OF ADVERTISEMENTS HE RUNS) 45)`. Using recursive use of format matching complex sentences and coordinate sentences in the problem are resolved into simple sentences free of 'if', 'comma', 'and' etc. Stage-wise transformations are then carried out with necessary substitution (e.g. '2 times' for 'twice'), truncation (e.g. 'square of' to 'square') etc. until these sentences themselves become variables and are related with basic operators (like plus, times, percent etc.) to form a set of equations representing the problem. After such transformation the above problem becomes:

```
(EQUAL X00001(NUMBER OF CUSTOMER TOM GETS))
(EQUAL (NUMBER OF ADVERTISEMENTS HE RUNS) 45)
(EQUAL (NUMBER OF CUSTOMERS TOM GETS) (TIMES 2 (EXPT
(TIMES 0.2 (NUMBER OF ADVERTISEMENTS HE RUNS)) 2)))
```

Using standard techniques of solving simultaneous equations answers to the problems are finally found. Apart from some general information like "twice always means 2 times" or "12 inch equals 1 foot" no information is made permanent store of knowledge in the system.

Slagle's **DEDUCOM** (Deductive Communicator) (Slagle 1965; Simmons 1970) was a LISP program that stored LISP expressions of data statements such as: `'There are 5 fingers on a hand'`, `'There is one hand on an arm'`, `'There are 2 arms on a man'` and inference rules in the form of conditional statements that included variables as: `If there are` *m* `X's on a V and if there are` *n* `V's on a Y, then there are` *mn* `X's on a Y.` By substituting data statements for the variables in conditional expressions, DEDUCOM answers questions such as the following:

```
Q: How many fingers on a man?
A: 10.
```

Out of several different types of questions asked DEDUCOM correctly answered only 10 questions when fed with 68 facts but was found very slow in responding. Some of these questions required DEDUCOM to make logical deductions in predicate calculus. Slagle used some heuristic programming approach and depth-first search procedure for making such deductions and observed that the search procedures, in particular, had some inherent weakness that caused the system failing to respond correctly in many cases. However, it distinctly performed better when more relevant facts were given in the right order.

Another system called **WORDPRO** was developed by Fletcher (1985) for understanding and solving arithmetic word problems. WORDPRO uses a set of propositions to represent the meaning of a problem text. For example, `'Joe had 3 marbles'` is represented as: `(((P1 (EQUAL X JOE))) ((P2 (PAST P3)) ((P3 (HAVE X P4)) ((P4 (3 MARBLE)))`. The concept of set 'schema' (Marshall 1995) is applied as the basis of construction of problem model. Structurally, a 'change' schema holds relation amongst start-set (`'Joe had 3 marbles'`), transfer-set (`'Then Tom gave him 5 marbles'`) and result-set (`'How many marbles does Joe have now?'`). Refer Table 1 which lists the instantiations of four 'schemas' used in WORDPRO.

The program solves a problem guided by certain rules (13 'meaning postulates', 12 'arithmetic strategies' and 11 'problem solving procedures') which are applied sequentially for add/change/delete actions based on the content of the short term memory (STM) of the program; the rules add information to the problem model, add new schema to the STM, add request for schema with certain specification to complete a higher-level schemata, and eventually finds solution value for the problem. Two other programs, namely—**CHIPS** (Briars and Larkin 1984) and **ARITHPRO** (Dellarosa 1986) could also solve one-step arithmetic

Table 1 Schemas in WORDPRO

Schema	Instantiations
Change-in	`Joe had 3 marbles. Then Tom gave him 5 marbles. How many marbles does Joe have now?`
Change-out	`Fred had 3 boxes. Then he gave 2 boxes to Susan. How many boxes does Fred have now?`
Combine	`Lucy has 3 dimes. Sarah has 6 dimes. How many dimes do they have altogether?`
Compare	`Dan has 6 books. Jill has 2 books. How many books does Dan have more than Jill?`

word problems with only one possible operation—addition/subtraction. The models used by CHIPS and ARITHPRO categorize simple word problems into the same three categories: compare, combine and change. But for both these models rigid limitations were imposed on the change verb (only 'to give') and on the order of the problem sentences (the first sentence of the problem must describe the number of objects the owner had to begin with, whereas the second sentence must contain the verb 'gave').

Bakman (2007) developed a more advanced simulation program **ROBUST** that could understand free-format multi-step arithmetic word problems and that too with extraneous information (Muth 1992). An example having complex sense and also extraneous information is:

```
David gave 3 candies to Ruth, and John gave 2 candies to
David. Now David has 4 candies more than Ruth has. How
many candies does David have now, if Ruth had 7 candies
in the beginning?
```

The concept of 'schema' as used in the earlier works is adopted with introduction of 'change formulae' and further expansion of 'change' schemas into six distinct categories (namely, 'Transfer-In-Ownership', 'Transfer-Out-Ownership', 'Transfer-In-Place', 'Transfer-Out-Place', 'Creation' and 'Termination') as against only two used earlier (namely, 'Change-In' and 'Change-Out'). Such 'change formulae' exhaustively describes a change situation i.e. the initial number of items, number of items transferred and final number of items depending on the type of change (i.e. transfer of items to or out from a place or person).

The ROBUST simulation works by first parsing the problem text to split all sentences into propositions or simple sentences like `'David gave 3 candies to Ruth'`, `'David has ? candies'` etc. for the above problem. The propositions having complex change verbs like 'give' are further split into elementary ones (like `'David gave 3 candies to Ruth'` is splitted into `'David forfeited 3 candies'` and `'Ruth got 3 candies'`). Then the change formulae are applied to the elementary propositions by substituting the constant values for the corresponding variables. The formula propositions are matched to their instantiations in the problem text and the relevant schema is attached to the list of schema instantiations (LSI). The process of understanding results from the representation of the problem through the schema instantiations (listed in the LSI) of all compare, combine and change schemas.

Many computer aided instruction (CAI) systems are developed to help students learn how to solve algebra and arithmetic word problems. As for example, **Pump Algebra Tutor (PAT)** (Koedinger and Sueker 1996; Koedinger et al. 1997), **WORDMATH** (Looi and Tan 1998), **DISCOVER** (Steele and Steele 1999),**WPS Tutor** (Wheeler and Regian 1999) and **MathCAL** (Chang et al. 2006). All these systems have predefined strategy of instructing

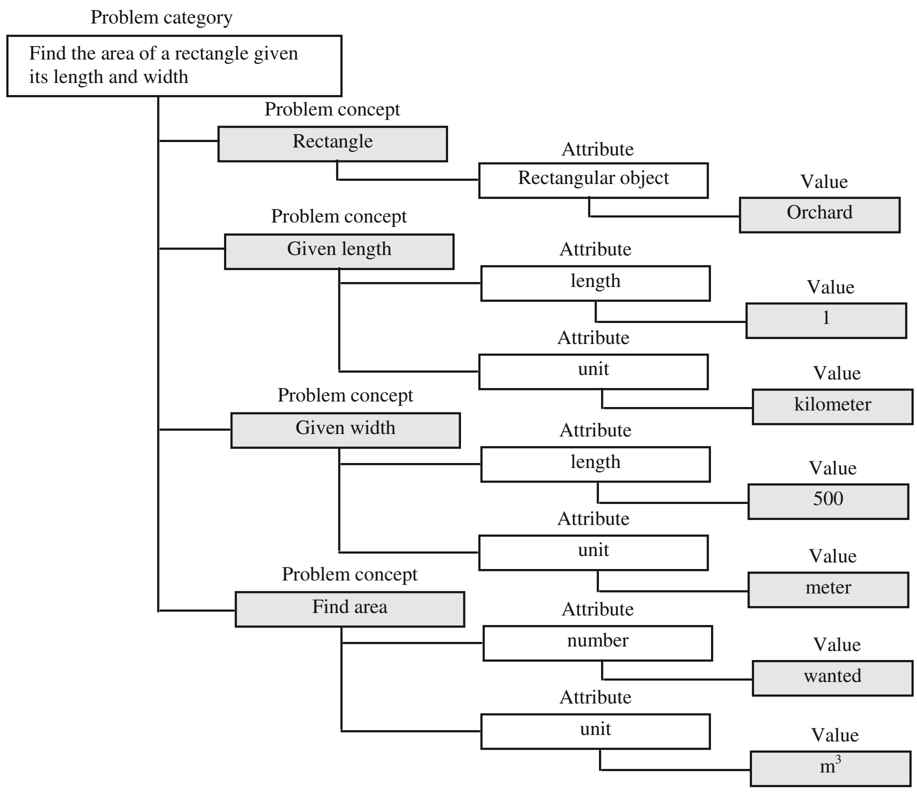

Fig. 2 Knowledge structure used in LIM-G

the learner on solution methods of model problems stored in the database. Such systems are not intelligent enough to comprehend and solve previously unseen problems given by the learner.

2.1.2 Geometry problem

A Learner-initiating Instruction Model, **LIM-G** (Wong et al. 2007) is a proposed CAI model that can deal with unknown geometry problems of certain types. After comprehending any problem it can provide necessary information, diagrams and incremental guidance to a learner to solve the problem. The domain of the system is guided by problems taken from Taiwan's elementary school level textbooks and are textual descriptions (without any graphical illustration) involving seven classes of geometric shapes including squares, rectangles, circles, triangles, trapezoids, parallelograms and sectors e.g. `'There is a rectangular land with length 1 km and width 500 m. What is its area?'`

LIM-G can comprehend a new geometry problem by extracting from it relevant information and representing it in a tree-type cognitive knowledge structure that contains hierarchical nodes of schematic knowledge, problem concepts and linguistic knowledge. For the above problem the knowledge structure is shown in Fig. 2 where the 'problem category' is the schematic knowledge and 'attributes' and 'values' are linguistic knowledge.

To solve the problem semantic knowledge like 1 km = 1,000 m, area of rectangle = length × width is also used. Using Scalable Vector Graphics (SVG), a tool for graphical applications in XML, simple diagrammatic representation of the problem is made out of the concept-attribute content of the problem. Also the content is used by the comprehension diagnosis module to ask relevant questions to the learner to assess his comprehension level and approach for solution.

LIM-G has a cognitive knowledge base constructed with an ontology-based knowledge engineering tool called InfoMap (Hsu et al. 2001) and containing generic template nodes for problem class, problem-concept, lexical knowledge, lexicon etc. Using template matching mechanism the exact category of a given problem and most fitting problem-concept set is determined to finally retrieve the concept's attributes and values from a problem.

Borrowing major ideas of WordNet (Miller et al. 1990; Miller 1990; Fellbaum et al. 1990; Fellbaum 1990; Beckwith et al. 1993; WordNet), a popular lexical database often used for natural language processing tasks, a simple knowledge base structure, termed as GeometryNet, is already reported (Mukherjee et al. 2007) that lists geometric entities (e.g. *line, parallelogram, circle, triangle, quadrilateral* etc.), entity parameters (e.g. coordinates, angle, slope, length etc.), entity attributes (e.g. *isosceles, concentric, common* etc.) and interactions/relations (e.g. *produce, intersect, bisect* etc.) between the entities. The purpose of GeometryNet is to facilitate machine understanding of school-level geometric problems stated in English and drawing relevant geometric figures automatically.

The basic framework proposed for the GeometryNet was based on relational pointers linking the different geometric terms—entity, attribute, relation each pertaining to a syntactic category—noun/adjective/verb and accordingly organized in three text files. Initially, a rather simplistic implementation is done with a single GeometryNet text file containing all the terms in random order (with no relational pointers as such), description of each term delimited by an 'end' statement as shown below (for entity 'parallelogram').

```
.........parallelogram
n=4
vertex_1
vertex_2
vertex_3
vertex_4
side_1
side_2
side_3
side_4
#((x2-x1)^2+(y2-y1)^2)^0.5=((x4-x3)^2+(y4-y3)^2)^0.5
#((x3-x2)^2+(y3-y2)^2)^0.5=((x4-x1)^2+(y4-y1)^2)^0.5
#m1=m3
#m2=m4
end.........
```

The # indicates parameter conditions. The parameter conditions of parallelogram entity are the four mathematical relations prefixed with # and depicting 'opposite sides of a parallelogram are parallel and equal in length'. One of the characteristics of this GeometryNet structure is inheritance of properties of the subordinate entities from the superordinate entities and thus repetition of information is avoided. For example the information that side_1, side_2, side_3, side_4 (of a parallelogram) each have two points and a slope m is not listed explicitly against each 'side' rather those information are found by correlating the generic facts that 'side' *is a*

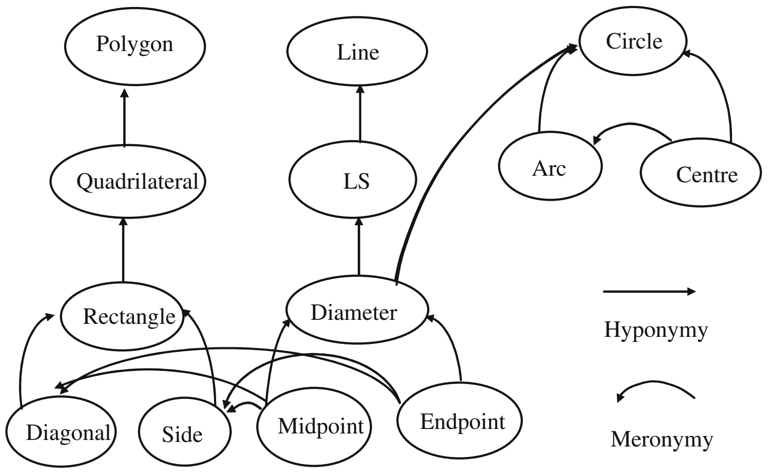

Fig. 3 Network representation of GeometryNet relations of geometric entities

ls (line segment) and ls *has* point_1, point_2 and *m*. When a program searches for knowledge about 'parallelogram' in GeometryNet it triggers sequential searching of knowledge about 'vertex', 'side', 'point' and 'ls' entity types each of which is listed separately.

There are total 51 odd varieties of geometric entity listed in GeometryNet which are found after analyzing around 300 school-level (standard VII to X) geometry problems. The entities listed exhibit typical subordinate–superordinate (hypernym–hyponym) and part–whole (meronym–holonym) relation (refer Fig. 3) that characterizes the framework of GeometryNet.

Similar to nouns 50 different entity attributes (adjectives) are listed in the GeometryNet file, for example—

```
.........concentric
circle_1
circle_2
#r1!r2
#xc1=xc2
#yc1=yc2
end.........
```

This implies 'concentric' relates two circles 'circle_1' and 'circle_2'. The generic parameters of circle (i.e. xc, yc, r) are derived from the data set of 'circle' already listed as an entity. The three #conditions depicts mathematically that the centers of the circles are identical.

Total 35 verbs are listed in the GeometryNet file, each term describing some sort of entity to entity interaction or relation, for example—

```
.........meetsll
ls_1
ls_2
point_5
#y5-m1x5=y1-m1x1,y5-m2x5=y3-m2x3
#m1!m2
end
```

The system using GeometryNet have an NLP component that first analyzes the text and summarizes the information of a geometry word problem. Taking this summarized information as input the GeometryNet is consulted to extract other relevant information (not provided in the problem) to understand the complete geometric meaning of the problem. The machine understanding is next converted into a language-free intermediate mathematical representation (Mukherjee and Garain 2009) suitable for diagram drawing. Finally the equivalent figure is drawn with input from previous stage using computer graphics routines.

2.1.3 Rate problem

The program, **CARPS** i.e. Calculus Rate Problem Solver (Simmons 1970; Charniak 1968, 1969) written by Eugene Charniak reads, understands and solves rate problems stated in English. The program is similar to STUDENT (Bobrow 1964a, b), the primary difference being introduction of 'structures' as the internal model of representing information extracted from a problem. The structure is basically a tree which has its head the name of an object, and at various levels beneath the head all the information the program could abstract from the problem.

The program has solved 14 rate problems pertaining to two types—distance and volume, mostly taken from standard calculus texts. For example, consider the following rate problem:

```
(WATER IS FLOWING INTO A CONICAL FILTER AT THE RATE OF
15.0 CUBIC INCHES PER SECOND. IF THE RADIUS OF THE BASE
OF THE FILTER IS 5.0 INCHES AND THE ALTITUDE IS 10.0 INC-
HES FIND THE RATE AT WHICH THE WATER LEVEL IS RISING
WHEN THE VOLUME IS 100.0 CUBIC INCHES.)
```

Upon reading this problem the program ascertains the type (distance or volume) it belongs to and the relevant formulae stored in memory by searching for certain clues and keywords (e.g. `'LEVEL'` and `'CONICAL'`) in the sentences after parts of speech tagging of the words. Next each sentence is decomposed into simpler sentences by finding match with stored patterns of sentences. For example the second sentence of the problem is matching with the pattern `'IF-ANYTHING-,-QUESTION WORD-ANYTHING'` where the question word is `'FIND'`; the associated operation is to break the sentence into two simpler sentences namely `'IF THE RADIUS .........10.0 INCHES'` and `'FIND THE RATE.........100.0 CUBIC INCHES'`.

Once the sentence decomposition process is over, obtaining six simple sentences of the like of '`((THE RADIUS OF THE BASE OF THE FILTER (IS VERB) 5.0 (IN UNIT))`' the transformation module translates each of these simple sentences into substructures by identifying the basic form of a sentence (e.g. `'NVP-IS-NUMBER-UNIT'`) and/or its noun phrase (e.g. `'ANYTHING-OF-A/THE/PRONOUN-ANYTHING'`) from a stored list. The substructure corresponding to the simple sentence mentioned above is the 2nd branch from right (`'FILTER-BASE:G0016-RADIUS:G0017-VALU:(5IN)'`) in the complete tree structure (Fig. 4) constructed by coherent integration of all the substructures. Here `G0017, G0018` are symbols generated by LISP system and are known as GENSYMS.

Using the information gathered in the tree structure and the knowledge it stores about the word 'conical' the program finally forms three equations relevant to the problem. For instance, the equation it forms for the question sentence of the example problem is:

```
((EQUAL (G0020) (DERIV (G0019 WATER FILTER)))
```

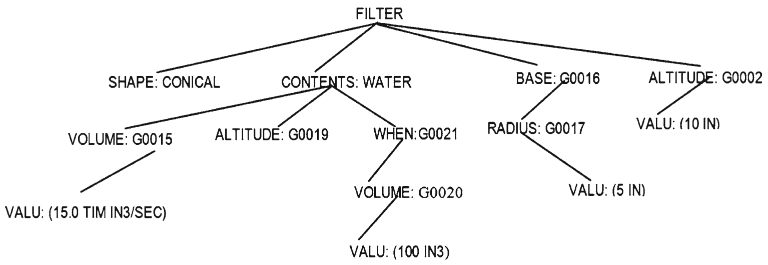

Fig. 4 Tree structure used by CARPS for representing information extracted from a problem

The usual notation of this equation is `G0020`$=\frac{d}{dt}$`(G0019)` where value of `G0020` will be the answer of the problem. The basic formulae retrieved from storage to form the other two equations are: volume $= 1/3 \times \pi \times \mathrm{RC}^2 \times \mathrm{AC}^2$ and $\mathrm{RC} \times \mathrm{A} = \mathrm{AC} \times \mathrm{R}$, where RC and AC are respectively the radius and altitude of water content in the conical at any time, R and A are respectively the radius and altitude of the conical.

2.1.4 Probability problem

A program which solves basic probability problems stated in English is the **HAPPINESS** program written by Gelb (1971a, b). 'HAPPINESS' stands for 'Heuristic Analysis of Probability Problems in a Natural-language Environment with Symbolic Solutions'. Given an input problem, `WHAT IS THE PROBABITY OF GETTING TWO OR MORE HEADS OR EXACTLY 3 TAILS WHEN FOUR COINS ARE TOSSED ONCE? the` output given by the program is: `15/16 or 0.9735`

HAPPINESS is similar to Charniak's CARPS; it builds a tree structure or descriptor-list (desclist) representing a problem and selects a solution method based on the occurrence of keywords (e.g. dice, cards, coins) in the problem statement. The language analyzer module decomposes the input sentences into simple clauses and phrases (e.g. `((TOSS/PASSIVE) (4 COINS) (1 TIMES)))` by some pattern matching. Next, semantic scan (finding keywords as 'coin', 'distribute' etc. to ascertain the type of problem) and subsequently syntax analysis is done on these simple clauses using a context free grammar; the result is a symbolic description of the simple probability events described in the problem, together with a specification of the combination of these events required to solve the problem. This information, combined with the results of the semantic scan, constitute the 'desclist' of the problem as shown in Fig. 5.

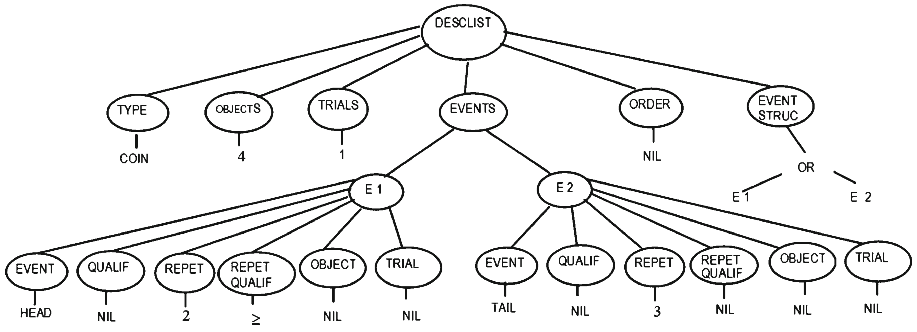

Fig. 5 Desclist as used by HAPPINESS for the example problem

Using the desclist information the solution generator module then produces the solution, first by producing symbolic result (e.g. P(E1) + P(E2), E1: $\geq$2 heads, E2: 3 tails) followed by expansion of symbolic probability terms entailing permutation, combination, fraction and arithmetic operation (e.g. P(E2) expressed as $(4/3)(1/2)^3(1/2)^1$ and P(E1) expressed as $(4/2)(1/2)^2(1/2)^2 + (4/3)(1/2)^3(1/2)^1 + (4/4)(1/2)^4(1/2)^0$) and finally numeric result (15/16 or 0.9375). Heuristics are applied when the 'desclist' information is insufficient or incomplete to expand the symbolic result.

2.1.5 Matrix problem

An interactive natural language problem solver has been developed by Biermann et al. (1982) that allows user to manipulate object displayed on a computer terminal via typed or spoken English sentences (commands). It is an improvement over the earlier system NLC (Ballard and Biermann 1979; Biermann and Ballard 1980) that provided a matrix computation facility and allowed users to display matrices, enter data and manipulate the entries, rows and columns using natural language commands. The interactive solver can additionally execute voice and tactile (touch) commands as: `'Find the largest number in this column'` (*pointed on the screen by finger*), `'Add this column to that column putting the result here'` (*three touch input*).

The basic system design includes following functional modules: (1) token acquisition, (2) parsing, (3) noun group resolution, (4) imperative verb execution, (5) flow-of-control semantics and (6) system output. The token acquisition module receives typed inputs, guesses word from the speech recognizer and screen coordinates from the touch panel. These inputs are preprocessed and passed to the parser which uses an augmented transition network to discover the structure of the command and the roles of the individual tokens. Noun group resolution attempts to discover what domain objects are being referred to and the verb execution module transform those objects as requested by the imperative verb. The flow-of-control semantics module manages the execution of meta-imperative verbs such as 'repeat' and handles user-defined imperatives. Finally the system output module displays the state of the world on the screen.

2.2 Problems of physics

2.2.1 Mechanics problem

The **NEWTON** program written by Kleer (1975a, b) is an expert problem solver in the domain of mechanics, specifically, relating to kinematics of objects moving on surface. By employing different representation and reasoning techniques NEWTON demonstrates how to solve simpler problems through simpler steps or techniques than those required for complex problems.

To solve a kinematics problem NEWTON first employs qualitative knowledge or arguments to predict the sequence of possible path/position of object movement and also different states of the object at different times. This prediction process is termed as 'envisioning', which, for example, tells without much calculation that an unsupported object will fall. The system extracts the gross kinematics features from a problem and represents the envisioned path/positions in a tree-type structure, each node of the tree being a possible position of the object. For instance, consider the envisioning tree (Fig. 6b) that may be constructed to qualitatively represent the following problem: `"whether the ball rolling along the`

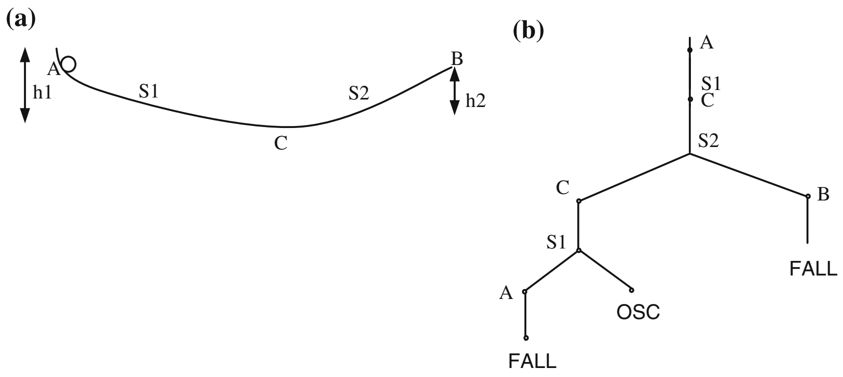

Fig. 6 **a** The ball rolls along path A–C–B. **b** Envisioning tree for the rolling ball problem

`frictionless curved surface could reach point B starting from point A?"` (Refer Fig. 6a). The tree describes that the object starts at point A, slides through segment S1, reaches point C, slides through segment S2, either slides back on segment S2 or reaches point B and so forth. So the possible answers to the above problem can be predicted by tracing the tree only and without necessitating any numerical analysis by using 'conservation of energy' rules etc. But in order to answer more detailed questions (referring the same problem) like `"what will be the velocity of the ball at point B?"` quantitative knowledge like heights and distances of points A and B, other relevant variables and constants and basic kinematics rules/formulae are required.

Quantitative knowledge is organized in terms of FRAME structure, which packages together mathematical equations involving the variables, like the following one.

```
FRAME energy OF object surface t1 t 2
         VARIABLES:
               (vi: VELOCITY OF object AT TIME t1,
                vf: VELOCITY OF object AT TIME t2,
                h : HEIGHT OF surface)
         vf² - vi² = 2 G h.
```

NEWTON maintains a database (McDermott and Sussman 1974) to represent the semantics of variables. On carefully analyzing the envisioning tree the relevant FRAMES are invoked, a subset of which is instantiated with variables and values taken from the problem and subsequently examined for solution.

The program **MECHO** developed in Prolog by Bundy et al. (1979a, b) solves mechanics problems stated in English in the areas of pulley problems, statics problems, motion on smooth complex paths and motion under constant acceleration. The underlying focus is to get a formal representation of a problem from an English statement and to use this representation as well as an expert inferential system to solve the problem. From input text to the final solution the program works through the stages shown in boxes in Fig. 7 using various types of knowledge input (shown as labels of arrows in Fig. 7).

Upon parsing a sentence a set of assertions are produced (by instantiating out the referents) about the objects in the problem. Some of the assertions produced for an example

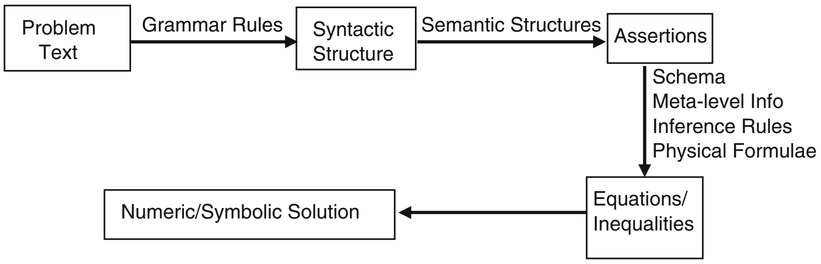

Fig. 7 Schematic diagram of the processing stages of MECHO

problem statement `'Two particles of mass m1 and m2 are connected by a light string passing over a smooth pulley'` are:

```
isa(particle,p1), isa(pulley,pull), coeff(pull,zero),
mass(p1,mass1,period1), measure(mass1,m1),
given(mass1)
```

Schema is a structure of certain key words and object configuration and is required to supplement for default information ('house rules' in the domain) which are not given explicitly in the problem. For example, a typical pulley system schema where the objects satisfy ideal-type constraints is defined as,

```
sysinfo ( pullsys,
        [pull,str,p1,p2],
        [pulley,string,solid,solid],
        [  supports (pull,str),
           attached (str,p1),
           attached (str,p2)]).
```

The input assertions provide meta-level information about whether certain quantities are given or sought. Moreover, before using an equation its applicability in the context of the problem is checked and/or the situation is created whereby an equation can be applied using meta-level information/reasoning. The following is one of the meta-level structures used for the above problem statement,

```
Isform( resolve,situation(Obj,Set,Dir,Time),
        F = M * A)
   <-- mass(Obj, M, Time) &
      accel(Obj,A,Dir,Time) &
      sumforces(Obj,Set,Dir,Time,F).
```

A general knowledge of mechanics which is required to solve a problem is formalized in a set of inference rules (object-level) e.g.

```
Relaccel(p1,p2,zero,Dir,Period)
    <--constrelvel(p1,p2,Period).
```

This rule states that the relative acceleration between two points p1, p2 is zero if there is a constant relative velocity between them over a certain period.

MECHO demonstrates how the technique of using and controlling knowledge about the domain by inference at the meta-level can be applied to a wide range of mechanics problems. It clearly exploits the advantage that meta-level knowledge representation has over object-level knowledge representation.

2.2.2 General physics problem

ISAAC program written by Novak (1976) can read, understand, solve and draw pictures of physics problems stated in English. It has been successfully tested with 20 problems taken from high school and college physics text. One such problem is:

```
The foot of a ladder rests against a vertical wall and on
a horizontal floor. The top of the ladder is supported
from the wall by a horizontal rope 30 ft long. The ladder
is 50 ft long, weighs 100 LB with its center of gravity
20 ft from the foot and a 150 LB man is 10 ft from the top.
Determine the tension in the rope.
```

Along with numerical answer (`120.00000 LB`) a figure (Fig. 8) is also outputted by ISAAC for the above problem.

The schematic diagram (Fig. 9) shows the major stages of processing done by ISAAC. The parsing programs transform the linear string of words into a structure that explicitly expresses the syntactic relationships of the words and phrases. The parser programs written in pure LISP actually implements the grammar of a phrase as an augmented transition network (ATN) influenced by the ATN formalism of Woods (1970); the grammar functions parse the phrases or clauses into a network of interlinked nodes which bear strong resemblance to the semantic networks proposed by Simmons (1973). Each node in the network is a GENSYSM atom whose name is 'TOK' followed by a number (e.g. TOK290) and represents the 'main' word of a phrase (e.g. TOK290 can represent 'ladder'). Features of the nodes are stored on its property list under named indicators like 'LFRAME' (indicating type of 'linguistic frame' like noun phrase—NP, verb phrase—VB etc.), 'MODS' (modifier) etc. Then the semantic routines associated mostly with the verbs and prepositions of a phrase are invoked to determine the proper sense-meaning of the words just like the phrase 'top of the ladder' having the preposition 'of' belongs to the sense-meaning class '<location>OF<object>'. Some semantic routines identify the referents (e.g. 'top of') and modifiers (e.g. 'a 150 LB man') of the noun phrases. Finally each parsed structure is transformed into semantic frame (SFRAME) form whereby more indicators are added to the property list of the nodes or totally new frames created. Some of these indicators explicitly denote the link of a node or a phrase to other objects (SEMOBJ) or referents (RFNT) in the problem to which the node or phrase refers.

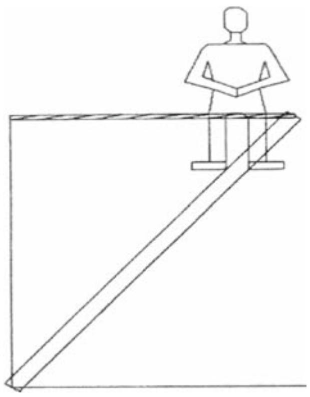
Fig. 8 Picture output of ISAAC for the example problem

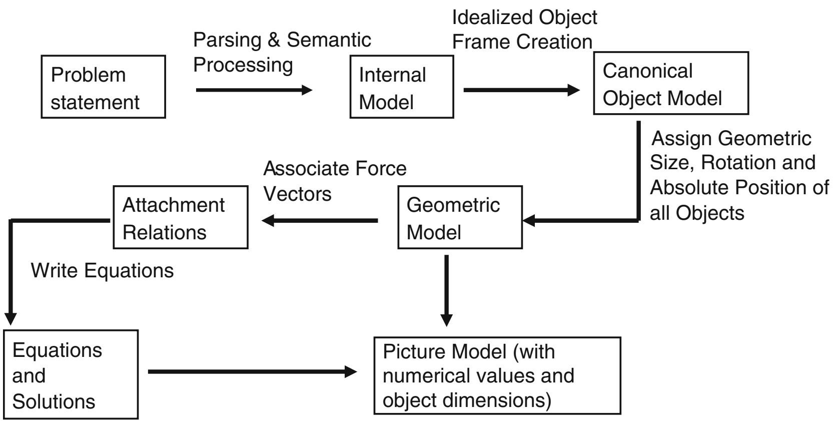
Problem statement
Parsing & Semantic Processing
Internal Model
Idealized Object Frame Creation
Canonical Object Model
Assign Geometric Size, Rotation and Absolute Position of all Objects
Geometric Model
Associate Force Vectors
Attachment Relations
Write Equations
Equations and Solutions
Picture Model (with numerical values and object dimensions)

Fig. 9 Schematic diagram of the processing stages of ISAAC

Thus SFRAME are designed to reflect the attributes and relationship of objects as described in the problem.

Completion of syntactic and semantic processing of all the problem statements ultimately results in formation of the internal model of a problem which is nothing but a collection of SFRAMEs like the one shown below.

```
LADDER291((TOK.LADDER) (ENTITY.PHYSENT) (LOCS LOC297
LOC309 LOC317 LOC322) (ATTACH ATTACH298 ATTACH299
ATTACH310 ATTACH323) (SUPPORTBY FLOOR296 WALL294
ROPE305) (COFG LOC317) (LENGTH 50 FT) (WEIGHT 100 LB))
```

Here `LOC297,LOC309` etc. under the indicator LOCS are list of all locations like ‘foot’ or ‘top’ etc. of the ladder, ATTACH298, ATTACH299 etc. are list of all ATTACHments concerning ladder like the one between the foot of ladder and floor, another between the foot of ladder and wall etc. COFG denotes the location of the center of gravity of the ladder by pointing to the SFRAME `LOC317.` `LOC317` describes a location of the ladder 20 ft from its foot as shown below.

```
LOC317 ((FRAME.LOCATION)(ENTITY.LOCATION)(OBJECT.LAD-
DER291) (LOCNAME.FOOT) (REFLOC.LOC297)(RELPOS FROMLOC
(20 FT)))
```

There is provision in ISAAC for representing some common-sense knowledge about usual features and relationships of particular type of objects programmatically (e.g. a procedure to decide which object is referred to as the ‘force’ in the problem) or in the form of data structure (e.g. typical geometry and location names of a ‘lever’). Such knowledge is required in addition to the information extracted from the problem to make the internal model complete for understanding a problem.

To convert the internal model into a model suitable for writing equations describing the interaction of the objects of a problem canonical object frames (idealized objects) are chosen for all the objects in the model. A person, for example, might be modeled as a WEIGHT (having no geometric size) when sitting on a pole or as a PIVOT when carrying the pole. A total of seven canonical object types are used in ISAAC: LEVER, WEIGHT, SPRING, PIVOT, ROPE, SURFACE and FORCE. After a canonical object frame is made for an object its geometric size and its absolute rotations are found or assumed and absolute location names and coordinates are assigned to all of its locations. This helps construct the geometric model of the problem which basically represents the spatial position and orientation of each object.

Next, the frame completion routine is invoked by ISAAC for completing the attachment relations by associating appropriate force vectors for each object. The forces exerted by an object and the geometric position of the point at which each force is exerted are collected and put in the attachment frame under the indicator FORCES. The next functional module called by ISAAC is the problem-solving module; the function ATTDRIVER write equations for each attachment relation according to the physical laws like sum of X forces and sum of Y forces must each be zero for an object in static equilibrium using LISP format alike those used by STUDENT program (Bobrow 1964a, b). For example,

```
(EQUALS 0 FORCE179)
(EQUALS 0 (PLUS (TIMES TENSION327 -1.00000) FORCE 332))
```

Similarly the ‘SOLVELEVER’ routine calculates the moments of the forces (identified as acting on the lever) about the PIVOT point, sums them and sets the sum to zero. The equations so formed are solved by SOLVEQ and PSOLVER routines.

Lastly a picture model of a problem is generated to allow a diagram of the problem to be drawn. This model is similar to the geometric model, except that all dimensions of objects are numerical. Sizes are chosen for objects, which have zero size in the geometric model (e.g. a person represented as a point mass).

The program **BEATRIX** (Novak and Bulko 1993) is an advancement over ISAAC. It understands not only text but also diagrams and can thus solve physics problems described in terms of statements and supporting illustrations. It uses a single unified model that incorporates information from both text and diagram and establishes correspondence between parts of the text and diagram that refers to the same object or feature. To simulate the natural human approach of reading and understanding a problem with illustrations BEATRIX is organized to allow co-parsing of the two input modalities using the BB1 blackboard system.

ALBERT developed by Oberem (1987) is an intelligent tutoring (CAI) system that not only understands and solves physics (more specifically kinematics) problems but can also teach a student how to solve them. An intelligent discussion of a kinematics problem with a student is a more complex task than understanding a textbook physics problem but ALBERT accomplishes both using separate natural language systems for each task. A lexical database of words grouped into 25 categories based on their semantic function is maintained. Traditional linguistic groupings such as noun and verbs are not used. In the first pass (lexical analysis), the words in the input text are parsed into numerical string where each number in the string corresponds to the lexical category of the associated word. Regular patterns occur in such numerical strings and around 100 commonly occurring syntactic patterns are maintained in ALBERT’s knowledge base. In the second phase of the parse, the numerical string derived earlier is searched for matching syntactic patterns. When a particular pattern is identified a parameter-driven semantic routine is called to extract information from the corresponding text string. A small number of data elements are associated with each syntactic pattern to act as parameters of the appropriate semantic routines. When semantic processing is complete, all the information in the input will have been used to instantiate variables in the computational model of the solver.

FREEBODY (Oberem et al. 1993) is an adoption of ALBERT and a speciality software that understands the free-body diagram of a physics problem drawn by a student in a graphic user interface; it assesses whether or not a diagram is reasonable in terms of situation described in the problem statement and dynamically guides the student while each force is drawn by him. More or less the same lexicon and knowledge base as used in ALBERT is used by this program. The semantic routines are restructured to deal with more qualitative data, since there is no numerical problem solving in FREEBODY.

2.3 Chemistry problem

ECLIPS (Oberem 1994) is a program designed to understand and solve chemistry problems of molarity stated in English. The program closely resembles ALBERT as it uses both the language system and generic problem solver of ALBERT with minor modifications in the semantic routines. One additional semantic routine was needed to identify chemical formulae and to extract information from them. An example of a problem solved by ECLIPS is: `How many moles of H`$_2$`SO`$_4$` are required to make a 0.1 M solution with a volume of 375 ml?`

Information extracted from the problem and relevant knowledge gathered from knowledge base by ECLIPS are: the solute is H_2SO_4–atomic mass = 98.08 a.m.u—the mass of a mole is 98.08 g, the volume of the solution ($= V$) = 0.375 l, solution strength ($= c$) = 0.1 M, number of moles ($= n$) is required to be calculated. The solution steps are:

Identify the relevant formulae: $c = n/V$
Rearrange the formula (to keep the unknown variable on the left hand side): $n = cV$
Put the values of $V = 0.375$ and $c = 0.1$ in the above formula
Find $n = 0.037$ mol

2.4 General numerical problem

A language based problem solver was designed and implemented by Ramani (1969, 1979) that can respond to a problem stated in natural language based on analogy with similar problem, solution steps and answer formats stored and learnt by the system previously. The system uses an analogy directed language processor that works irrespective of any specific language. The response of the processor to any situation is based on pattern setting examples called ‘paradigms’ given earlier. A ‘paradigm’ consists of a ‘context’ (C), an input (Q) and a response or output (A) and is stored in a form called ‘schema’. A database storing sentential strings (questions, commands or informational sentences etc.) is consulted and the schema is extrapolated to create responses to new situation or ‘stimulus’. Considering a simple paradigm as:

```
C: The density of lead is 8 gm per cc
Q: Find the weight of a lead sphere 6 cm in diameter
A: Find the volume of the sphere
     The volume is 0.75 * PI * (0.5*6)^3
   Multiply the volume by 8
   The result is the weight of the sphere
```

given a new situation,

```
C: The density of a metal is 10 gm per cc
Q: Find the weight of that metal sphere 4 cm in diameter
```

the processor would logically work as following to find the weight:

```
A: Find the volume of the sphere
     The volume is 0.75 * PI * (0.5*4)^3
     Multiply the volume by 10
     The weight of the sphere =
```

'Schematization' of paradigm serves essentially to assign a structure to the 'paradigm' facilitating extrapolation. Any input stimulus S, a string of words and symbols, triggers off a search activity in the processor. The search is for the 'schema' of a 'paradigm' p[C,Q,A] which has an input component, Q, maximally resembling S. If no suitable schema is found for a given 'stimulus' S, it is treated as informative input to be directly sent to the sentential store.

2.5 Proof presentation

So far most of the system descriptions focus on the direction from natural language to machine readable input data/information. It is also of interest of the present study how a machine-generated mathematical solution (or proof) is translated back into natural language in order to output readable information to the user. Presentation of mathematical proofs in natural language to the user is one such problem addressed by few automated systems.

Proving of mathematical theorems by a computer program or automated deduction, is a matured subfield of automated reasoning (AR). Many systems have been developed so far that produces proofs in formal language like first-order predicate logic but these are quite complex and hard to understand. There have been attempts to express such proofs in a more natural or intuitive way using higher order and modal logics but those are also not as expressive as the text book proofs made in informal natural language. The research in this field took a new direction when transforming logic-level natural deduction (ND) proofs into natural language was first attempted by (Chester 1976) in his system EXPOUND. Equipped with more advanced techniques of natural language generation, a more coherent translation was obtained by the MUMBLE system of McDonald (1983). A more recent attempt can be found in THINKER (Edgar and Pelletier 1993), where different styles of explaining ND proofs are exploited. Meanwhile there is significant contribution by Lingenfelder (1989, 1990) and Lingenfelder and Pracklein (1991) who used different kind of heuristics to improve the quality of ND proof in terms of making those more structured and direct with lesser redundancy. This result was exploited lately by Huang (1996) who introduced an intermediate representation (assertion level) of the ND proofs that facilitates generation of easily understandable natural language proofs.

The prototypical proof presentation system PROVERB (Proof Verbalization) employed by Huang et al. (1992) is embedded as the explanation component in an interactive proof development environment called Ω-MKRP. Any prolog-based theorem prover invoked (as logic engines) in Ω-MKRP produces, from an input theorem, the logic level ND proof which is translated in several steps into an assertion level proof. It is basically a *proof tree* containing mostly inferences (or proof plans) in terms of semantically meaningful operators, which apply a definition or a theorem or a previously proved lemma valid in the context. PROVERB takes as input an assertion level proof and generates a sequence of preverbal messages (referring the domain knowledge), from which a proof in natural language can be produced by an appropriate grammatical treatment done by TAG-GEN—a tag based syntactic generator (Refer Fig. 10).

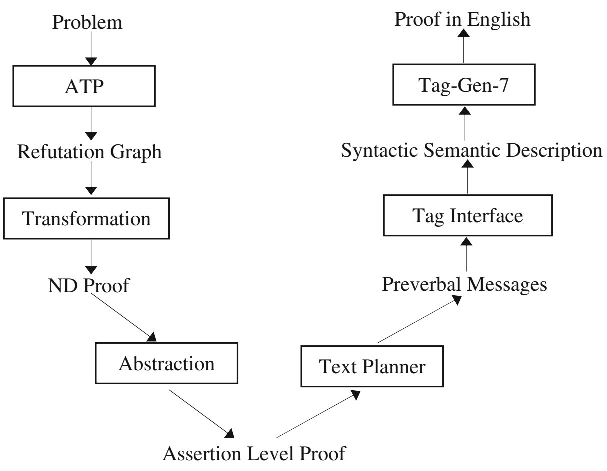

Fig. 10 Different stages from input problem to proof generation

Following is an example of a preverbal message.

```
(Derive Reasons:((((ELE a U) explicit))
                  ((SUBSET U F) omit))
        Conclusion:(ELE a F)
        Method:(Def-Subset omit))
```

The natural language output from the above message is: "`Since a is an element of U, a is an element of F`"

In this context we must mention some recent work that concerns understanding and verifying natural language mathematical proofs. Though the earliest experiment in the said domain dates back to 1960 when Paul Abrahams implemented a Lisp program named PROOF-CHECKER for the machine verification of mathematical proofs, lately Zinn (1998) attempted to implement a prototype system (using Prolog) for automatic verification of text book proof on elementary number theory. Later Zinn (2003) presented a novel computational framework for understanding informal mathematical discourse by employing a proof planner and a proof representation structure (PRS). Progressing in the line few tutoring systems evolved that guided a student attempting to build the proof steps of a mathematical theorem by conducting a natural language dialogue with the student. Proof steps are analyzed and evaluated by the system to give domain-specific hints/corrections (if required) that extends the current partial proof towards the complete proof. Zinn et al. (2002) proposed a 3-Tier planning architecture for managing tutorial dialogue in mathematics. Subsequently, a mathematical tutoring system DIALOG (Benzmüller et al. 2007; Pinkal et al. 2008) is developed with an elaborate natural language dialog management component. Motivated by the results of DIALOG project Buckley and Dietrich (2007a, b) augmented the standard mathematics tutorial model by modelling the explicit task-at-hand (of the tutor) within the dialog manager. The task model includes current_task_ status, subtask_status, last_proof_step, evaluation, proof_step_history, hint_history etc that should facilitate appropriate behaviour of the simulated tutor.

3 Discussion

While all the systems described above accept English sentential input only in typed text form, Biermann's work (Biermann et al. 1982) is unique in the sense it allows, though in a very restricted manner, the user to state a problem through typed, voice and touch commands (imperative sentences) in English. The work is interesting because speech and touch are two of the most natural input modes for a computer system simulating human problem solving mechanism. We should also mention Beatrix (Novak and Bulko 1993) in this context because it can recognize diagrams apart from textual input. These two systems, though not proved to be based on a generic and scalable model, point to a future direction of research in this field.

Bobrow's STUDENT (Bobrow 1964a, b) will always have special mention as it is the first natural language problem solver in the mathematics domain. Bobrow used the concept of relational model (object-relationship-media) quite effectively in STUDENT as was used by Lindsay (1963) in his SAD SAM program for internal representation of the input problem. This model proved to be quite useful for algebra problems as standard techniques are there for finding numerical values that satisfy sets of simultaneous equations, thus taking full advantage of the relational language media. The main challenge of Bobrow's approach was to map the English sentences directly into the relational model. Though it worked for a number of algebra story problems it has definite limitations. The technique works only when the sentences express algebraic relationships among quantities. The 'variable' phrases must be similar in each occurrence so that they can be matched properly and the key words must not be used in multiple ways which might confuse the pattern matcher. Moreover, the method adopted for extracting simple sentences from complex and compound input statements is adhoc (heuristic) and primitive and might not work for sentences of great grammatical complexity. It may be noted that STUDENT did not require storing knowledge except having a glossary of few global facts.

DEDUCOM (Slagle 1965) developed after STUDENT was a much simpler application; it identified the phrases of input sentences that provided some data and linked it to the relevant inference rules to find solutions to simple arithmetic problems. The program explores the deductive power of inference rules in the form of conditionals with variables and transformations—another form of the pattern operation rule. Although this form of transformation is powerful enough to deduce an answer to a problem, given appropriate data statements, its application to a large set of data is very expensive without the inclusion of appropriate heuristics.

Much work has been done in the field of cognitive science to simulate the psychological process involved in human understanding of word problems. WORDPRO is such a program developed by Fletcher (1985) in Interlisp-D that implements a theory (Kintsch and Greeno 1985) of comprehension and solution of a restricted set of arithmetic word problems used by a child. WORDPRO has distinct limitation—it cannot accept problem sentences in its natural form. It begins with a set of propositions that represent the text's meaning. On the other hand WORDPRO is quite matured in the sense that it uses a rule base, a knowledge structure in the form of set schema and higher level schemata for representing the problem and also short-term (STM) and long-term memory (LTM) for cyclical processing of information. WORDPRO, however, was tested only with 14 problems; its success is not as a generalized arithmetic word problem solver but in successfully demonstrating the theory of cognitive science upon which it is based. Though the program attempts to simulate human approach in understanding and solving an arithmetic problem it cannot adapt itself to different levels of ability as like the humans do. Successful computer implementation in the domain of one-step arithmetic word problems was first done by Briars and Larkin, followed by Fletcher and

Dellarosa. But enough intelligence was not built into these programs to handle information that is not relevant to the problem. Dealing with extraneous information is a problem in itself for humans while solving a problem.

The pioneering study that elucidates the strategies children use to cope with irrelevant information in a particular type of one-step word problems is the study of Littlefield and Rieser (1993). Their proposed model attempts to logically identify only relevant subsets of a superset (quantities requested in the problem) towards understanding the actual problem. This study influenced the recent development of ROBUST (Bakman 2007) that marks significant advancement over its ancestors as it can handle problems stated in a free format English (using a natural language parser) and that too containing large amount of extraneous information. It shows that transposition of problem sentences and/or addition of irrelevant data to the problem text does not affect the understanding or solution of the problem. The overall strategy of the ROBUST model is to create schema instantiation and attach the relevant ones to LSI every time an elementary change verb is encountered. Though the relevancy of a schema cannot be ascertained until the problem is solved, ROBUST uses a heuristic method to estimate schema relevancy at an early stage thus discarding in all possibility the extraneous data. This effectively fosters the understanding and solution process. We must note that ROBUST cannot be termed as a general arithmetic word problem solver as it cannot handle problems that entail operations other than addition/subtraction to solve.

Following an instructor-initiating strategy most CAI systems use a pre-designed database of systems and do not address the issue of understanding or solving unknown and non-standard problems. In contrast, LIM-G (Wong et al. 2007) is an intelligent tutor that does not use any database of model problems or standard solution steps. Rather it uses a powerful knowledge base and can not only comprehend standard problems but can also deal with those having missing or extraneous information. The system is tested with fairly large number of school-level elementary geometry problems and yielded an impressive success rate of about 85%. However, it cannot fully comprehend problems that involve more than one shape and cannot deal with problems which involve complicated geometric shapes other than the seven classes of original shapes defined in the knowledge base. Any geometric condition or constraint satisfaction is also not within the scope of the system which limits its utility to picking the correct formulae for a geometric shape and substituting numeric values for variables. It is yet to be explored whether LIM-G model is extendable to other domain as claimed by the authors.

CARPS (Charniak 1968, 1969) written in LISP and CONVERT to solve calculus word problems is most contemporary to Bobrow's STUDENT. These two systems are similar in the sense that both takes a heuristic approach to the analysis of English sentences using pattern—operation rules for both syntactic and semantic operations. Significant in Charniak's approach is the single tree structure of internal representation of a problem. By virtue of this structure using 'marker' or backward pointer CARPS is better able to recognize that 'the filter' in the second sentence of a problem refers to 'the conical filter' already established in the data structure by the first sentence. Similarly 'the altitude', 'the water level', 'the volume' all refer back to the attributes of 'the conical filter'. Whereas STUDENT has only one solution method (i.e. solution of linear equations) CARPS has several (simplification, differentiation etc.) and can decide which is appropriate for a given problem. Unlike earlier developed math word problem solvers CARPS stores knowledge like length, area, volume formulae in terms of basic parameters like radius, altitude, thickness etc. about shapes like conical, trough, spherical etc. and refers those while forming the final equations with values of parameters substituted from tree structure. However, it cannot solve rate problems that require some 'common sense' knowledge to supplement missing information in a problem statement. As Charniak pointed out, one weakness of the program is its dependence on the keywords to identify the type of the problem and the method of solution to be used. Same word ('keyword') may concern dissimilar categories of problem entailing different solution formulae when used in differing problem-context. Thus keyword based fixed solution selection can lead to complete erroneous result.

HAPPINESS (Gelb 1971a, b), written in LISP (1.5), is equally weak in that sense though more sophisticated techniques for syntactic and semantic analysis were postulated. The tree structure used by HAPPINESS is slightly different from the one used by CARPS as the nodes are categorically pre-specified while the leafs of the tree refer to the semantic objects. Probability problems are more diverse than simple arithmetic or calculus rate problems and hence demands specially tailored solution function as opposed to mere substitution into a generalized form—in that way HAPPINESS is more flexible than STUDENT or CARPS which deal with limited formulations.

The use of tree structure is further observed in the kinematics problem solver NEWTON (Kleer 1975a, b). Qualitative information derived from a problem is represented in terms of tree which Kleer calls 'envisioning tree'. Probably the first time use of 'frame' structure is seen in NEWTON. 'Frames' here act as quantitative knowledge structure describing relations (equations) between variables concerning particular kinematics concepts or events. Though it might have some apparent similarity with so called procedures it is important to note that frames are not procedures. This dual representation scheme (tree and frame) made possible using varied degree of argument and knowledge depending on the level of difficulty of the problem just like humans do. The concept of 'envisioning' as applied by Kleer might be useful in areas where problem solving depends on 'mechanical intuitions'. But Kleer's model works for single point object in one dimension only; it remains to be seen how the model can be made more generalized so as to handle two dimensional kinematics involving motion and interaction (collision) of more than one object. Another limitation of NEWTON is its inability of exception handling (e.g. occurrence of multiple roots) arising out of mathematical operations (on the equations or symbolic structure) and interpreting those in terms of actual physical situations or cases. The reason is improper handshaking between the frames representing quantitative knowledge and mathematical routines that are imported from MACSYMA (Mathlab Group 1974).

We find MECHO (Bundy et al. 1979a, b) developed in Prolog quite interesting in the domain of Mechanics problem solving. While many of the earlier programs including STUDENT and CARPS used lot of heuristics and rudimentary parsing to collect useful information from natural language problems, MECHO exploited the advances on NLP techniques; the NLP module of MECHO produced a set of predicate calculus 'assertions' about the objects (of a problem) to facilitate problem solving. It uses 'schema' and 'meta-level structure' similar to the 'frame' structure used by Kleer for representing input-supplementary yet conceptual information and inference rules. Marples (1974) search algorithm is applied strategically to extract equations from the combination of input assertions, schemas and inference rules. The philosophy underlying the operation of MECHO is the application of 'meta-information'—'meta-terms', 'meta-assertions' and 'meta-rules'(Davis and Buchanan 1977) in semantic processing, inference mechanism and knowledge representation which is principally different from object-level representation used in other natural language problem solvers. Many researchers argue that meta-level inference is superior to other strategies as it results in more transparent and modular representation of knowledge that forms the core of an expert problem solving system. It is hypothesized that human subjects solving mechanics problems employ similar model building techniques as applied in MECHO. Thus it could serve as a cognitive model as is evident from the work of Luger (1981). MECHO is especially significant as it led to the development of meta-level reasoning, a distinct subfield of AI. For example, MECHO led to PRESS (Sterling et al. 1980), a seminal piece of work in automated equation solving using meta-level inference as its basis.

ISAAC (Novak 1976), though tested on only 20 Statics problems, seem to be a more complete natural language scientific problem solver with well defined functional modules and generic structure of the internal representation model. Since the model makes explicit all of the features and relationships of the objects, it is many times larger than the original problem statement and comprehensive enough to answer any question related to a problem. It has the potential to draw force diagrams for a problem if required (in addition to its capability of drawing illustrations) and can even facilitate machine translation. The SFRAME (semantic frame) concept used by Novak for semantic interpretation of problem statements is likely to be important in any domain as far as the process of understanding is concerned. The application of canonical object frame (Minsky 1973) is unique. It can be really powerful if extended to handle dynamic attributes (e.g. velocity, acceleration as function of time) or model more complex objects in engineering domain. In addition, since such a system could handle symbolic as well as numeric calculation it could perform analysis task which most of the specialized programs cannot handle at all. ISAAC can be used as a test bed for investigating different strategies for solving a given problem. However, like all the other systems, ISAAC has its own limitations. It cannot handle new type of problems accurately without further adjustment in the program or addition of vocabularies. Secondly, the procedures used for referent identification are fairly rudimentary and deal only with extensionally specified referents. Finally, the number of equations generated per problem is much more (ten on an average) than what humans usually generate. There remains scope to apply some redundancy check and generate only the critical equations more directly.

Though ALBERT (Oberem 1987) includes a generic problem-solver its natural language understanding capability makes it different from other CAI tools that demonstrates problem solving by constraining the user to menu selection, command options or multiple-choice questions. By avoiding the keyword matching approach adopted by earlier systems ALBERT can engage students in a fairly natural problem-solving dialogue and even takes care of incomplete sentence fragments. The program keeps track of the student's qualitative and quantitative knowledge as data in the program. These data comprise the student model that is used to monitor the flow of the discussion and guide the tutorial process. The vocabulary of one-dimensional kinematics is very limited and largely free of ambiguity. But due to the modular construction of the program it has been possible to use the basic structure of its natural language system and the expert problem solver in other CAI tools like FREEBODY (Oberem et al. 1993), ILONA (Oberem et al. 1992), CICERO (Lambiris and Oberem 1993), ECLIPS (Oberem 1994) etc. ALBERT's language system has more transportability and scalability across diverse subject area, programming language and hardware platform compared to the problem solving module.

FREEBODY and ECLIPS exploit the basic architecture of ALBERT. FREEBODY like BEATRIX (Novak and Bulko 1993) can interpret diagrams but the problem of interpreting vector diagrams or free body diagrams is simpler than understanding or drawing figures that represent a complete physics problem. Though derived from ALBERT, ECLIPS has no tutorial component. The syntactic patterns of textbook molarity problems were found identical to those required for ALBERT. Also there is inherent similarity in the way information is presented in kinematics problems and molarity problems. It was mostly required to substitute the molarity equations for the kinematics equations and change some of the variable names. However, as natural language solvers ECLIPS and FREEBODY work well within a very limited domain.

The most distinctive feature of Ramani's work (Ramani 1969, 1979) is analogy based problem solving. It is a good application of language learning and use and equally good is its language processing capability. Ramani points to the fact that there is no uniform procedure for arriving at such an algorithm which maps sentences in a given language into formulas in a chosen logical calculus. So, in an attempt to explore an alternative of typical syntax-directed analysis it simulates the human tendency to solve problems based on learning from examples. It uses schema as knowledge (paradigm) representation structure and uses extrapolation algorithms instead of rudimentary pattern-operation rule for analogizing. Though the system can guarantee solutions to any problem in a class which is algorithmically solvable it can give contradictory solutions and can go into endless loops for somewhat unstructured problems.

As far as natural language mathematical proof presentation is concerned one common problem found in EXPOUND (Chester 1976), MUMBLE (McDonald 1983), THINKER (Edgar and Pelletier 1993) is that the NL proofs produced contain too many minute details, which obscures the main line of reasoning. The ground work for minimizing the redundancy or proof steps was done by Lingenfelder (1989) based on which Huang (1996) optimized the NL proof output by introducing the assertion level proof (tree-based internal representation). His presentation model PROVERB splits the presentation task into subtasks thus following the line of top-down hierarchical planning usually pursued by RST[1]-based text planners. The computational model used by Huang for informal mathematical reasoning is a blending of meta-level planning and object-level verification demonstrating strong influence of Bundy's theory (Bundy 1988; Bundy 1990) of automated reasoning.

Ω-MKRP is a strong proof development environment following the framework proposed by Bundy (1990) for the planning mechanism and using the reasoning power of an automated theorem prover. It could ideally allow the user to guide a theorem prover in its search for a proof (for more difficult problems) by feeding high level proof plans into the system preferably in natural language. But Ω-MKRP cannot produce output other than texts like equations, graphs etc. As admitted by Huang there is no provision of user model in the system which, if there, could have helped reader-dependent simplification of proofs by omitting subproofs or trivial steps. Development of appropriate user model thus remains a distinct research problem in this field of AI.

In the reverse direction (NL proof verification), in the same domain of mathematical proof Zinn (1998, 2003), Zinn et al. (2002) made one of the pioneering studies in natural language understanding in conjunction with automated reasoning. Their approach for understanding and verifying a given mathematical proof is characterized by—(i) linguistic analysis of the text focusing on typical linguistic phenomena of mathematical discourse, (ii) extraction of the proof structure and discourse relations from the proof, (iii) semantics construction in terms of proof plan recognition and instantiation by matching and attaching. Though Zinn's model holds enough potential we find very limited application of it. Similarly the DIALOG system (prototype) is experimented mostly with elementary set theory and that too supported by ΩMEGA mathematical proof assistant (Siekmann et al. 2002) and ACTIVEMATH learning environment (Melis et al. 2001a, b). The primary contribution of the project is separate modeling of static mathematical domain knowledge and the domain-independent dynamic knowledge necessary to engage in an intelligent tutoring.

The paper introducing the concept of GeometryNet actually builds a domain specific ontology for geometrical terminology, where the semantics of certain terms are finally expressed in terms of equations involving their arguments. The structure of the knowledge base has similarity to that of WordNet database (Beckwith et al. 1993; WordNet) and is fairly simple

[1] Rhetoric structure theory (Mann and Thompson 1987).

as far its construction and use is concerned. The idea of automating diagram drawing from simple geometric problem statements using the proposed Net might play an important role in machine-processing of problems that require some kind of diagrammatic representation (made with basic geometric entities) in the way to solution. One may find these types of problems in the fields of engineering graphics, electrical and electronic circuit diagrams, engineering structures, etc. As not yet implemented in full scale the actual capability of the GeometryNet is not fully demonstrated. Research is on to find most suitable language-independent intermediate representation (of problems) that could also facilitate machine translation of geometric problems stated in English besides drawing figures automatically.

Table 2 summarizes the basic features of the systems reviewed in this study.

4 Conclusion

Though we have made a sincere attempt to trace through the prominent research works in the area of machine understanding of natural language (Winograd 1972) mathematical problems it has not been possible to capture the wide spectrum of problems and research efforts even in this extremely focused area of AI. Contemporary systems which we could not include might differ in the style and low-level stages of functioning from the ones we have discussed here but we have been able to highlight some techniques that have been developed over the years for—(1) language processing and analysis of the problem text, (2) storage and retrieval of information, rules and domain knowledge, (3) machine understanding of the problem in the context of the domain and its intermediate representation, (4) implementation of solution strategies. While the earlier systems had to use tailor-made language processing routines, mostly using format matching techniques, the later systems benefited from the parallel developments in the area of NLP. We have seen how different schemes (like ‘paradigm’, ‘schema’, ‘proposition’, ‘atom’ etc.) or structures (like ‘tree’, ‘frame’, ‘meta-structure’, ‘ATN’ etc.) or database or files are used to store information, inference rules and knowledge. In many cases, adhoc and heuristic programming approach or weak search procedure is adopted that seems quite useful for typical cases but renders too narrow a scope for proper understanding and representation of problems of diverse nature. Overall, the analysis of these systems gives us a fair idea about the challenges of computer modeling of human intelligence, knowledge and reasoning ability for understanding and solving mathematical problems.

At the same time it appears to us that some of the systems (arguably, MECHO, ISAAC, GeometryNet and PROVERB) are based on scalable models and can be extended or generalized to address a wider application area. Some systems have even influenced other important areas of AI like automated reasoning (e.g. as MECHO did), knowledge representation (like ISAAC, GeometryNet, LIM-G), NLP techniques (like Zinn’s model, NLC) and also cognitive science (like MECHO, ROBUST). It must be remembered that the systems cited in this survey were not developed in isolation rather each of them were benefited by contemporary ideas of the other closely related research areas as mentioned above. For example, cognitive science theories greatly influenced the arithmetic problem solvers while GeometryNet makes full use of the already available NL parsers and graphics functions.

One important gap in research in the area of our survey is that there is hardly any benchmark test-data set or any clear evaluation strategy for most of the systems. Should we give partial credit to a system that can produce a partially correct output? If so, what would be the basis for such generalized credit system? We did not find the answer reported anywhere in the literature to the best of our knowledge. Hence there is little scope to compare the systems, even in a given domain, quantitatively.

Table 2 Summary of the systems/approaches

Solver	Author (year)	Domain	Basic feature
STUDENT	Bobrow (1964a, b)	Algebra	Keyword-based pattern matching, relational model (object-relationship-media)
DEDUCOM	Slagle (1965)	Simple arithmetic and logical problems	Pattern matching with inference rules in the form of conditional statements with variables
WORDPRO	Fletcher (1985)	One-step arithmetic problems	Set schema and rule base
ROBUST	Bakman (2007)	Multi-step arithmetic problems with extraneous information	Schema instantiation, application of change formulae to elementary propositions
LIM-G	Wong et al. (2007)	Elementary geometry (mensuration)	Knowledge base using InfoMap, frame-template structure, problem representation thru tree structure
GeometryNet	Mukherjee et al. (2007)	Geometry	Knowledge base (lexical resource) GeometryNet, standard NLP tool, language-free intermediate representation
CARPS	Charniak (1968)	Distance & volume rate problems	Tree structure, keyword-based pattern matching, knowledge base of formulae etc.
HAPPINESS	Gelb (1971a, b)	Probability	Tree structure or descriptor-list, keyword-based solution method, pattern matching, tailored solution function
Interactive Solver with NLC as prototype	Biermann et al. (1982)	Matrix	Input in typed, voice and touch mode, augmented transition network
NEWTON	Kleer (1975a, b)	Mechanics (kinematics)	Tree structure (qualitative information), frame structure (quantitative information)
MECHO	Bundy et al. (1979a, b)	Mechanics (statics & kinematics)	Input assertions, schema (house-rules), meta-level structure (information and inference rules)
ISAAC	Novak (1976)	Physics	Augmented transition network, semantic frame, canonical object frame, knowledge representation thru programs & data structure, geometric model

Table 2 continued

Solver	Author (year)	Domain	Basic feature
ALBERT	Oberem (1987)	Physics (CAI for kinematics prob.)	Lexical database, knowledge base of syntactic patterns, syntactic pattern matching
FREEBODY	Oberem et al. (1993)	Physics (CAI for FBD drawing)	NLP and problem solving scheme basically same as that of ALBERT
ECLIPS	Oberem (1994)	Chemistry (molarity problems)	NLP and problem solving scheme basically same as that of ALBERT
	Ramani (1969)	General numerical problem	Analogy-directed language processor, schema/paradigm (knowledge), extrapolation algorithms for analogizing
PROVERB	Huang (1996)	Mathematical proof	Assertion level proof as the intermediate stage for translation to NL proof from ND proof

Finally, it transpires that more systematic research is required towards development of natural language mathematical problem solvers in various domains of science and technology. The significance of the research is not only limited to establishing a new dimension of computer-aided problem solving but also lies in the possibility of enriching allied research domains like machine intelligence and machine learning, NLP, KR and also computer based teaching (CBT) and cognitive science.

References

Bakman Y (2007) Robust understanding of word problems with extraneous information. http://aps.arxiv.org/abs/math.GM/0701393. Accessed 31 Mar 2008

Ballard BW, Biermann AW (1979) Programming in natural language: NLC as prototype. In: Proceedings of the 1979 ACM national conference, pp 228–237

Beckwith R, Miller GA, Tengi R (1993) Design and implementation of the WordNet lexical database and searching software. Working Paper, Princeton University

Benzmüller C, Horacek H, Kruijff-Korbayova I, Pinkal M, Siekmann J, Wolska M (2007) Natural language dialog with a tutor system for mathematical proofs. In: Lu R, Siekmann J, Ullrich C (eds) Cognitive systems, LNAI 4429:1–14

Biermann AW, Ballard BW (1980) Towards natural language computation. Am J Comput Linguist 6(2):71–86

Biermann A, Rodman R, Ballard B, Betancourt T, Bilbro G, Deas H, Fineman L, Fink P, Gilbert K, Gregory D, Heidlage F (1982) Interactive natural language problem solving: a pragmatic approach. In: Proceedings of the first conference on applied natural language processing. Santa Monica, California, pp 180–191

Bobrow DG (1964a) Natural language input for a computer problem solving system. Report MAC-TR-1, Project MAC, MIT, Cambridge, June

Bobrow DG (1964b) Natural language input for a computer problem solving system. Ph.D. Thesis, Department of Mathematics, MIT, Cambridge

Briars DL, Larkin JH (1984) An integrated model of skill in solving elementary word problems. Cogn Instr 1:245–296

Buckley M, Dietrich D (2007a) Integrating task information into the dialogue context for natural language mathematics tutoring. In: Medlock B and Séaghdha Dó (eds) Proceedings of the 10th annual CLUK research colloquium. Cambridge

Buckley M, Dietrich D (2007b) Verification of proof steps for tutoring mathematical proofs. In: Luckin R, Koedinger KR, Greer J (eds) Proceedings of the 13th international conference on artificial intelligence in education. Los Angeles, pp 560–562

Bundy A (1988) The use of explicit plans to guide inductive proofs. In: Proceedings of 9th international conference on automated deduction, pp 111–120

Bundy A (1990) A science of reasoning: extended abstract. In: Proceedings of 10th international conference on automated deduction. Springer, pp 633–640

Bundy A, Byrd L, Luger G, Mellish C, Palmer M (1979a) Solving mechanics problems using meta-level inference. In: Proceedings of IJCAI-79. Tokyo, pp 1017–1027

Bundy A, Byrd L, Luger G, Mellish C, Milne R, Palmer M (1979b) Mecho: a program to solve mechanics problems. Working paper no. 50. Department of Artificial Intelligence, Edinburgh

Chang KE, So YT, Lin HF (2005) Computer-assisted learning for mathematical problem solving. Comput Educ (in press)

Chang K-E, Sung Y-T, Lin S-F (2006) Computer-assisted learning for mathematical problem solving. Comput Educ. 46(2): 140–151

Charniak E (1968) CARPS, a program which solves calculus word problems. Report MAC-TR-51, Project MAC, MIT, Cambridge, July

Charniak E (1969) Computer solution of calculus word problems. In: Proceedings of international joint conference on artificial intelligence. Washington, DC, pp 303–316

Chester D (1976) The translation of formal proofs into English. Artif Intell 7: 178–216

Davis R, Buchanan BG (1977) Meta-level knowledge: overview and applications. IJCAI'77, pp 920–927

De Kleer J (1975a) Multiples representation of knowledge in a mechanics problem-solver. In: Proceedings of IJCAI-77. Cambridge, Massachusetts, pp 299–304

De Kleer J (1975b) Qualitative and quantitative knowledge in classical mechanics. Artificial Intelligence Laboratory TR-'352, MIT, Cambridge

Dellarosa D (1986) A computer simulation of children's arithmetic word problem solving. Behav Res Methods Instrum Comput 18:147–154

Edgar A, Pelletier FJ (1993) Natural language explanation of natural deduction proofs. In: Proceedings of the first conference of the pacific association for computational linguistics. Centre for Systems Science, Simon Fraser University

Fellbaum C (1990) English verbs as a semantic net. Inter J Lexicogr 3(4):278–301

Fellbaum C, Gross D, Miller K (1990) Adjectives in WordNet. Inter J Lexicogr 3:265–277

Fletcher CR (1985) Understanding and solving arithmetic word problems: a computer simulation. Behav Res Methods Instrum Comput 17:565–571

Gelb JP (1971a) Experiments with a natural language problem solving system. In: Proceedings of IJCAI-71. London, England, pp 455–462

Gelb JP (1971b) The computer solution of English probability problems. Ph.D. Thesis, Computer Science Program, RPI, Troy, New York

Huang X (1996) Human oriented proof presentation: a reconstructive approach. Infix St. Augustin, Germany

Huang X, Kerber M, Kohlhase M, Melis E, Nesmith D, Richts J, Siekmann J (1992) Ω-MKRP—a proof development environment. SEKI Report SR-92-22, Fachbereich Informatik, Universitat des Saarlandes, Saarbrucken, Germany, 1992. A shorter version presented at the Workshop on automated theorem proving, IJCAI-93, Chambery, France

Hsu WL, Wu SH, Chen YS (2001) Event identification based on the information map—INFOMAP. In: Proceedings of the 2001 IEEE systems, man, and cybernetics conference. Tuscon, Arizona, pp 1661–1672

Kintsch W, Greeno JG (1985) Understanding and solving word arithmetic problems. Psychol Rev 92: 109–129

Koedinger KR, Sueker ELF (1996) PAT goes to college: evaluating a cognitive tutor for developmental mathematics. In: Proceedings of the second international conference on the learning sciences, pp 180–187

Koedinger KR, Anderson JR, Hadley WH, Mark MA (1997) Intelligent tutoring goes to school in the big city. Inter J Artif Intell Educ 8:30–43

Lambiris MAK, Oberem GE (1993) Natural language techniques in computer-assisted legal instruction: a comparison of alternative approaches. J Legal Educ 43(1):60–78

Lindsay RK (1963) Inferential memory as the basis of machines which understand natural language. Computers and thought. McCraw-Hill, New York

Lingenfelder C (1989) Structuring computer generated proofs. In: Sridharan NS (ed) Proceedings of the 11th IJCAI. Detroit, Michigan, pp 378–383

Lingenfelder C (1990) Transformation and structuring of computer generated proofs. Ph.D. Thesis, Fachbereich Informatik, UniversitatKaiserslautern, Kaiserslautern, Germany

Lingenfelder C, Pracklein A (1991) Proof transformation with built-in equality predicate. In: Proceedings of the 12th IJCAI. Sydney, pp 165–170

Littlefield J, Rieser J (1993) Semantic features of similarity and children's strategies for identifying relevant information in mathematical story problems. Cogn Instr 11(2):133–188

Looi CK, Tan BT (1998) A cognitive-apprenticeship-based environment for learning word problem solving. J Res Math Educ 17(4):339–354

Luger G (1981) Mathematical model building in the solution of mechanics problems: human protocols and the mecho trace. Cogn Sci 5(1):55–77

Mann WC, Thompson SA (1987) Rhetorical structure theory: a theory of text organization. Technical Report ISI/RS-87-190, USC Information Sciences Institute

Marples D (1974) Argument and technique in the solution of problems in mechanics and electricity. Technical Report CUED/C-Educ/TRI, Department of Engineering, Cambridge, England.

Marshall SP (1995) Schemas in problem solving. Cambridge University Press, Cambridge

Mathlab Group (1974) MACSYMA reference manual. MIT, Cambridge

McDermott D, Sussman GJ (1974) The conniver reference manual, artificial intelligence laboratory. A1M-259a. MIT, Cambridge

McDonald D (1983) Natural language generation as a computational problem: an introduction. In: Brady M, Berwick RC (eds) Computational models of discourse. MIT Press, Cambridge, pp 209–266

Melis E, Andres E, Goguadse G, Libbrecht P, Pollet M, Ullrich C (2001a) Activemath: system description. In: Proceedings of the international conference on artificial intelligence in education, http://citeseer.ist.psu.edu/article/melis01activemath.html

Melis E, Andres E, Budenbender J, Frischauf A, Goguadse G, Libbrecht P, Pollet M, Ullrich (2001b) Activemath: a generic and adaptive web-based learning environment. Int J Artif Intell Educ 12(4):385–407

Miller GA (1990) Nouns in WordNet: a lexical inheritance system. Inter J Lexicogr 3(4):245–264

Miller GA, Beckwith R, Fellbaum C, Gross D, Miller K (1990) Introduction to WordNet: an on-line lexical database. Inter J Lexicogr 3(4):235–244

Minsky M (1973) A framework for representation of knowledge artificial intelligence laboratory AIM-306. MIT, Cambridge

Mukherjee A, Garain U, Nasipuri M (2007) On construction of a GeometryNet. In: Proceedings of IASTED international conference on artificial intelligence and applications (AIA 2007). Innsbruck, Austria, pp 530–536

Mukherjee A, Garain U (2009) Understanding of natural language text for diagram drawing. In: Proceedings of 13th international conference on artificial intelligence and soft computing

Muth D (1992) Extraneous information and extra steps in arithmetic word problems. Contemp Educ Psychol 17:278–285

Newell A, Shaw JC, Simon HA (1959) Report on a general problem-solving program. In: Proceedings of the international conference on information processing, pp 256–264

Novak GS (1976) Computer understanding of physics problems stated in natural language. The University of Texas at Austin, Ph.D. Thesis, arch 61

Novak GS, Bulko WC (1993) Diagrams and text as computer input. J Vis Lang Comput 4(2):161–175

Oberem GE (1987) ALBERT: a physics problem solving monitor and coach. In: Proceedings of the first international conference on computer assisted learning (ICCAL'87). Calgary Alberta, Canada, pp 179–184

Oberem GE (1994) Transfer of a natural language system for problem-solving in physics to other domains. In: Proceedings of ED-MEDIA 94—World conference on educational multimedia and hypermedia. Vancouver, British Columbia, pp 424–431

Oberem GE, Mayer O, Makedon F (1992) ILONA: an advanced CAI tutorial system for the fundamentals of logic. Educ Comput 8(3):245–254

Oberem GE, Shaffer PS, McDermott LC (1993) Using a computer to investigate and address student difficulties in drawing free-body diagrams. A paper presented at the summer meeting of the American association of physics teachers, Boise, ID

Parsaye K, Chignell M (1988) Expert systems for experts. Wiley, London

Pinkal M, Siekmann J, Benzmüller C, Kruijff-Korbayova I (2008) DIALOG: natural language-based interaction with a mathematics assistance system. Project report in the Collaborative Research Centre SFB 378 on resource-adaptive cognitive processes

Ramani S (1969) Language based problem-solving. Ph.D. Thesis, Computer Group, Tata Institute of Fundamental Research

Ramani S (1979) A language based problem solver. In: Proceedings of IJCAI-71. London, pp 463–473

Siekmann J et al (2002) Proof development with Ω MEGA. In: Proceedings of the 18th conference on automated deduction, LNAI 2392. Springer, Copenhagen

Simmons RF (1970) Natural language question-answering systems: 1969. Commun ACM 13(1):15–30

Simmons RF (1973) Semantic networks: their computation and use for understanding english sentences. In: Schank RC, Colby KM (eds) Computer models of thought and language. Freeman and Co., pp 63–113

Slagle JR (1965) Experiments with a deductive question-answering program. J-CACM 8(12):792–798

Steele M, Steele J (1999) DISCOVER: an intelligent tutoring system for teaching students with learning difficulties to solve word problems. J Comput Math Sci Teach 18(4):351–359

Sterling L, Bundy A, Byrd L, O'Keefe RA, Silver B (1982) Solving symbolic equations with PRESS. In: Proceedings of the European conference on computer algebra, LNCS, pp 109–116

Wheeler JL, Regian JW (1999) The use of a cognitive tutoring system in the improvement of the abstract reasoning component of word problem solving. Comput Human Behav 15:243–254

Winograd T (1972) Understanding natural language. Academic Press, New York

Wong WK, Hsu Sheng-Cheng, Wu Shih-Hung, Lee Cheng-Wei, Hsu Wen-Lian (2007) LIM-G: learner-initiating instruction model based on cognitive knowledge for geometry word problem comprehension. Comput Educ (Elsevier) 48((4):582–601

Woods WA (1970) Transition network grammars for natural language analysis. Commun ACM 13(10): 591–606

WordNet: A lexical database for the english language. Cognitive Science Laboratory, Princeton University, Princeton http://wordnet.princeton.edu/

Zinn C (1998) Verifying textbook proofs. Technical Report, Technical University of Vienna

Zinn C (2003) A computational framework for understanding mathematical discourse. Log J IGPL 11(4): 457–484

Zinn C, Moore JD, Core MG (2002) A 3-tier planning architecture for managing tutorial dialogue. In: Proceedings of intelligent tutoring systems, 6th international conference, vol 2363 of LNCS. Springer, Biarritz, pp 574–584