LEARNING ARITHMETIC PROBLEM SOLVER

Masamichi SHIMURA and Seiichiro SAKURAI
Tokyo Institute of Technology
Department of Computer Science
Ohokayama, Meguro, Tokyo

ABSTRACT

In this paper we describe a problem solving system with a learning mechanism (Learning Arithmetic Problem Solver, LAPS), which can solve arithmetic problems written in natural languages. Since LAPS has knowledge about arithmetic problems in the form of rules, it can solve many different problems without alteration of the program. When LAPS cannot solve a given problem because of a shortage of knowledge, it asks the user how to solve the problem. According to the user's advice LAPS acquires knowledge and rules. Using these rules, LAPS can solve problems. Furthermore, LAPS can improve its performance at problem solving by synthesizing rules that are applied.

I INTRODUCTION

Recently many researchers in A.I. or K.E. have developed several practical knowledge based systems. However, such systems are restricted to rather narrow fields. In general-use systems the knowledge required is excessive and knowledge acquisition is a bottleneck. This paper presents a knowledge acquisition method in problem solving systems. For problem solving, the system needs knowledge to understand the problem and to derive equations. Our LAPS can solve algebraic problems given in natural language. When knowledge is lacking, LAPS can acquire some knowledge to solve a given problem through interaction with a user or teacher. The knowledge obtained from him is generalized and stored in the system. Through the process of problem solving LAPS can get problem-solving knowledge by synthesizing rules that are applied. Once LAPS succeeds in solving the problem, it can solve a similar problem without backtracking. In other words, LAPS can improve its performance at problem solving by learning.

Early attempts at solving algebraic problems given in natural language are the programs by Bobrow and Charniak[1]. However the elementary parsing technique and simple semantic structures used by Bobrow and Charniak are inadequate for any but the easiest problems. Bundy[2]'s MECHO solves a wide range of mechanics problems given in English. MECHO uses meta-level inference, which provides powerful techniques for controlling the use of knowledge. MECHO, however, must be provided with full knowledge in order to solve problems. LAPS has a learning module and can fill in knowledge deficiencies. Davis[3]'s TEIRESIAS can obtain knowledge through interaction with users, but TEIRESIAS's purpose is as a knowledge acquisition system for an expert system rather than a learning system. Our intention has been to build a system which can understand a natural language, solve problems, and learn through the interaction with a teacher or by problem solving.

II LAPS PROGRAM

As shown in Figure 1 LAPS consists of a natural language processor, problem solver, rule generator, rule modifier and knowledge base.

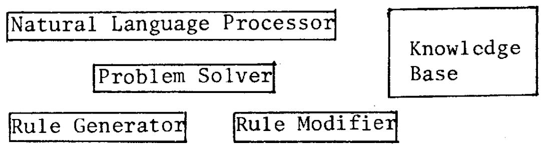

Figure 1. The structure of LAPS

In our system, the input statements describing a given problem are translated into an accessible and modifiable structure for the system by the natural language processor. The natural language processor consists of a syntactic parser and a semantic analyzer. As the syntactic analyzer, the extended LINGOL[4] is used. LINGOL generates multiple parsed trees from an input statement when there is syntactic ambiguity. Each parsed tree is not only the structured data but also a program for the semantic analyzer. After selecting the most plausible one from the multiple trees, LAPS invokes the semantic processing routine in order to produce the appropriate structure. This structure is called a "fact-graph" which is a kind of semantic network. In the fact-graph, nodes correspond to objects represented by subjective and objective words in the problem statement, and links correspond to objects' property represented by their modifiers. The process of semantic analysis proceeds by the generation of nodes and the connection of two nodes with a link. Thus a data base about the given problem is constructed in the system.

Figure 2 shows an example of the hierarchy of concepts in the knowledge base represented by the connection of nodes with links. Such hierarchical knowledge is used for the process of generalizing acquired knowledge.

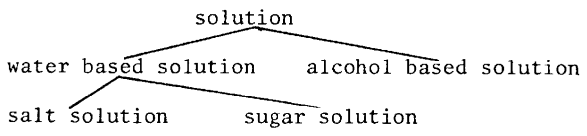

Figure 2. Example of the hierarchy of concepts

For solving arithmetic problems, the system derives equations using the data base constructed from the problem description. In our system, some knowledge rules are used in deriving the appropriate equations for the given problem. That is, using heuristics, the system chooses the most likely equation among the candidates. Then the derived equation is rewritten by symbol manipulation so that variables are moved to the left hand side of the equation. A repetitive substitution of value into variables is made until no variables appear in the representation. In the above manipulation, the algebraic formula for solving a given problem is stored in a tree structure. Then the problem solver works toward extraction of simultaneous equations, and solves them by using an equation extraction rule.

The equation extraction rule consists of a lambda part, a target part, a condition part, and an action part. The lambda part is a list of variables, and the target part contains a condition about the target variable. When LAPS solves a problem, it determines a target variable for the derivation of an equation and then generates an applicable rule list by checking the target part of each rule. By matching the condition part with the fact-graph, the correspondence of the variables in the condition part with the nodes and links in the fact-graph is performed. After replacing the variables in the action part according to this correspondence, an equation is extracted by the evaluation of the action part. Since the value of any variable in the action part does not need to be known, the rules are used in a variety of cases. The condition part is represented in the form of a Lisp function or the form defined by the user. Operators, "and", "or", and "not" are used for connection of the conditions. An example of a rule for extracting equations is shown as follows. "Condition-add", "action-equation", "action-term" and "action-expression" are names of forms defined by the user. The following rule in Figure 3 is paraphrased as:

If *z is a sum of *x and *y then extract the equation *z=*x+*y

Since LAPS has knowledge about extracting equations in the form of such rules, LAPS's knowledge of solving problems can be extended easily by adding rules.

```
(rule-1 eq-rule
   (lambda = (*target *property))
   (target = (member *property
                (cardnum weight price ...)))
   (condition = (condition-add *x *y *z))
   (action = (action-equation
                (action-term *z *property)
                =
                (action-expression
                 (action-term *y *property)
                 +
                 (action-term *x *property)))))
```

Figure 3. Example of a rule

Figure 4 shows the LAPS's main routine. When no rules match the given problem, LAPS asks a teacher how to solve it. LAPS analyzes the teacher's answer given in natural language and stores it in the form of a generalized rule. This generalized rule is produced by the rule generator module. The obtained rule, however, is not always correct because of the generalization process. If LAPS finds that the obtained rule is inappropriate for the given problem, the rule is revised according to the teacher's supervision. For the modification of the rule, LAPS invokes the rule modifier module and interacts with the teacher.

```
repeat
  get a problem from a teacher
  repeat
    if a rule matches the given problem then
      apply it (no learning)
    else get an instruction from a teacher
         and generate a rule
  until the problem is solved
until the teacher satisfies
```

Figure 4. LAPS's main routine

III GENERATING A RULE

A. Generating A Rule From A Teacher

When LAPS cannot solve a problem because of a shortage of knowledge, LAPS resolves this lack of knowledge in order to solve the problem. The teacher's advice given in natural language is transformed to the internal form. Consider the following advice.

The weight of a solution of salt equals the total number of the water and the salt.

The follwing equation can be obtained from the above statement, where "solution", "water" and "salt" are variables which correspond to the objects appearing in the problem statements.

weight(solution) = weight(water)+weight(salt)

This rule can be obtained by the simple transformation of an input statement. It is, however, a very specific equation and can be used only for problems about salt solutions. For example, the above rule cannot be used for problems about sugar solutions. Even if we extend the LAPS's inference mechanism so that LAPS can infer by using the hierarchy of concepts, its execution time may be excessive unless the user controls the inference mechanism. The generation of such specific rules may cause the explosion of the rule base. Our system generates more generalized rules which are applicable to a wider variety of problems.

Generalization is done by 1) dropping conditions, 2) replacing constants with variables, and 3) abstracting the concepts by moving up the hierarchy of concepts in the knowledge base. Disjunctive generalization is made by adding additional conditions or replacing conjunction by disjunction.

For the generalization of a rule, the standard generalization technique of Mitchell[5][6] has been used. However his candidate elimination algorithm is not very efficient because it is data driven. For the efficiency of generalization, LAPS restricts the initial description of the hypothesis so that the search space is comparatively small. For example, if all constants appearing in the generated rule are replaced by variables, LAPS could not derive an adequate equation because correspondence of variables with the nodes and links in the fact-graph cannot be obtained. To extract an adequate equation by applying learned rules, all the variables in the action part must be included in the condition part. Once a new variable is introduced into the action part by applying the constants replacing rule, the condition part must be fulfilled for generating new variables. Also, the condition part, even if some condition is dropped, must still include all the variables in the action part. If infinite disjunctive generalization is permitted, the space of the hypotheses becomes infinite. Disjunctive generalization is made, therefore, only when LAPS determines the initial description and there exist alternative hypotheses. To determine the initial description, LAPS generalizes the input data so that there exist no more general descriptions which include all the variables in the action part.

1. Action Part

In order to translate the advice given in natural language, the equation extraction rules are utilized. The input statement is translated into a fact-graph, and then equations are extracted from the fact-graph. As a result of solving the simultaneous equations, an equation, which includes only one variable, is obtained. Finally, the system can get an executable form by using a pattern matcher, since the obtained equation is structured data. In this way, LAPS can obtain the action part from a teacher's advice.

2. Condition Part

It is not easy for a teacher to input a complete form of the conditions. Therefore, LAPS generates the condition part by translating the original problem statement into the internal format. Since the simply translated form is about a specific problem, the translated form should be generalized. In the generalization process redundant or noisy parts of the original problem must be ignored and over-generalization should be avoided. In order to modify the translated form without backtracking, two conditions are employed in association with the rules generated from the advice. These two conditions are the maximally general condition and the maximally specific condition, which hold information about the problem states where application of the rule is appropriate or inappropriate.

In applying rules, LAPS checks whether the given problem satisfies both these conditions or not. If only the general condition is satisfied then LAPS identifies whether the application of the rule is appropriate or not. To identify applicability of the rule, LAPS generates an English statement from the rule and interacts with the outside teacher. If the application of the rule is appropriate, LAPS generalizes the maximally specific condition so that the current problem satisfies the new condition. If the application of the rule is inappropriate, LAPS specializes the maximally general condition so that the current state does not satisfy the new condition. After proceeding with the process of the modification of a rule, LAPS acquires a new complete rule, if the maximally general condition is equal to the maximally specific condition. LAPS can solve problems by using the incomplete rule during rule learning.

B. Generating A Rule From The Execution Process

As described above, LAPS combines equation extraction rules so as to solve problems. When some rules are applicable to the current target variable, LAPS can select a better rule by using heuristic information to resolve rule conflict. When an inadequate rule is selected, LAPS backtracks so as to obtain better rules.

1. Action Part

In order to improve performance, LAPS generates a new rule by synthesizing rules which were applied during problem solving. As LAPS constructs a tree structure which contains the information for combining equations, LAPS can synthesize the action parts of the applied rules by traversing the tree and using the information stored in the structure. LAPS composes equations by symbol manipulation and then translates the composed equation into executable form. This process proceeds almost identically to the process of making an action part from the teacher's statement. The difference between the two processes is that by solving a problem the composition process requires generalization of the equation. In other words, the tree structure generated is for a specific problem, and the rule generated by using only such specific information is a specific one. To get more generalized rules, new variables are introduced into the action part by replacing constants so that the generated rule can be used in similar types of problems. Unless the range of the value of the introduced variables is restricted, however, the application of a generated rule results in extracting an incorrect equation. Consider the following equation where X is a variable.

speed(X) = distance(X) / time(X)

Since the above equation is derived by rules, the system can recognize that the following equation is incorrect.

distance(X) = speed(X) / time(X)

Equation extraction rules represents not only equations for solving problems but also constraints on the equations. In LAPS, the range of the variables is restricted by using the equation extraction rules. When a rule is extracted in the execution process, the constraints of the introduced variables are composed into the condition part of the rule.

2. Condition Part

In order to determine the necessary conditions for extraction of the synthesized equation, LAPS connects the condition parts of all the applied rules with conjunctions and unifies the variables, that is the variables representing the same object are replaced by a unique variable. As described before, LAPS extracts the constraints of the variables introduced for generalization of the action part of the rule from each rule in the system, and appends the extracted constraints to the generated condition part. Such constraints are placed in the condition part of each equation extraction rule. These constraints restrict only the range of the introduced variables, and do not always guarantee the correctness of the generalization. The rules generated by the problem solving process, therefore, require modification just as the rules generated from a teacher's advice do. Rules generated by problem solving are revised during solving other applicable problems.

C. Comparative Review

An equation extraction rule can be considered as a model of a problem, since its condition part represents the problem statements and its action part represents the problem solving procedures. In order to generate a new equation extraction rule, the rules initially given in the system are used as domain dependent knowledge. Hence our method is applicable for knowledge acquisition by altering the rules.

Since the process of generating a rule from a teacher is based on a data driven method, it requires many examples to complete a new rule. On the other hand, the process of generalizing a rule is a model driven method guided by the equation extraction rules and does not require as many examples. If appropriate models are given, knowledge acquisition in LAPS is realized in comparatively short time.

IV IMPLEMENTATION

LAPS is written in UTILISP[9] on MC68000 (12.5MHz), and the program contains about 7000 lines. There are about 20 grammar rules and almost 400 words in the dictionary. There are 10 equation extraction rules, but LAPS can solve many problems by combining these rules. When LAPS lacks knowledge to solve a given problem, LAPS asks the user how to solve the problem. Figure 5 shows the example of a dialogue when LAPS queries. The statement preceded by "->" are the user's input.

```
-> There is water with a weight of 95 grams.
Please continue.
-> We dissolve salt with a weight of 5 grams in
the water.
Please continue.
-> What is the concentration of the solution?


Excuse me, please teach me how to solve the
problem.
-> The concentration of a solution is the weight
of the salt divided by the weight of the solution
times 100.
Thank you very much.
I'll try to solve the problem.


The answer is 5.
```

Figure 5. Example of a dialogue when LAPS lacks knowledge.

In Figure 5 the first attempts to solve the given problem results in a failure because of a shortage of knowledge. Then LAPS asks the user how to solve the problem, solves it by using the acquired knowledge, and acquires a new rule. However, since the acquired rule is not guaranteed correct, LAPS uses the newly acquired rule while checking its applicability. If only one condition of the learned rule is satisfied, LAPS asks the user whether the rule is applicable or not. According to the user's advice, LAPS can then solve the second problem while modifying the rule.

To improve the performance of LAPS, LAPS generates a new rule by synthesizing rules applied during solving problems. Figure 6 shows the example of a dialogue with LAPS. The cardnum in Figure 6 represents the function which returns the number of elements of the given objects.

```
-> The total number of cranes and turtles is 20.
Please continue.
-> The total number of legs of cranes and turtles
is 60.
Please continue.
-> How many cranes are there?
As the number of crane's legs is 2,
cardnum(crane-1) = cardnum(leg-of-crane-1) * 1/2
As the total number of legs of cranes and turtles
is 60,
cardnum(leg-of-crane-1) =
      (60 - cardnum(leg-of-turtle-1) )
As the number of a turtle's legs is 4,
cardnum(leg-of-turtle-1) = cardnum(turtle-1) * 4
As the total number of cranes and turtles is 20,
cardnum(turtle-1) = (20 - cardnum(crane-1) )
Consequently,
    cardnum(crane-1) = 10
The answer is 10.
```

Figure 6. Example of a dialogue

After solving the problem shown in Figure 6, LAPS generates a new rule which may produce the following equation.

```
cardnum(crane-1) =
   (cardnum(animal-1) * cardnum(leg-of-turtle)
    -
    cardnum(leg-of-animal-1))
   /
   (cardnum(leg-of-turtle)
    -
    cardnum(leg-of-crane))
```

Figure 7. Example of a composed equation

In Figure 7 cardnum(leg-of-turtle) and cardnum(leg-of-crane) represent the numbers of turtle's legs and crane's legs, respectively. Cardnum(animal-1) represents the total number of cranes and turtles, and cardnum(leg-of-animal-1) represents the total number of legs of cranes and turtles. LAPS generalizes the above equation so that LAPS can solve a similar problem by applying the newly generated rule. In the above equation, the constants are replaced by the different variables, and the conditions which include the newly introduced variables are appended to the condition part of the rule. Then the newly synthesized rule will be tested in the future problem solving. To identify the applicability of the new rule, LAPS interacts with the teacher. If the generalization by replacing constants is incorrect, specialization of the rule by replacing variables to constants will be done.

V CONCLUSIONS

In this paper, we presented a problem solving system which employs learning, problem solving and natural language processing together. With the aid of a teacher, our system can acquire new knowledge and utilize it. Learning from examples creates equation extraction rules that can be used in the problem solver. However, LAPS cannot acquire disjunctive concepts because of using Mitchell's candidate elimination algorithm. And if the maximally specific condition is overly generalized, the rule learning results in a failure because LAPS cannot restore the information that were discarded.

REFERENCES

[1]Charniak, E. "Compute solution of calculus word problems", In Proc. IJCAI-69, Washington D.C., 1969, pp. 303-316

[2]Bundy, A., Byrd, L., Luger, G., Mellish, C. and Palmer, M., "Solving Mechanics Problems Using Meta-level Inference", In Proc. IJCAI-79. Tokyo, Japan, August, 1979, pp. 1017-1027.

[3]Davis, R. and Buchanan, B. G. "Meta-level knowledge: overview and applications", In Proc. IJCAI-77. Cambridge, USA, August, 1977, pp. 920-927.

[4]Unemi, T. Master thesis, Tokyo Institute of Technology, Tokyo, Japan, March, 1980.

[5]Mitchell, T. M. "Version spaces: a candidate elimination approach to rule learning", In Proc. IJCAI-77, Cambridge, USA, August, 1977, pp. 305-310.

[6]Mitchell, T. M. "Generalization as Search" Artificial Intelligence 18:2 (1982), 205-226

[7]Neves, D. M. "Learning procedures from examples and by doing" In Proc. IJCAI-85. Los Angels, USA, August, 1985, pp. 624-630

[8]Michalski, R. S. "A Theory And Methodology Of Inductive Learning", Machine Learning, tioga, 1983, pp. 83-134.

[9]Chikayama, T. UTILISP Manual, METR 81-6, Department of mathematical engineering and instrumentation physics, University of Tokyo, 1981.