

KNOWLEDGE-BASED PROBLEM SOLVING BY A LABELLED PRODUCTION SYSTEM

Yuichiro Anzai, Naoto Ishibashi*, Yuichi Mitsuya and Shoji Ura

Department of Administration Engineering
Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223
Japan

The paper summarizes a system that understands, solves, and learns how to solve geometry problems. The system is represented homogeneously as production rules of Labelled Production System. A brief introduction to the system architecture is given, followed by an example of how the system works. Then the paper discusses briefly some theoretical problems focused upon in design and implementation of the system.

1. INTRODUCTION

Recent development of cognitive science has revealed that problem solving is a total activity of understanding, solving, and learning how to solve problems, controlled on a unified knowledge base. This paper summarizes an effort toward building a system that encompasses such a broad range of knowledge-based problem solving, taking elementary geometry as a task domain.

Our fundamental assumption is that knowledge acquisition exerts a considerable influence on problem-solving behavior. This leads to our emphasis on post-solution analysis for acquisition of new knowledge from the solution process. It also leads to a homogeneous representation of the system as production rules of Labelled Production System (LPS): knowledge acquisition on a well-structured knowledge base of geometry requires a representation which has capability to represent structured knowledge and high learnability at the same time. Also we believe that all information necessary in solving a given problem is not always available when the problem was understood. Some information might be necessarily inferred through subgoals, or popped up by demon-like procedures during the solution process. So interaction of subgoal-oriented and pattern-directed processing is one of our concerns in designing a knowledge-based problem solver.

The above assumptions and thus motivations make our system different from traditional geometry theorem provers (e.g., [6]). Our main concern is to theorize problem solving from a broader viewpoint. The current version of LPS is written in INTERLISP, and running on DEC-20. The version

is only for an experimental use, but all the features described here are already implemented in it.

2. SYSTEM ARCHITECTURE

The system comprises four components: the Natural Language Understander, Problem Solver, Post-solution Analyzer, and factual knowledge about geometry. Understander transforms a (Japanese) problem sentence to a set of semantic primitives such as geometric objects and relations, and goals for the proof. Solver works on this set toward solving the problem. It is able to try drawing additional line segments by using a partial-match algorithm and knowledge acquired through solving similar problems. Analyzer undertakes Solver's output, and tries to create possibly helpful productions. First Analyzer generates a hypothesis of a proposition. If the proposition was tested successfully by Solver, Analyzer adds it to the knowledge base. Interaction of subgoal-oriented and pattern-directed processes is realized by ACTIVATE as seen later.

The above process proceeds by utilizing factual knowledge about geometry. All the four components of the system are represented homogeneously as production rules of LPS. LPS is, in short, a production system in which knowledge elements are *labelled*. Although global control of LPS is basically similar to 'recognize-act cycle' type production systems [4], its labelled structure greatly facilitates dealing with structured knowledge and reorganization of it. For example, labels such as CATEGORY and A-PART-OF can be used for inference, similarly to slot-names in frame systems [2]. Also we can use labels for referring to a knowledge element in another production directly without using working memory (WM). See [1] for the details of LPS.

* Presently at SONY Corporation.

3. KNOWLEDGE-BASED PROBLEM SOLVING: AN EXAMPLE

An example outlines the system's behavior most explicitly. Fig.1 shows a simple geometry problem. First of all, Understander transforms the sentence to a set of semantic primitives, which are shown in Fig.2.

Problem: Senbunno chuutenwo toori, koreni suichokuna chokusenjouno tenwa senbunno ryoutankara toukyorini aru. (A point on the line through the midpoint of a line segment and perpendicular to it is equally distant from the endpoints of the line segment.)

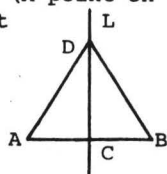


Fig. 1 Example problem.

```
((PGOAL(LEQUAL(LS D A) (LS D B))) (LS D A) (LS D B)
(EP A (LS A B)) (EP B (LS A B)) (PERP (LS A B)
(LS C D)) (ON (LS C D) (LINE L)) (LS C D) (ON D
(LINE L)) (PT D) (PERP (LS A B) (LINE L)) (ON C
(LINE L)) (LINE L) (ON C (LS A B)) (LEQUAL(LS A C)
(LS C B)) (LS A C) (LS C B) (MP C (LS A B)) (PT C)
(PT A) (PT B) (LS A B) (READNIL) (READ-START))
```

Fig. 2 Understander's output for the example problem. (For simplicity, notation is slightly different from the original output.)

Evoked by the goal (PGOAL(LEQUAL(LS D A) (LS D B))), meaning to prove equality of some quantity (length here) of the same type of objects (line segments here), Solver tries to ACTIVATE the pattern (CONGRUENCE) to WM. ACTIVATE collects productions related to (CONGRUENCE) by chaining productions backwards, and executes production-firing only on the collected subset of productions. One of collected productions, TRIANGLE, pops up two new triangles, (TRIANGLE D A C) and (TRIANGLE D B C). Then these triangles make the production CONGRUENCE executed, and (CONGRUENCE) deposited into WM. Note that the two triangles did not exist in WM when Solver started to work (Fig.2). They were necessarily recognized in the ACTIVATE function mode. Solver needs to recognize some more relations to attain the goal. It is the end of the problem-solving process when the geometric relation, (LEQUAL(LS D A) (LS D B)), PGOAL's argument, is placed in WM.

The current version of Analyzer intends to construct new propositions from Solver's output by means of three different ways. First, note that Solver concludes not only the conclusion for the

problem, but also other two relations generated by a side-effect: (1) line segments DA and DB have equal length, (2) angles $\angle ADC$ and $\angle CDB$ have equal degrees, and (3) angles $\angle DAC$ and $\angle DBC$ have equal degrees. So Analyzer builds three separate productions, 'if conditions of the problem hold, then (k) is true,' for $k=1,2$ and 3.

Productions generated in this manner can deal only with problems that have essentially the same conditions as the original problem. Analyzer's second way of knowledge acquisition is to build a production: for a given problem X, 'if most of geometric objects, relations and goals generated by Solver for solving the example problem matched WM for the problem X, then try to apply to solve X geometric concepts and propositions used in solving the example problem.' 'Most of' implies that, if this production can be executed, X is similar to the example problem if not totally the same. Hence conditions of this production should match partially.

Analyzer's last way of restructuring Solver's output is to use task-dependent heuristics to extract some small amount of mutually related objects and relations, and construct a 'speculative' hypothesis. Fig.3 shows an example of such hypothesis. It needs to be tested by Solver, which goes successfully in this case, and the production in Fig.3 is registered as a new production. Testing the hypothesis by Solver, in this case, happens to involve drawing an additional line segment. Analyzer searches for a production already acquired and similar to the hypothesis, and here draws the segment which connects D with C, the midpoint of AB, in Fig.1.

Analyzer builds up several productions consecutively in the above manners. It is the end of the system's problem-solving process: after Analyzer worked out, the system waits for another problem, which is to be fed first to the understander.

```
(G0900
#A0901 (PGOAL(LEQUAL(ANGLE $A0906 $A0909 $A0907)
(ANGLE $A0906 $A0907 $A0909)))
#A0902 (LS $A0906 $A0907)
#A0903 (LS $A0906 $A0909)
#A0904 (LS $A0909 $A0907)
#A0905 (LEQUAL(LS $A0906 $A0909)
(LS $A0906 $A0907))
ACT (RESULT(LEQUAL(ANGLE $A0906 $A0909 $A0907)
(ANGLE $A0906 $A0907 $A0909)))
(LEQUAL(ANGLE $A0906 $A0909 $A0907)
(ANGLE $A0906 $A0907 $A0909))
CATEGORY (TRIAL))
```

Fig. 3 A hypothesis production created by Analyzer.

4. ISSUES IN KNOWLEDGE-BASED PROBLEM SOLVING

4.1 Partial-match and drawing additional line segments

Some work on geometry problem solving is concerned with drawing additional line segments [3]. It is a tempting task for AI because it is a good example of generating new appropriate representations.

'Drawing additional line segments,' in our case, is a process of finding a once-solved similar problem, and generating a set of new geometric objects and relations that reduces the difference of the once-solved and currently-attacked problems. Selecting a similar problem corresponds to deciding a best-match between the current WM and condition sides of some learned productions. We have tried to avoid the curse of computational explosion inherent in partial-match by introducing a few heuristics: only geometric objects and relations are taken into account, and an object which is A-PART-OF another object is neglected. This kind of semantic categorization and inference uses the labelled structure of LPS, and saves a fair amount of computation time.

4.2 Interaction of subgoal-oriented and pattern-directed processing

Psychological research [5] tells us that elementary geometry problem solving involves organization of subgoals. This basic problem-solving structure is represented in our system by the function ACTIVATE. ACTIVATE tries to achieve a specified subgoal by organizing related productions as exemplified in the last section.

ACTIVATE is a powerful function for our purpose because knowledge of geometry is well-structured enough to activate reasonable subsets of productions. However, popping up objects or relations in a pattern-directed manner also plays a key role in solving problems, as it might generate useful but formerly unrecognized objects or relations. But since bottom-up processing may easily blow up computationally, efficient embedding of some interactive structure of top-down and bottom-up processes is highly desirable in any knowledge-based problem solver. Using ACTIVATE actions within pattern-evoked productions is a realization of this general framework.

4.3 Structuring new productions from solution experience

To build a hypothesis production as shown in Fig.3, Analyzer must select suitable elements in WM for creating conditions and actions. The

mechanism for gathering them is analogous to the heuristics incorporated in the algorithm for drawing auxiliary line segments, but slightly more complex. Analyzer picks up one geometric relation arbitrarily, and makes it as the action in the hypothesis. Then Analyzer collects objects and relations related (in a certain sense) to the action, and assumes them as conditions. The mechanism makes use of LPS's labelled structure.

Note that the example in Fig.3 is equivalent to: the two base angles of an isosceles triangle is congruent. The system succeeded in generating the production in Fig.3 *without* solving this problem. We believe that the effort of Analyzer deserves doing, as this kind of self-organizing performance is the heart of intelligent tasks.

5. CONCLUSION

We have chosen geometry as the task domain mainly because it satisfies our general motivations stated in Section 1, and involves issues described in Section 4. We believe that those motivations and issues are general and central in any knowledge-based problem solving task. LPS is a general-purpose representational system, and our research presented here is a working study toward a general theory of knowledge-based problem solving. Though some detailed part of the used techniques are task-specific, and we have yet little experience on LPS in other domains, our work has achieved a step toward a computational unification of understanding, solving and learning within a general framework of problem solving.

REFERENCES

- [1] Anzai, Y., Ishibashi, N., Mitsuya, Y. & Ura, S. "Knowledge-based Problem Solving by a Labelled Production System." Keio Engineering Report, Faculty of Engineering, Keio Univ., to appear.
- [2] Bobrow, D.G., Winograd, T. et al. "Experience with KRL-0: One Cycle of a Knowledge Representation Language." In *Proc. IJCAI-77*. MIT, Cambridge, Mass., August, 1977, pp.213-233.
- [3] Elcock, E.W. "Representation of Knowledge in a Geometry Machine." In E.W.Elcock & D.Michie (eds.), *Machine Intelligence 8*, 11-29, Ellis Horwood, 1977.
- [4] Forgy, C. & McDermott J. "The OPS2 Reference Manual." Dept. of Computer Science, Carnegie-Mellon Univ., 1978.
- [5] Greeno, J.G. "Processes of Understanding in Problem Solving." In N.J.Castellan et al.(eds.), *Cognitive Theory 2*, 43-83, Lawrence Erlbaum, 1977.
- [6] Nevins, A.J. "Plane Geometry Theorem Proving Using Forward Chaining." *Artif.Intell.* 6 (1975) 1-23.