Knowledge Acquisition (1989) **1**, 113–137

Acquisition of associative knowledge by the frustration-based learning method in an auxiliary-line problem

MASAKI SUWA† AND HIROSHI MOTODA‡

† Department of Nuclear Engineering, University of Tokyo, Hongo, Tokyo 113, Japan

‡ Advanced Research Laboratory, Hitachi, Ltd, Kokubunji, Tokyo 185, Japan

(Based on a paper presented at the 3rd AAAI Knowledge Aquisition for Knowledge-Based Systems Workshop, Banff, November, 1988)

We have developed a learner, AUXIL, which has the ability to solve auxiliary-line problems in geometry in an intelligent way. First, we show that a basic mechanism for producing auxiliary-lines is to associate a certain condition or subgoal in the problem with an appropriate *figure-pattern* and that AUXIL can produce a successful auxiliary-line by making use of associative strategies, which we call *figure-pattern strategies*. Secondly, we proposed a new method, *frustration-based learning*, which can acquire associative strategies through experiences of solving a variety of auxiliary-line problems. AUXIL simulates the following expert behavior. When an expert tries to solve such a problem, he feels frustrated because enough information is not given in a problem space for him to proceed an inference and to find a correct path from given conditions to the goal. Here, he concentrates himself on the conditions or subgoals which have caused frustration. After he has produced an auxiliary-line and made a complete proof-tree, he would learn several associative strategies. Each frustration-causing condition or subgoal will constitute the if-part of each strategy. He will then recognize several lumps of *figure-patterns* in the proof-tree, each of which has contributed to resolving each frustration. All pieces of geometrical information of each figure-pattern will constitute the then-part of each strategy. Learning an auxiliary-line problem through frustration-based learning means to understand it as a composition of figure-patterns each of which has those features represented in the THEN-part of the corresponding strategy. The frustration-based learning method is regarded as a method for learning some essential figure-patterns which underlie and structurize a problem solving process of elementary geometry.

1. Introduction

Auxiliary-line problems are ones which cannot be solved without making additional lines or points to the properties given in the problem. In this paper, we restrict our interest to a class of auxiliary-line problem in which only one new point is required. Three typical examples are shown in Fig. 1. The general feature distinguishing each of these from other non-auxiliary line problems in elementary geometry is that there lies a discontinuous gap between the goal and given conditions and that there exist some problem elements to which any geometrical basic theorem cannot be applied on the *both* sides of this gap. Producing appropriate auxiliary-lines corresponds to

1042-8143/89/010113 + 35$03.00/0

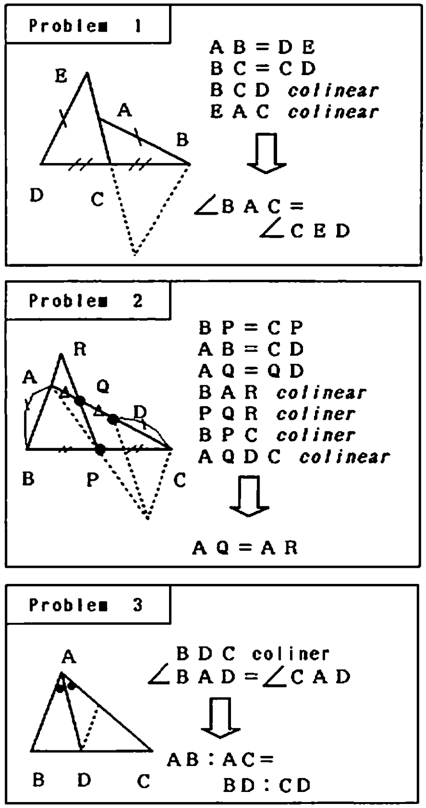


FIG. 1. Typical auxiliary-line problems.

throwing a bridge over the discontinuous gap to find the path to the goal by successive applications of theorems.

This work has been motivated by a naive question; why can human beings select only a few appropriate candidates out of almost innumerable sets of auxiliary-lines? In order to find a plausible answer to this question, we try to make the following hypothesis about what kind of thinking processes are occurring in our mind to solve auxiliary-line problems. If a man, who has just acquired some strategies of producing auxiliary-lines in solving the first problem in Fig. 1 (problem 1), encounters the second one in Fig. 1 (problem 2), he would find a similarity between the two problems and would be able to produce such an auxiliary-line for problem 2 as drawn with dotted lines in Fig. 1. The reason why he finds a similarity in these two problems is that there are common elements to which he pays deep attention in struggling to solve each. If he can produce an appropriate auxiliary-line and make a succesful proof tree of problem 2 by making use of the "strategies" acquired from problem 1, he will recognize the similarity more clearly again and memorization of the strategies will be reinforced as effective ones in his mind. He will become more and more proficient in solving a variety of auxiliary-line problems as he undergoes such experiences many times.

The purpose of this paper is to make the following two issues clear;

1. What kind of knowledge is the strategy of producing auxiliary-line?
2. How can human beings acquire such strategies by solving some auxiliary-line problems?

The proposals which hopefully answer these two issues will be made in the second and the third section respectively. In the fourth section, we describe the performance of a computer program AUXIL which can acquire the strategies and apply them to future problems efficiently. In the fifth section, the general characteristics of the newly proposed learning method is made clear through comparison with an explanation-based learning method (e.g. Mitchell, Keller & Kedar-Cabelli, 1986; Dejong & Mooney, 1986).

2. Basic mechanism of producing auxiliary-lines

In the 1970's there were some efforts (Wong, 1972; Goldstein, 1973; Ullman, 1975) to make a theorem-prover solve these problems. Wong, who proposed the method of solving them most concretely, showed that the strategies for producing auxiliary-lines can be classified into three types (Wong, 1972). However, the assertion that there are several different types of strategies for producing auxiliary-lines is not desirable for a learning system, because this assertion forces us to prepare different learning methods separately for each of the different types of strategies. It is most desirable to hypothesise that there is only one general mechanism for producing auxiliary-lines.

We have made a hypothesis that associating specific conditions or subgoals, which are generated through constructing a proof tree, with an appropriate figure-pattern is a basic mechanism to draw an auxiliary-line. The (associative) knowledge for performing such an association (called figure-pattern strategy) is expressed as follows;

IF *there exist specific* **conditions** *or/and* **subgoals**
THEN *recall a certain* **figure-pattern**

Figure 2 shows this basic mechanism by use of a figure-pattern strategy. Suppose that a man happens to pay attention to the equality condition, $BP = PC$, because he wants to use it somehow but does not know how to use theorems he knows of. If he hits upon a figure-pattern strategy such as;

IF *there exists a condition* $(XY = YZ)$
THEN *recall a figure-pattern shown in Fig. 2*

then auxiliary-lines suddenly emerge to his eyes by trying to match the pattern to the problem with a newly-added point.

This mechanism can be fully automated by a unification technique of logic programming, Prolog. First, we must express geometrical information with logical predicates of Prolog. Geometrical information is divided into two types; information for constructing geometrical objects and information for constraining relations between constructed objects. We call each of them "individual information" and "relational information", respectively. For example, individual information consists

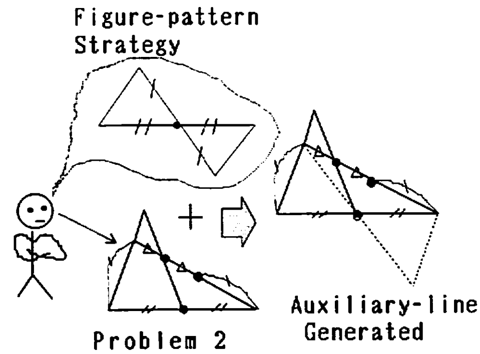


FIG. 2. Basic mechanism for generation of auxiliary-lines.

of information showing the existence of points and lines (segments) and each of them can be expressed with *point* predicate and *line* predicate respectively. Relational information consists of the equality relations of two segments or two angles or two ratios between segments, relations that two segments (lines) are parallel to each other and finally relations that two triangles are congruent or similar. Each of them can be expressed with *equal* predicate, *parallel* predicate, *congruent* predicate and *similar* predicate respectively.

A problem figure can be represented as an assembly of the assertions whose arguments are instantiated to constants corresponding to the points in the problem. On the other hand, a figure-pattern strategy can be expressed in the following form;

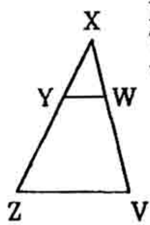
IF ?-*condition*(a predicate showing a given condition) or/and
?-*subgoal*(a predicate showing a subgoal/given goal)
THEN ?-*make_pattern*(a name of a figure-pattern, list of points concerned)
where
make_pattern(a name of a figure-pattern, list of points concerned):
(a set of figure-determining information)→
(all the other information of the figure-pattern).

Its body part consists of *if*-part and *then*-part. Figure-determining information (*if*-part) is the minimum required piece of information for determining the features of the figure-pattern. Therefore, *make_pattern* predicate works as a macro operator which shows a concrete procedure for recalling a figure-pattern. All the information included in both *if*-part and *then*-part of *make_pattern* amounts to all the features the figure-pattern has. We call the set of those features pattern-information. The figure-pattern strategy of Fig. 2, for instance, is shown as strategy 4 in Fig. 3(a).

We show the way of producing auxiliary-line by use of a figure-pattern strategy. First, the figure-pattern strategy must be selected out whose **IF**-part matches to a specific problem element (given condition or subgoal), for example $BP = PC$ in problem 2. And then its **THEN**-part (*make_pattern* predicate) is to be invoked, searching for an appropriate set of substitutions between constants in the problem figure and variables in the figure-pattern strategy. An auxiliary-line can be produced by allowing the constant *new*† to match to a variable existing in the figure-pattern

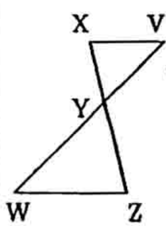
† The *new* point is the one which is to be generated by auxiliary-lines.

Strategy 1



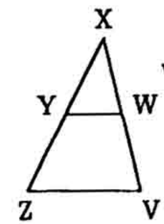

```
IF ?-subgoal(equal(ratio(seg(X,Y),seg(Y,Z)),P)).
THEN    ?-make_pattern(s1,[X,Y,Z,W,V]).
where
 make_pattern(s1,[X,Y,Z,W,V]):-
  (point(X),point(Y),point(Z),point(W),
   point(V),line([X,Y,Z]),line([X,W,V]),
   parallel(line(Y,W),line(Z,V)),line([Y,W]),
   line([Z,V]) →
  ($similar(tri(X,Y,W),tri(X,Z,V)),
   $equal(ang(X,Y,W),ang(X,Z,V)),
   $equal(ang(X,W,Y),ang(X,V,Z)),
$equal(ratio(seg(X,Y),seg(X,Z)),ratio(seg(X,W),seg(X,V))),
$equal(ratio(seg(X,Y),seg(X,Z)),ratio(seg(Y,W),seg(Z,V))),
$equal(ratio(seg(Y,W),seg(Z,V)),ratio(seg(X,W),seg(X,V))),
$equal(ratio(seg(X,Y),seg(Y,Z)),ratio(seg(X,W),seg(W,V)))).
```

Strategy 2



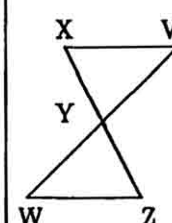

```
IF ?-subgoal(equal(ratio(seg(X,Y),seg(Y,Z)),P)).
THEN    ?-make_pattern(s2,[X,Y,Z,W,V]).
where
 make_pattern(s2,[X,Y,Z,W,V]):-
  (point(X),point(Y),point(Z),point(W),point(V),
   line([X,Y,Z]),line([W,Y,V]),line([X,V]),
   line([W,Z]),parallel(line(X,V),line(W,Z)) →
  ($similar(tri(X,Y,V),tri(Z,Y,W)),
   $equal(ang(X,Y,V),ang(Z,Y,W)),
   $equal(ang(X,V,Y),ang(Z,W,Y)),
   $equal(ang(V,X,Y),ang(W,Z,Y)),
$equal(ratio(seg(X,Y),seg(Y,Z)),ratio(seg(V,Y),seg(Y,W))),
$equal(ratio(seg(X,Y),seg(Y,Z)),ratio(seg(X,V),seg(W,Z))),
$equal(ratio(seg(V,Y),seg(Y,W)),ratio(seg(X,V),seg(W,Z)))).
```

Strategy 3




```
IF      ?-condition(equal(seg(X,Y),seg(Y,Z))).
THEN    ?-make_pattern(s3,[X,Y,Z,W,V]).
where
 make_pattern(s3,[X,Y,Z,W,V]):-
  (point(X),point(Y),point(Z),point(W),point(V),
   line([X,Y,Z]),line([X,W,V]),line([Y,W]),
   line([Z,V]),equal(seg(X,Y),seg(Y,Z)),
   equal(seg(X,W),seg(W,V))       ) →
  ($similar(tri(X,Y,W),tri(X,Z,V)),
   $equal(ang(X,Y,W),ang(X,Z,V)),
   $equal(ang(X,W,Y),ang(X,V,Z)),
   $parallel(line(Y,W),line(Z,V)),
   $equal(ratio(seg(Y,W),seg(Z,V)),ratio(1,2)) ).
```

Strategy 4




```
IF      ?-condition(equal(seg(X,Y),seg(Y,Z))).
THEN    ?-make_pattern(s4,[X,Y,Z,W,V]).
where
 make_pattern(s4,[X,Y,Z,W,V]):-
  (point(X),point(Y),point(Z),point(W),point(V),
   line([X,Y,Z]),line([W,Y,V]),line([X,V]),
   line([W,Z]),equal(seg(X,Y),seg(Y,Z)),
   equal(seg(W,Y),seg(Y,V))       ) →
  ($congruent(tri(X,Y,V),tri(Z,Y,W)),
   $equal(ang(X,Y,V),ang(Z,Y,W)),
   $equal(ang(X,V,Y),ang(Z,W,Y)),
   $equal(ang(V,X,Y),ang(W,Z,Y)),
   $equal(seg(X,V),seg(Z,W)),
   $parallel(line(X,V),line(W,Z))).
```

FIG. 3(a). An example of figure-pattern strategies.

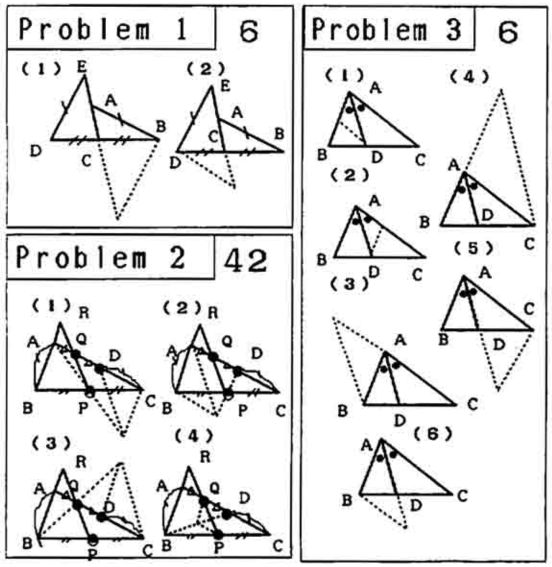


FIG. 3(b). Successful auxiliary-lines produced by figure-pattern strategies.

strategy during the execution of the predicates of *if*-part of *make_pattern*. In order to make the execution of *then*-part of *make_pattern* successful, we must give AUXIL a rule that the call of a predicate including *new* as its arguments always succeeds. The rule is justified by the fact that information including *new* expresses the way of producing auxiliary-lines and is never inconsistent with already existing information. If the call of *make_pattern* predicate ends in success, then the predicates in the *if*-part of *make_pattern* which include *new* as their arguments express information determining the locations of a new point and auxiliary-lines, and the predicates in the *then*-part which include *new* as their arguments express all the new information that holds in the new problem figure after producing auxiliary-lines. In general, there are many successful sets of the substitution by this matching process, each of which corresponds to each of the different auxiliary-lines.

The validity of our hypothesis has been recognized in several auxiliary-line problems. As an example, we show in Fig. 3(b) the results that AUXIL has solved the three problems in Fig. 1 mechanically by making use of four kinds of figure-pattern strategies in Fig. 3(a), which we had given to AUXIL in advance. The number written in the right of each problem name in the figure indicates the number of all of the auxiliary-lines which can possibly be produced by use of those strategies. Only the successful sets of auxiliary-lines are illustrated in the figure.

Here, we are faced with a point at issue; the application of this basic mechanism to those auxiliary-line problems which have many points or many given conditions may considerably reduce the ratio† of the number of the successful auxiliary-lines to the number of all the possible ones. Its typical example is problem 2. In an unfortunate case, AUXIL may discover a successful auxiliary-line only after

† We call it "efficiency ratio" in this paper.

exhaustive repetition with bad auxiliary-lines. Such a performance is clearly different from that of human beings; it is probably because human beings make use of some effective heuristics that they hit upon only a few sets of auxiliary-lines in which a successful one may be included. In the fourth section we will propose one of the plausible answers to what kind of heuristics human beings have.

Finally, the characteristics of figure-pattern strategies are summarized. A figure-pattern strategy is a kind of associative knowledge. The specific element constituting the **IF**-part of a strategy is only a partial element in the problem, and it makes an easy association possible independently of the other elements existing together in the problem, if only there exists the specific element. This feature gives itself both merits of flexible association and demerits of potentially low efficiency ratio.

3. Frustration-based learning

In this section, we propose an algorithm for automatically acquiring figure-pattern strategies through the experiences of solving auxiliary-line problems. Our learning method is based on the concept that a problem element constituting the **IF**-part is one to which a man has paid much attention during the process of solving problems. The problem-solving process, through which several strategies are to be learned, consists of four processes; frustration-identification process, auxiliary-line generation process, proof process and learning process. In the first process, AUXIL focuses on some problem elements which have caused difficulties in reasoning. It then constructs a complete proof tree in the second and third processes. Finally AUXIL acquires some figure-pattern strategies from the problem figure. Each figure-pattern of those strategies has a particular relevance to each of the problem-elements focused in the first process.

3.1. FRUSTRATION-IDENTIFICATION PROCESS

AUXIL is given only a set of basic theorems in elementary geometry as domain knowledge.† Due to the discontinuous gap that exists in the problem space before any auxiliary-line has been produced, there necessarily appear during the process of problem-solving some problem-conditions or subgoals to which any basic theorems cannot be applied. Those problem elements are called frustration elements in this paper. The role of this process consists in identifying these elements during the process of problem solving.

Two kind of frustration elements should be defined because AUXIL has two methods of reasoning, forward and backward reasoning; they are a frustration condition and a frustration subgoal.

Definition 1 (Frustration conditions). *Out of given conditions and problem conditions proved by forward reasoning, those to which any theorems cannot be applied in the forward direction are called frustration conditions.‡ The situation that any theorem cannot be applied to a condition in the forward direction means that even if*

† It does not include algebraic knowledge, such as transformation of equations or calculations accompanied with ratios.

‡ Frustration conditions are limited to relational information. The reason is that individual information tends to be paid less attention to in a reasoning than relational one and thus it will not bring about frustration even if it cannot be made use of.

there is a theorem which has in its body-part a unifiable member to the condition, all the rest member of the body-part cannot be unified to other problem conditions.

Frustration conditions are themselves the roots of the proof forest which has been constructed by forward reasoning using given conditions. This type of frustration corresponds to a man's behavior, he is embarrassed because he does not know how to use the condition.

For the purpose of defining frustration subgoals, we need to determine when backward reasonings should stop. A subgoal which is produced by backward application of a theorem to the upper subgoal is divided into two types; a variable-free subgoal and a variable-included one. A variable-included subgoal is produced when a theorem having in its body-part more than one variable which do not appear in its head-part is applied to an upper subgoal.

Definition 2 (Temporary variable-free subgoals). *Suppose that a set of subgoals have been produced by backward application of a theorem to an upper subgoal and all of them are variable-included subgoals. Some of them may be unifiable to problem conditions. As a result of the unification of some of the subgoals to problem conditions, each of the rest of the subgoals may possibly become variable-free. Such a variable-free subgoal is called a temporary variable-free subgoal.*

Definition 3 (Same-level subgoals). *In general, more than one set of subgoals are generated for an upper goal (subgoal) because there are possibly several theorems to be applied to the goal (subgoal). Then these sets of subgoals are called same-level subgoals to one another.*

Here, we make a hypothesis placed on backward reasoning.

Hypothesis 1. *A backward reasoning can be successively chained to generate new subgoals only under variable-free subgoals. The reason for making this hypothesis is that applications of theorems to variable-included subgoals will increase the indeterminancy of the lower subgoals and that the indeterminancy tends to deprive those subgoals of their value as targets to be proved.*

According to this hypothesis, variable-included subgoals constitute all the lowest leaves in the proof tree whose root is the goal of the problem.

After the above preparations, we can now give a precise definition of frustration subgoals.

Definition 4 (Frustration subgoals). *Out of variable-included subgoals, those which meet the following two requirements are called frustration subgoals.*

- *the variable-included subgoals for which none of the same-level subgoals are variable-free or temporarily variable-free,*
- *variable-included subgoals which cannot be made temporarily variable-free.*

The variable-included subgoals, for which some of the same-level subgoals are variable-free or temporarily variable-free, cannot be frustration subgoals because it is more natural for us to think that backward reasoning will be successively chained further under the variable-free subgoals in the same-level than to think that backward reasoning stops at the nodes of the variable-included subgoals. AUXIL, on the way to extending a backward proof tree, memorizes as candidates of

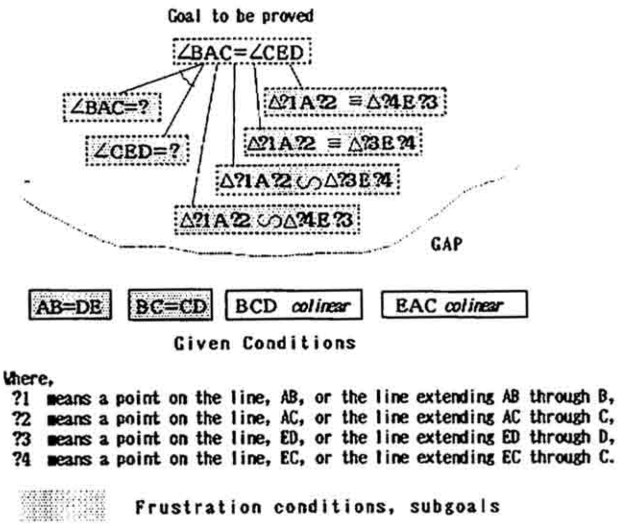


FIG. 4. A frustration tree (in the case of problem 1 in Fig. 1).

frustration subgoals, those variable-included subgoals which have at least one variable-free subgoal in the same level or which have been made temporarily variable-free.

Examples of frustration elements which AUXIL has generated in this process of problems 1 and 2 in Fig. 1 are shown in Figs 4 and 5 respectively (the tree constructed in this process is called a frustration tree). The shaded nodes represent frustration conditions and frustration subgoals. The variable-included subgoals shown in parenthesis are candidates of frustration subgoals.

3.2. AUXILIARY-LINE GENERATION PROCESS

AUXIL produces auxiliary-lines by making use of already acquired figure-pattern strategies or by conforming to suggestions from a user (in case it has no strategies applicable). Suggestions are to be realized by a user's asserting clauses which determines the location of auxiliary-lines and new points. For example, in problem 2, a suggestion that a new point should be an intersection between the line extending *DP* through *P* and the line parallel to *DC* passing *B* (see Fig. 7), can be realized by assertions, such as *line*([*d*,*p*,*new*]), *parallel*(*line*(*d*,*c*), *line*(*b*, *new*)).

In fact, AUXIL has another way of generating auxiliary-lines by itself without receiving suggestions even if it has no figure-pattern strategies. A precise description of the way is as follows; a frustrated-state that a theorem cannot be applied to a certain problem element is caused by the fact that there are some variables left in the theorem and they cannot be instantiated to constants in the problem. Then, if a new set of information obtained by an attempt to substitute the variables with the constants including *new* is consistent, this new set of information is itself information which determines the location of auxiliary-lines and a new point.

It does not matter in acquiring new figure-pattern strategies which way is to be taken, because it is merely one of the steps for constructing a complete proof tree.

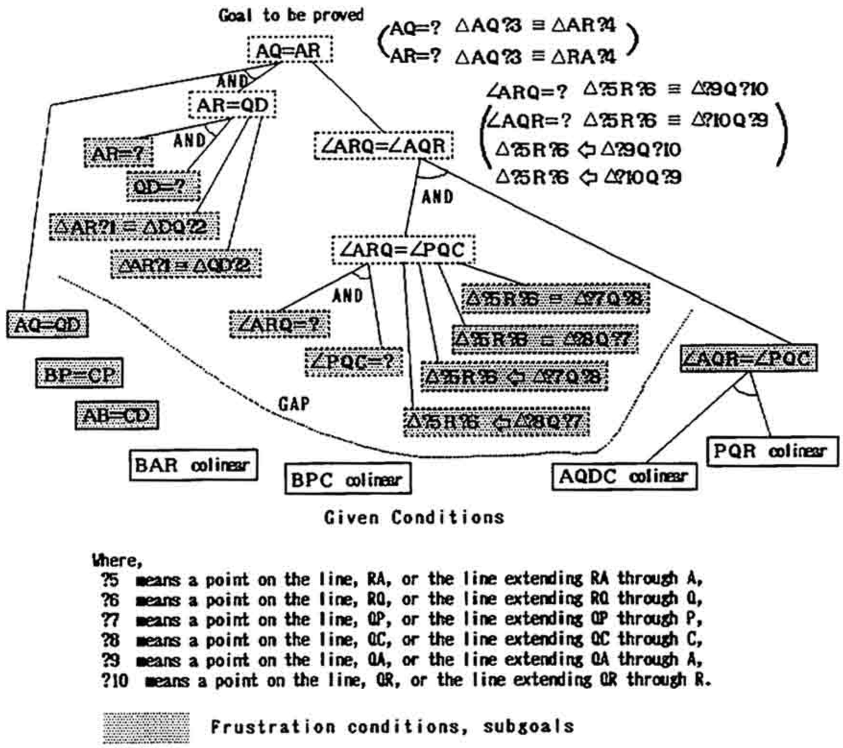


FIG. 5. A frustration tree (in the case of problem 2 in Fig. 1).

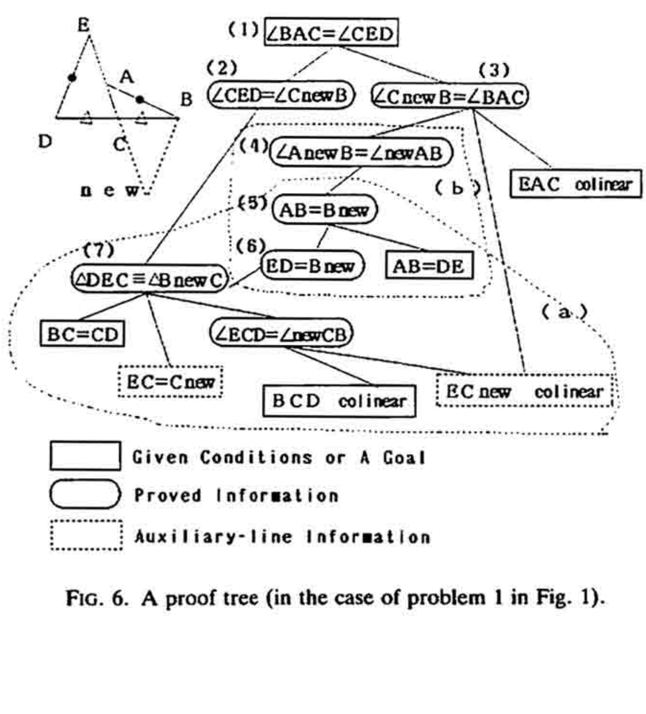


FIG. 6. A proof tree (in the case of problem 1 in Fig. 1).

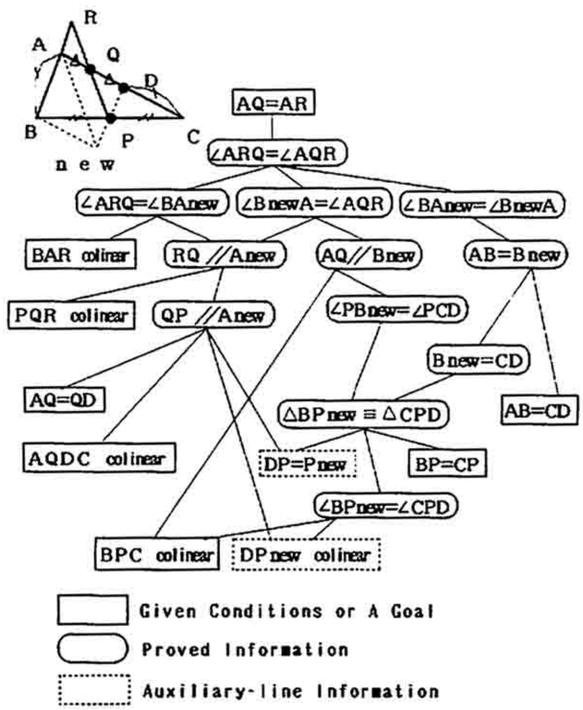


FIG. 7. A proof tree (in the case of problem 2 in Fig. 1).

In this paper, auxiliary-lines are to be provided by a user's suggestion in case AUXIL has no strategies at all.

3.3. PROOF PROCESS

AUXIL reasons by use of basic theorems, until it completes constructing a proof tree. If some new frustration conditions should be identified during the process of forward reasonings, AUXIL backtracks to the auxiliary-line generation process and redraws another line.† Examples of a complete proof tree of problem 1 and problem 2 are shown in Figs 6 and 7 respectively.

3.4. LEARNING PROCESS

In this process, AUXIL acquires some figure-pattern strategies from a problem figure. The following three subprocesses give a criterion about which part of the problem figure should be extracted as a strategy; frustration-tree analysis subprocess, figure-restriction subprocess, and limited forward-reasoning subprocess. In the last subprocess, figure-pattern generalization subprocess, those figure-patterns specific to the problem figure are to be generalized into figure-pattern strategies for future use.

† In case an auxiliary-line has been produced by the user's proper suggestion, such a frustrated state does not appear again.

3.4.1. Frustration-tree analysis subprocess

The role of this process is

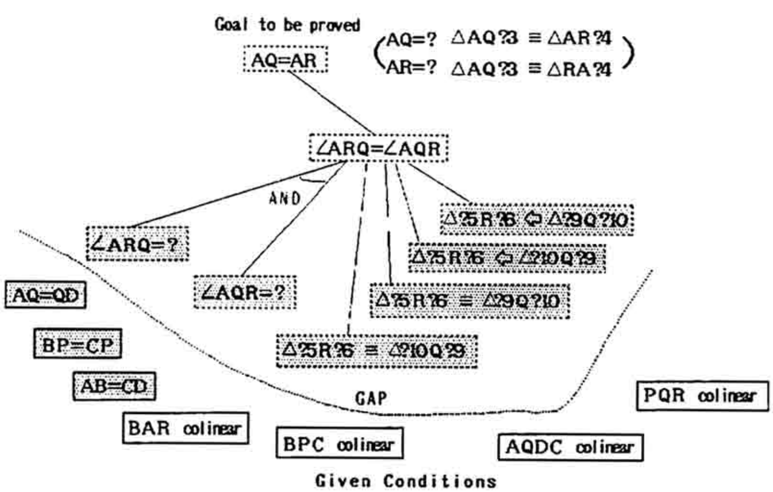
1. to investigate which part of the frustration tree constructed in the frustration-identification process has contributed to making a complete proof tree by comparing the frustration tree with proof tree, and then
2. to make a revised frustration tree by re-identifying frustration elements which turn out to be resolved.

The revised frustration tree of problem 2 is illustrated in Fig. 8. All the lower subgoal structures under *equal*(*ang*(*a*,*r*,*q*),*ang*(*a*,*q*,*r*)) have been removed by the first task, and then those candidates of frustration subgoals for *equal*(*ang*(*a*,*r*,*q*), *ang*(*a*,*q*,*r*)) which have been reserved in the frustration-identification process are now retrieved as the new frustration subgoals by the second task. The frustration tree of problem 1 (Fig. 4) does not receive any variation in this subprocess.

3.4.2. Figure-restriction subprocess

In this subprocess, the following two tasks are performed for each of the frustration elements in the revised frustration tree;

1. to investigate which single basic theorem is concerned with resolving each frustration element, and then
2. to restrict the range of the "figure world" within which the figure-pattern having relevance to each frustration element is to be determined in the next subprocess.



Where,
?5 means a point on the line, RA, or the line extending RA through A,
?6 means a point on the line, RQ, or the line extending RQ through Q,
?7 means a point on the line, QP, or the line extending QP through P,
?8 means a point on the line, QC, or the line extending QC through C,
?9 means a point on the line, QA, or the line extending QA through A,
?10 means a point on the line, QR, or the line extending QR through R.

Frustration conditions, subgoals

FIG. 8. A revised frustration tree (in the case of problem 2 in Fig. 1).

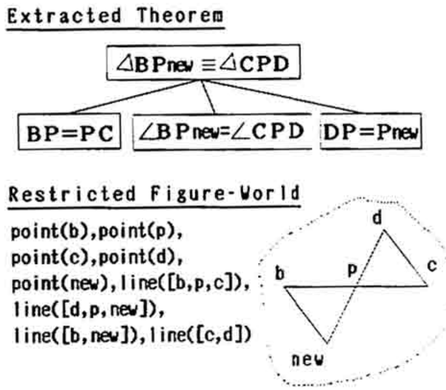


FIG. 9. Figure-restriction subprocess (in learning, motivated by the frustration condition, $BP = PC$, of problem 2).

In this paper, we think that "figure world" is made of individuals, such as points and lines, and relations on individuals.† From this concept, the above mentioned range can be restricted within the "figure-world" constructed out of

- those individuals which are the **points** included in the single basic theorem as well as the **lines** seen to be in the problem figure passing on those points, and
- relations on those individuals.

It is in this subprocess that the classification of information into two categories, which has been mentioned in the second section, exhibits its role in learning.

Figure 9 shows a basic theorem which is concerned with resolving the frustration condition $BP = PC$ of problem 2 as well as the restricted figure-world made of individual information included in the theorem, such as *point*(b), *point*(p), *point*(c), *point*(d), *point*(new), *line*([b,p,c]), *line*([d,p,new]), *line*([b,new]), *line*([c,d]).

3.4.3. Limited forward-reasoning subprocess

In this subprocess, AUXIL searches by forward-reasoning every piece of information which holds within the "figure-world" determined in the previous subprocess. We mean by the term "limited reasoning" reasonings performed within a restricted "figure-world".

In the above example of problem 2, *equal*(*seg*(b,new),*seg*(c,d)), *equal*(*ang*(b,new,p),*ang*(c,d,p)), *equal*(*ang*(new,b,p),*ang*(d,c,p)), *parallel*(*line*(b,new),*line*(c,d)) are obtained as new information, as shown in Fig. 10(a).‡

The role of this subprocess is to identify the figure-pattern which has essential relevance to each frustration element. For that purpose, two main tasks are performed:

(1) one is to collect all the pieces of information which hold within the restricted "figure-world";

† In situation semantics (Barwise, 1983) also, situations are made of individuals and relations.

‡ *congruent*(*tri*(b,p,d),*tri*(c,p,new)), *equal*(*seg*(b,d),*seg*(c,new)) etc. should not be obtained because individual information, such as *line*([b,d]), *line*([c,new]), do not exist.

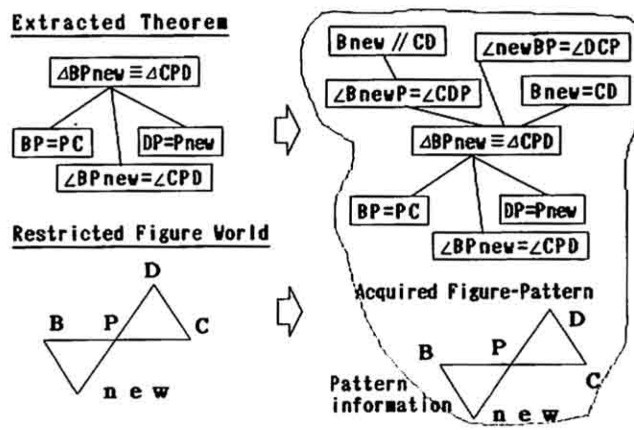


FIG. 10(a). Limited forward-reasoning subprocess (in learning, motivated by the frustration condition, $BP = PC$, of problem 2).

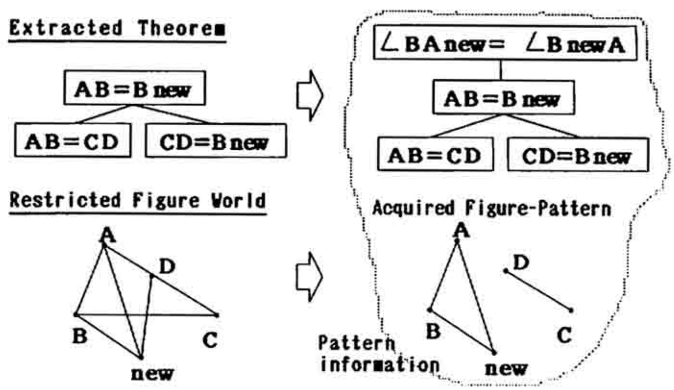


FIG. 10(b). Limited forward-reasoning subprocess (in learning, motivated by the frustration condition, $AB = CD$, of problem 2).

(2) the other is to remove *line* information which has not been mentioned in this forward-reasoning. We mean by the term "mentioned *line* information" the one that is used in this reasoning or that constitutes at least one relational information used in this reasoning. For example, in Fig. 10(b), which shows forward-reasoning subprocess for the frustration condition $AB = CD$ of problem 2, the following pieces of information are removed; information that points D and C are colinear with the point A, information that there is a line between the points B and C, and information that there is a line between the points D and *new*.

Through these tasks, all the pieces of relevant information (pattern-information) to each frustration element are identified, and they constitute a figure-pattern. It also includes pieces of information that happen not to have been used in the proof tree but may be useful in future problems, for example *equal*(*ang*(*new*,*b*,*p*),*ang*(*d*,*c*,*p*)) in Fig. 10(a).

The significance of limited forward reasoning is that it enables AUXIL to collect

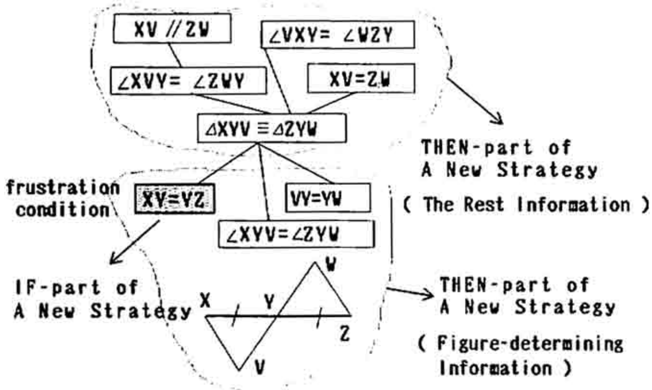


FIG. 11. Figure-pattern generalization subprocess (in learning, motivated by the frustration condition, $BP = PC$, of problem 2).

all the pieces of information of the object which will be a target of association in a future problem. These include not only the ones which construct a complete proof tree of the specific problem but also the ones which have not been directly concerned with the problem-solving process. This reasoning is extremely significant in learning because our learning method is based upon the concept that information truly relevant to a frustration element cannot be found out from the tree of the processes which have contributed to the problem-solving of the specific problem.

3.4.4. Figure-pattern generalization subprocess

All the pieces of pattern-information obtained are instantiated by constants in the problem. AUXIL needs to generalize them as strategies for future use. The method for generalization is basically the same as that in EBL (Mitchell *et al.*, 1986); to generalize each node of the proof tree constructed in the limited forward reasoning, with the constraints among variables in the used theorem reserved. A generalized tree in the above example is shown in Fig. 11.

3.4.5. Formation of figure-pattern strategies

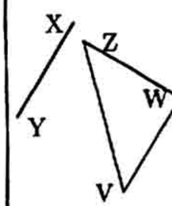
For every frustration element which has been a motivation for learning, a corresponding figure-pattern strategy is created. A frustration element constitutes the **IF**-part of a strategy, and all the generalized pattern-information constitutes the **Then**-part, *make_pattern* predicate. The *if*-part in the body of *make_pattern* predicate, figure-determining information, is made up of both individual information of the figure-pattern and the body-part of the theorem extracted in the figure-restriction subprocess. The *then*-part is made up of all the rest of pattern-information. An example of forming a figure-pattern strategy for the frustration element, $BP = PC$, of problem 2 is illustrated in Fig. 11.

Some examples of acquired strategies are to be described. In case AUXIL solves problems 1 and 2 by constructing the proof trees of Figs 6 and 7, it acquires the strategies shown in Figs 12(A, B, C, D) and 13(A, B, E, F, G) respectively. In case it solves problem 2 by generating such auxiliary-lines as (4) in Fig. 3(b), it acquires strategies E, F, H in Fig. 13.

Strategy A

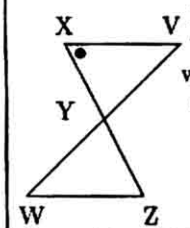
Same as *Strategy 4* in Fig.3(a)

Strategy B



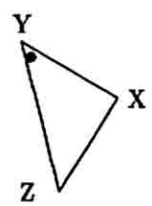

```
IF      ?-condition(equal(seg(X,Y),seg(Z,W))).
THEN    ?-make_pattern(p_b,[X,Y,Z,W,V]).
where
 make_pattern(p_b,[X,Y,Z,W,V]):-
  (point(X),point(Y),point(Z),point(W),
   point(V),equal(seg(X,Y),seg(Z,W)),
   equal(seg(X,Y),seg(W,V)),line([X,Y]),
   line([Z,W]),line([W,V]),line([Z,V]) ) →
  ($equal(seg(Z,W),seg(V,W)),
   $equal(ang(W,Z,V),ang(W,V,Z))).
```

Strategy C




```
IF      ?subgoal(equal(ang(V,X,Y),P)).
THEN    ?-make_pattern(p_c,[X,Y,Z,W,V]).
where
 make_pattern(p_c,[X,Y,Z,W,V]):-
  (point(X),point(Y),point(Z),point(W),
   point(V),congruent(tri(X,Y,V),tri(Z,Y,W)),
   line([X,Y,Z]),line([W,Y,V]),
   line([X,V]),line([W,Z])       ) →
($equal(seg(X,Y),seg(Y,Z)),$equal(seg(W,Y),seg(Y,V)),
 $equal(seg(X,V),seg(Z,W)),$parallel(line(X,V),line(W,Z)),
 $equal(ang(X,Y,V),ang(Z,Y,W)),
 $equal(ang(X,V,Y),ang(Z,W,Y)),
 $equal(ang(V,X,Y),ang(W,Z,Y)) ).
```

Strategy D




```
IF      ?-subgoal(equal(ang(X,Y,Z),P)).
THEN    ?-make_pattern(p_d,[X,Y,Z]).
where
 make_pattern(p_d,[X,Y,Z]):-
  (point(X),point(Y),point(Z),
   equal(seg(X,Y),seg(X,Z)),
   line([X,Y]),line([X,Z]),line([Y,Z])) →
  equal(ang(X,Y,Z),ang(X,Z,Y)).
```

FIG. 12. Acquired figure-pattern strategies from problem 1.

- The frustration condition, $AB = DE$, in problem 1 has motivated AUXIL to learn strategy B. Some pieces of individual information which were selected in the figure-restriction subprocess for this frustration condition, such as *line*([*e*,*a*,*new*]) and *line*([*b*,*d*]), have been removed in the limited forward reasoning, resulting in strategy B's lacking in some constraints which determine a relative location of segment *XY* against triangle *ZWV*. Consequently, application of strategy B would not be able to locate a new point uniquely because of the weak constraints in its

Strategy A

Same as *Strategy 4* in Fig.3(a)
but different in *make_pattern* procedure

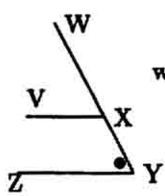
Strategy B

Same as *Strategy B* in Fig.8

Strategy E

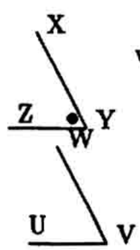
Same as *Strategy 3* in Fig.3(a)

Strategy F



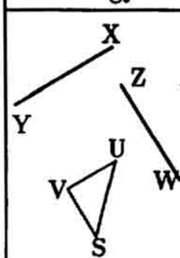

```
IF      ?-subgoal(equal(ang(X,Y,Z),P)).
THEN    ?-make_pattern(p_f,[X,Y,Z,W,V]).
where
  make_pattern(p_f,[X,Y,Z,W,V]):-
    (point(X),point(Y),point(Z),point(W),
     point(V),parallel(line(X,V),line(Y,Z)),
     line([W,X,V]),line([X,V]),line([Y,Z]) ) →
    $equal(ang(X,Y,Z),ang(W,X,V)).
```

Strategy G




```
IF ?-subgoal(equal(ang(X,Y,Z),P)).
THEN ?-make_pattern(p_g,[X,Y,Z,W,V,U]).
where
  make_pattern(p_g,[X,Y,Z,W,V,U]):-
    (point(X),point(Y),point(Z),point(W),
     point(V),point(U),
     parallel(line(X,Y),line(W,V)),
     parallel(line(Z,Y),line(U,V)),line([X,Y]),
     line([Y,Z]),line([W,V]),line(U,V])) →
    $equal(ang(X,Y,Z),ang(W,V,U)).
```

Strategy H




```
IF ?-condition(equal(seg(X,Y),seg(Z,W))).
THEN ?-make_pattern(p_h,[X,Y,Z,W,V,U,S])
where
  make_pattern(p_h,[X,Y,Z,W,V,U,S]):-
    (point(X),point(Y),point(Z),point(W),
     point(V),point(U),point(S),
     equal(seg(X,Y),seg(Z,W)),
    equal(ratio(seg(X,Y),seg(V,U)),ratio(1,K)),
    equal(ratio(seg(Z,W),seg(V,S)),ratio(1,K)),
line([X,Y]),line([Z,W]),line([U,V]),line([V,S]),line([U,S]) )
 → ($equal(seg(V,U),seg(V,S)),
   $equal(ang(V,U,S),ang(V,S,U))).
```

FIG. 13. Acquired figure-pattern strategies from problem 2.

figure-determining information. A strategy like this which cannot generate unique auxiliary-lines for itself is called a passive strategy. On the other hand, a strategy which generates unique auxiliary-lines is called a positive strategy. We will explain how to make use of passive strategies in the next section in detail. Strategies A, C, E in Figs 12 and 13 are positive and the rest are passive.

- Strategy H is very close to strategy B. If a man learns both strategies, he will integrate both into forming a more generalized strategy, which says that if there exists a frustration condition, $XY = WZ$, then recall an isosceles triangle whose equal segments (VU and VS) are obtained by scaling XY and WZ in the same degree. The method of learning a higher-ordered, abstract concept like this can be found in (Shavlik & Dejong, 1987), whose ability is beyond current AUXIL.

3.5. THE MEANING OF FRUSTRATION-BASED LEARNING

The name of our learning method, frustration-based learning, is derived from the fact that the frustration elements generated during a problem solving process are themselves motivations for learning and then the resolution of each frustrated state gives instructions about the range of the "figure-world" within which the figure-pattern to be learned is restricted.

We discuss what it means to solve problems through the frustration-based learning method. As seen in Fig. 14, AUXIL has learned four strategies from problem 1, five from problem 2. In other words, AUXIL has recognized through frustration-based learning that problem 1 is composed of four figure-patterns corresponding to four acquired strategies and that problem 2 is composed of five figure-patterns. Notice that out of these four and five figure-patterns two are common to both problems. This fact will probably explain why a man finds some similarities between these two problems.

To sum up, solving a problem through frustration-based learning means at the

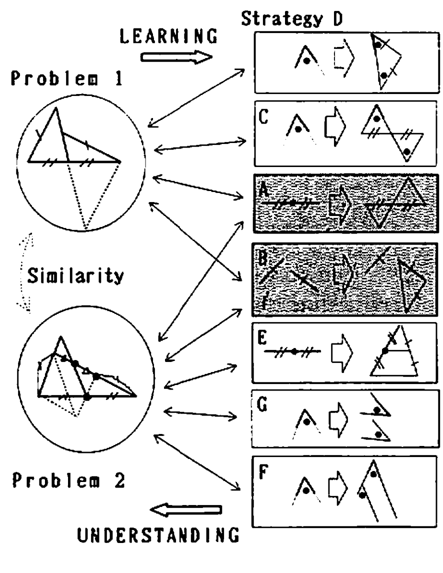


FIG. 14. The meaning of learning by the frustration-based method.

same time understanding it as a composition of figure-patterns each of which is made of pieces of information in the **THEN**-part of each acquired strategy. Such a way of understanding may simulate human comprehensive understanding. We can conclude that frustration-based learning is not only a method for learning associative knowledge but also can be regarded as a framework for understanding that catches some aspect of human abilities to visualize the overall problem.

4. Learning system AUXIL

We will describe how intelligently AUXIL comes to produce an appropriate auxiliary-line in a new problem after it has learned some figure-pattern strategies. Suppose in this section that AUXIL tries to solve problem 2 in a situation that it has learned strategies A, B, C, D from problem 1 and strategy E from another problem.

First, AUXIL makes a frustration tree shown in Fig. 5 in the frustration-identification process. Then, auxiliary-lines are generated by utilization of some strategies. For this purpose, AUXIL has the following heuristics about a multiple use of several strategies.

Heuristic 1. *A new point to be generated should be such that its addition would enable AUXIL to make multiple use of figure-pattern strategies, resulting in resolving as many frustrated states as possible.*

This heuristic can be implemented in the following procedures.

(1) First, a temporary set of auxiliary-lines and a new point are produced by applying a selected strategy in response to a certain frustration element of the present problem.
(2) Secondly, it is investigated if there is any other figure pattern found in the new problem figure after procedure 1 and as a result of this if one of the other frustrated states is resolved.† If there are any, procedure 2 is repeated until it finds no more frustration elements resolved. If there is no figure-pattern found other than in procedure 1, AUXIL backtracks to procedure 1 and retries to generate another temporary auxiliary-lines.
(3) If more than two frustrated states can be resolved through procedure 1 and 2, AUXIL adopts the temporary set of auxiliary-lines and a new point as a promising one.

The only way of using passive strategies is a passive application in procedure 2 together with other positive strategies for judging whether the temporary auxiliary-lines are promising or not.

Such a multiple use of strategies including passive ones is very significant because it enables AUXIL to simulate an efficient behavior of human beings, that is to find out only promising candidates by eliminating unpromising ones. We show some examples in Fig. 15. Applications of strategy A (positive one) to the frustration condition $BP = PC$ together with strategy E (positive one) to $AQ = QD$ and

† This is also one way of applying figure-pattern strategies because their application intrinsically means recalling or finding corresponding figure-patterns in the problem figure. When they are used for finding a figure-pattern as in this case, AUXIL does not employ the rule that the call of a predicate including *new* as one of their arguments always succeeds. Such a way of using strategies is called passive application.

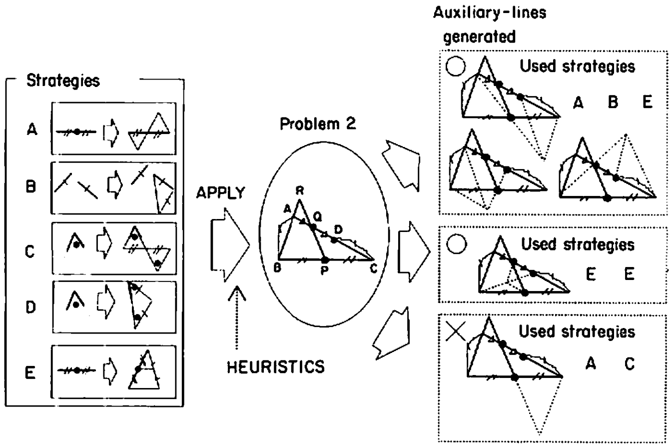


FIG. 15. Multiple use of strategies.

strategy B (passive one) to $AB = CD$ will selectively produce only two successful auxiliary-lines shown in (1) and (2) of Fig. 3(b). Applications of strategy A (positive one) to the frustration condition $AQ = QD$ together with strategy E (positive one) to $BP = PC$ and strategy B (passive one) to $AB = CD$ will selectively produce only one successful auxiliary-line shown in (3) of Fig. 3(b). Applications of strategy E to both frustration conditions, $BP = PC$ and $AQ = QD$, will produce only one successful auxiliary-line shown in (4) of Fig. 3(b). However, it sometimes happens that wrong auxiliary-lines are produced. The example is a set of lines produced by applications of strategy A to $BP = PC$ together with strategy C to the frustration subgoal $\angle ARQ \stackrel{?}{=} ?$,† which is a mistake that even a human being may sometimes make.

Now, it is evident that the low efficiency ratio in problem 2 as mentioned in Section 2 has been in fact caused by a single use of a strategy, either A or E. We conclude that a multiple use of strategies can be a plausible way of selectively producing only several promising patterns of auxiliary-lines. The multiple use owes the characteristics of a figure-pattern strategy that the **IF**-part is only a partial element in the problem and thus it provides an easy association.

Lastly, we must mention that our learning method has a remarkable effect only on an ability to produce promising patterns of auxiliary-lines selectively, not on other intellectual behaviors of human beings in solving problems in elementary geometry. The largest problem in implementation of a computer program for elementary geometry problem solving is probably that it cannot intrinsically avoid performing useless reasonings which can possibly be caused by the existence of multiple ways of matching between relational information constituting a basic theorem and a problem element. Especially, equality conditions of two angles potentially allow many ways of matching, and can be a main cause of useless reasonings. On the other hand, human beings perform effective reasoning by visually searching only promising ways of matching. Such intellectual behavior is beyond our learning method as well as

† A new point is the intersection between the line extending RP through P and the line parallel to RB passing C.

other problem solving systems which are based on symbolic knowledge representation.

5. Discussion

In this section, we will discuss the characteristics of frustration-based learning (FBL) through comparison with explanation-based learning (EBL) (e.g. Mitchell *et al.*, 1986; Dejong & Mooney, 1986; Mitchell, Utgoff & Banerji, 1986; Minton, Carbonell, Etziono, Knoblock & Knokka, 1987) macro operator (Cheng & Carbonell, 1986; Minton, 1985) and chunking mechanisms (Laird, Newell & Rosenbloom, 1987).

Although the three are all different from one another in the way of generalization and in the way of selecting targets to be learned, they are all within a framework of learning target-concepts by generalizing or chunking all the lower subgoal structures of the targets in the proof tree (explanation tree). EBL-knowledge is represented in the form of IF-THEN. The elements in its **IF**-part are conjunctions of sufficient conditions for proving the target-concept in the **THEN**-part because all the lower subgoal structures in the verified proof tree are to be chunked or generalized.

On the contrary, FBL is a method for learning knowledge which tells us how to associate a certain problem element with a certain object. Therefore, FBL-knowledge takes the following forms;

IF X **THEN** Y

where

$$Y: -A_1 \rightarrow B_1.\ Y: -A_2 \rightarrow B_2. \ldots\ Y: -A_n \rightarrow B_n.$$

The **IF**-part is an element for invoking an association, in other words, showing when to make use of this FBL-knowledge. The **THEN**-part is a procedure for association, whose body-part includes all the information of the object to be recalled. FBL-knowledge has a meaning that if you are confronted with such difficulty as X, then recall A_i in the problem and then you can find such new information as B_i.

The most distinguishing characteristics of FBL are that FBL-knowledge has an element indicating when to invoke itself, which cannot be found in any EBL-knowledge because EBL-knowledge itself does not have any information except IF-THEN production knowledge, and that FBL-system learns objects to be recalled, not sequences of proofs. Therefore, FBL-system can exhibit an excellent ability to cope with problems requiring associations and to understand them as compositions of several recalled objects. An auxiliary-line problem in elementary geometry can be regarded as a typical one of such problems.

FBL-knowledge has an inherent defect which is not in EBL-knowledge. That is, it is not assured that it will contribute to the solutions of future problems when it is applied to them, because FBL-knowledge is merely an instance of successfully resolving frustrations in the past problems and thus the connection between its if-part and then-part has never been verified to work unlike in EBL-knowledge. However, it is our belief that associative knowledge cannot be or should not be inherently verified, but merely reinforced by frequent occurrences in the past experiences. In that sense, the concept of FBL is similar to that of case-based learning (Hammond, 1986; Bain, 1986) and FBL, in a way, provides one particular way of making up databases of past instances. AUXIL can make up for this drawback by providing the following abilities:

1. it attaches a frequency value indicating how many times the strategy has served to resolve frustrations;
2. in the limited forward-reasoning subprocess it tries to obtain as much information as possible which a figure-pattern strategy has.

Here, we should explain that EBL-knowledge is not suitable for solving intelligently such problems as auxiliary-line problems that require some associations. Suppose that an EBL-system learns by solving problem 1, and after that it tries to solve problem 2 for itself. Let's consider three EBL-systems; (a) an EBL-system in a narrow sense which regards only the goal of the present problem as a target-concept; (b) an EBL-system in a broad sense which regards all the subgoals in the problem space as target-concepts and learns macro-operators; and finally (c) SOAR (Laird *et al.*, 1987) which learns control knowledge by chunking all the problem-solving processes lower than the problem-space level, such as decision level or evaluation level.

(a) This type of system learns a generalized form of the whole subgoal structure of problem 1. The EBL-knowledge cannot be applied to problem 2 because problem 2 does not have some of the constraints specific to problem 1, such as E, A, C are colinear (see Fig. 6). However, we can easily refute this because a man who has experienced problem 1 before may find it easier to solve problem 2 than one who has not. It is because both problems require the same way of association in spite of the difference in their whole problem structures.

(b) This type of system regards all the subgoals ((1) ~ (7)) in Fig. 6 as target-concepts. For example, a target concept of (5) is

IF XY = YZ, XYZ colinear, WY = YV, WYV colinear, XW = ST
THEN ZV = ST
(where X, Y, Z, W, V, S, T are all variables).

Out of these macro-operators, only knowledge (4) and knowledge (5) are useful for producing appropriate auxiliary-lines in problem 2. For example, a substitution such as {X/b, Y/p, Z/c, W/a, V/new, S/c, T/d} produces auxiliary-lines of (1) in Fig. 3(b). However, there is no straightforward way to find out a few useful macro-operators among many other useless ones learned by this type of learner. The most fatal drawback of this system is that it does not perform a selective learning and hence cannot catch essential strategies for solving problem 1 which are buried in the problem solving process.

(c) In a situation that there are several options for available actions in a problem space level out of which one action must be selected, SOAR makes a subgoal to decide which one is to be taken and goes down to the decision level under the subgoal. In order to make a decision at that level, SOAR should evaluate the effect of each action and compare one with another. Then, SOAR goes down further to an evaluation level. And after undergoing these lower levels, SOAR resolves the subgoal in the problem space level and at the same time learns control knowledge. Such a method would not be applicable to the domain of auxiliary-line problems in elementary geometry, because it seems that human beings produce one of the appropriate auxiliary-lines not by undergoing any explicit lower levels of problem solving processes, but rather by resorting to *associations within the problem space level.* Consequently, chunking mechanism of SOAR may not be effective to learn a type of control knowledge which is truly needed to solve those problems that requires appropriate association.

We can conclude that a target-concept learner, EBL-system, is not suitable for problems like auxiliary-line problems.

Here, let's consider why FBL-knowledge is flexible enough to be applied to several problems with different problem structures and why FBL method can avoid producing many useless pieces of knowledge.

1. A high flexibility of FBL-knowledge owes the principle that FBL learns a pair of a frustration element and an object which has made contribution to resolving the frustrated state. It is completely different from EBL in that it learns all the lower subgoal structures under a subgoal. For example, the above mentioned EBL-knowledge (5), whose target was a subgoal $AB \stackrel{?}{=} Bnew$, is a kind of macro operator constrained by all the information included in the dotted enclosure (a) in Fig. 6. It has a constraint that $ED = Bnew$ should be proved by a theorem about congruence of triangles. On the contrary, FBL-knowledge AUXIL has learned from a frustration condition $AB = DE$, which corresponds to that EBL-knowledge, is shown as an enclosure (b) in Fig. 6 (strategy B in Fig. 12). We can say that strategy B has been learned from the following ideas; that is, a learner has an intensive interest in how to use the condition $AB = DE$ (forming a new isosceles triangle by obtaining a new segment which is equal to DE in length and has one common point with AB), and does not in particular care about how to obtain the condition $ED = Bnew$. Consequently, strategy B has less constraint in application than EBL-knowledge (5).
2. It owes the fact that frustration elements are themselves motivations for learning that FBL can avoid producing many useless pieces of knowledge. On the other hand, chunking all the lower subgoal structures results in the fact that a piece of EBL-knowledge may be completely included in other pieces of EBL-knowledge. It mainly causes EBL-knowledge to learn many redundant and useless macro-operators.

Lastly, we will discuss the introduction of frustration as important guidance for learning. In FBL, an explicit definition of frustration in the process of making a proof tree serves to identify the problem elements to be paid attention to. Some researches have reported that an identification of focused elements has a significant effect on learning. The most typical one is, as we guess, UNIMEM by Lebowitz (1986). He introduces "interest heuristic" for controlling an explanation process as a learner's interest tells. However, FBL and UNIMEM are different in the following respect. In UNIMEM interest heuristic is made use of for cutting off unimportant details which do not attract a learner's attention, while in AUXIL frustration elements are themselves motivations for learning and at the same time serve to determine the scales of the strategies to be learned.

6. Conclusion

Appropriate associations invoked by certain problem elements lead to the production of successful auxiliary-lines. Such an association can be regarded as a kind of opportunistic forward reasoning, which is performed without profound considerations about whether it will contribute to proving the goal of a problem or not.†

In this paper, we propose a new learning method, named frustration-based learning. It can show us a method of acquiring a figure-pattern strategy which is a kind of associative strategy. A figure-pattern strategy takes the following form;

IF *specific problem elements* **THEN** *the procedure for association*

We will summarize the characteristics of frustration-based learning.

1. *Specific problem elements* are made up of frustration elements which have been generated in the process of making efforts to solve problems. Generation of frustration in the frustration-identification process is a motive force to learning. The explicit definition of frustration is indispensable to learning associative strategies.
2. *The procedure of association* is a kind of macro operator for producing or finding in the problem space a figure-pattern which is defined by all the pattern-information in its body-part.
3. Classification of geometrical information into two types is essential to learning; individual and relational.
4. Determining which pieces of information are truly relevant to each of the frustration elements means at the same time learning each figure-pattern strategy. These pieces of information are obtained through the following two steps. The first step is to devise a figure-world that is made of (1) individual information included in the single basic theorem which has been concerned with resolving each frustrated state, and (2) relational information that holds in the problem space only on individual information. The second step is to obtain by forward reasoning all the pieces of information which hold within that restricted figure-world.
5. Learning a problem through frustration-based learning means to understand it as a composition of figure-patterns each of which has those features represented in the **THEN**-part of the corresponding strategy. In conclusion, the frustration-based learning method is regarded as a method for learning some essential figure-patterns which underlie and structurize a problem solving process of elementary geometry.

References

ANDERSON, J. R. (1983). *The Architecture of Cognition*, chapter 4. Harvard University Press.

BAIN, W. M. (1986). A case-based reasoning system for subjective assessment. *Proceedings of AAAI-86*, pp. 523–527.

BARWISE, J. & PERRY, J. (1983). *Situations and Attitudes*. Cambridge, MA: MIT Press.

CHENG, P. W. & CARBONELL, J. G. (1986). The FERMI system: inducing iterative macro-operators from experience. *Proceedings of AAAI-86*, pp. 490–495.

DEJONG, G. & MOONEY, R. (1986). Explanation-based learning: an alternative view. *Machine Learning*, **1**, 145–176.

† Anderson (1983) also suggests that data-driven forward reasoning can distinguish experts from novices.

GOLDSTEIN, I. (1973). *Elementary geometry theorem proving.* AI Memo No. 280, MIT.

HAMMOND, K. J. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. *Proceedings of AAAI-86,* pp. 556–560.

LAIRD, J. E., NEWELL, A. & ROSENBLOOM, P. S. (1987). SOAR: an architecture for general intelligence. *Artificial Intelligence,* **33,** 1–64.

LEBOWITZ, M. (1986). Integrated learning: controlling explanation. *Cognitive Science,* **10,** 219–240.

MINTON, S., CARBONELL, J. G., ETZIONI, O., KNOBLOCK, C. A. & KUOKKA, D. R. (1987). Acquiring effective search control rules: explanation-based learning in the PRODIGY system. *Proceedings of the 4th International Workshop on Machine Learning,* pp. 122–133.

MINTON, S. (1985). Selectively generalizing plans for problem-solving. *Proceedings of IJCAI-85,* pp. 596–599.

MITCHELL, T. M., KELLER, R. M. & KEDAR-CABELLI, S. T. (1986). Explanation-based generalization: a unifying view. *Machine Learning,* **1,** 47–80.

MITCHELL, T. M., UTGOFF, P. E. & BANERJI, R. (1986). Learning by experimentation: acquiring and refining problem-solving heuristics. In R. S. MICHALSKI, J. G. CARBONELL & T. M. MITCHELL, Eds. *Machine Learning: An Artificial Intelligence Approach,* vol. 2. Morgan Kaufmann.

SHAVLIK, J. W. & DEJONG, G. F. (1987). BAGGER: an EBL system that extends and generalizes explanation. *Proceedings of AAAI-87,* pp. 516–520.

ULLMAN, S. (1975). *A model-driven geometry theorem prover.* AI Memo No. 321, MIT.

WONG, R. (1972). *Construction heuristics for geometry and a vector algebra representation of geometry.* Memo No. 28, MIT.