

PCLEARN: A Computer Model for Learning Perceptual Chunks

Masaki Suwa and Hiroshi Motoda

Advanced Research Laboratory, Hitachi Ltd., 2520, Hatoyama, Saitama, 350-03, Japan

Acquiring search control knowledge of high utility is essential to reasoners in speeding up their problem-solving performance. In the domain of geometry problem-solving, the role of “perceptual chunks”, an assembly of diagram elements many problems share in common, in effectively guiding problem-solving search has been extensively studied, but the issue of learning these chunks from experiences has not been addressed so far. Although the explanation-based learning technique is a typical learner for search control knowledge, the *goal-orientedness* of its chunking criterion leads to produce such search control knowledge that can only be used for directly accomplishing a target-concept, which is totally different from what perceptual-chunks are for. This paper addresses the issues of acquiring domain-specific perceptual-chunks and demonstrating the utility of acquired chunks. The proposed technique is that the learner acquires, for each control decision node in the problem-solving traces, a chunk which is an assembly of diagram elements *that can be visually recognizable and grouped together* with the control decision node. *Recognition rules* implement this chunking criterion in the learning system PCLEARN. We show the feasibility of the proposed technique by investigating the cost-effective utility of the learned perceptual chunks in the geometry domain, and also discuss the potential for the technique being applied to other domains.

1. Introduction

Acquiring search control knowledge (SCK) of high utility is essential to reasoners in speeding up their problem-solving performance. We have been investigating in the domain of geometry problem-solving what kind of SCK¹ should be learned and

how it should be learned from the problem-solving episode of a problem.

Let us look at the two problems in Fig. 1 to make the research issues of this paper clear. The first problem is to prove that $\angle BAC = \angle DEC$, when the conditions of Fig. 1 are given. We can solve it only after coming up with appropriate auxiliary-lines, CF and DF with a new point F such that A, C, F are on a line and $AC = CF$, because these additional lines enable us to use the midpoint condition $BC = CD$ as one of the antecedent conditions for proving congruence of triangles and consequently it prompts us to use the given condition $AB = DE$ to prove $\triangle DEF$ is an isosceles. The second problem, the more difficult one, is to prove $\triangle AQR$ is an isosceles. We can solve it similarly if we come up with auxiliary-lines like APE , CE and DE with a new point E such that A, P, E are on a line and $AP = PE$, because those lines enable us to use the two given conditions, $BP = PC$ and $AB = CD$ to prove that $\triangle ABP$ and $\triangle ECP$ are congruent and $\triangle CDE$ is an isosceles.

The important thing is that although both problems have quite different goal-structures from each other, they need to be solved by utilizing some of the given conditions in exactly the same way with similar auxiliary-lines. In other words, if such SCK as shown in the upper left of Fig. 4 is provided to a reasoner, it can solve both problems in an analogous way as human experts do; the SCK tells that if there is a midpoint condition $XY = YZ$ with X, Y, Z being on a line, then recall (or find) another col-linear line VYW passing through Y such that $VY = YW$ in order to prove that $\triangle XYW$ is congruent to $\triangle ZYV$. Preferably selecting to apply this SCK in reaction to the above midpoint condition, if all the other antecedent conditions of the SCK are satisfied, can lead the reasoner to the solutions in both problems.

This sort of SCK corresponds to what is called “perceptual-chunks” in the research field of cognitive science where the role of perceptual-chunks in effectively guiding problem-solving has been studied [6,7,9,14]. A perceptual-chunk is a chunk of

¹Geometry proof problem-solving has been studied by many researchers [3,4]. Its characteristic is that not only goal-oriented backward reasoning but also bottom-up forward reasoning is essential to constructing a proof-tree efficiently. In this paper, we focus only on such SCK that helps the solver direct forward reasoning.

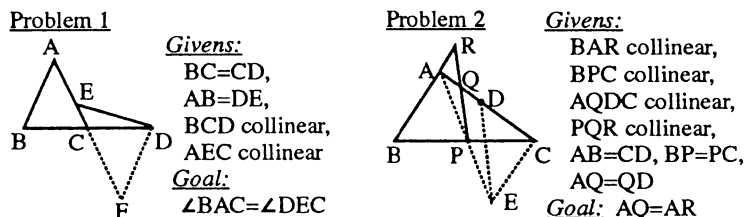


Fig. 1. Two auxiliary-line problems in geometry. Note that the two problems have something in common about how to draw additional lines and how to direct problem-solving search, although they have quite different goal-structures.

diagram elements which many problems share in common as a portion of the whole diagram. Recalling an appropriate perceptual-chunk at a control decision node during problem-solving processes and applying the macro-operator attached to the perceptual-chunk contributes much to controlling problem-solving search effectively. The most important characteristic of search-control by perceptual-chunks is that it is *not goal-oriented*; applying a perceptual-chunk is intended for locally recognizing it in the diagram of the problem, *without caring* whether or not its application will directly accomplish the goal of the problem. In spite of the beneficial role of perceptual-chunks, however, the past work has not addressed the issue of learning useful perceptual-chunks from experiences.

The explanation-based learning (EBL) module of PRODIGY [11] is a typical system that learns SCK from experiences. It explains why a selection at a control decision node has led the reasoner to a target concept and chunks only the features relevant to the explanation as the antecedent conditions of the learned SCK. The *goal-orientedness* of its chunking criterion produces the kind of SCK that is used for directly accomplishing a target concept, which is totally different from perceptual-chunks. In this sense, we cannot expect EBL systems to learn useful perceptual-chunks [15].

This paper addresses the issue of learning perceptual-chunks from an experience. Its basic concept is to chunk the *diagram elements that can be visually recognizable and grouped together* with the diagram elements corresponding to each control decision node of the problem-solving traces. "*Recognition rules*" implement this chunking criterion in the learning system PCLEARN (*Perceptual Chunk LEARNer*). Each of them is domain-specific knowledge which describes the necessary condi-

tions for each domain object to be visually recognizable, representing semantically how we human beings see diagram elements. Notice that the chunked area does not always include the goal of the problem because the goal is not always recognizable (i.e., too far away) at the local control decision node. This distinguishes the PCLEARN system from the EBL learner which chunks all the paths to the target concept of the problem, in most cases the goal of the problem.

In the second section, we show the basic notion of recognition rules and its use for determining the area to be chunked out. In the third section, we discuss the feasibility of the recognition rules as a perceptual criterion, by presenting some experimental data on the operability and cost-effective utility of the learned perceptual-chunks when the solver uses them as search control knowledge. In the fourth section, we discuss the thrust of this perceptual-chunking method in terms of comparisons with other methods as well as its applicability to other domains. Further, we discuss how the structure of "*recognition rules*" is designed so as to overcome the tacitness of the knowledge to be learned, which is one of the key issues in the research field of knowledge acquisition.

2. The PCLEARN Chunking Module

2.1. Recognition Rules

PCLEARN, after solving a problem², learns a perceptual chunk with macro-operator information

²In case of auxiliary-line problems as shown in Fig. 1, PCLEARN needs to be taught about how to draw lines in order to solve them. Note that once it learns a chunk, the chunk can be used to find auxiliary lines.

recognizable(X):- recognizable(s(X,Y)).
recognizable(s(X,Y):- recognizable(a(X,Y,Z)).
recognizable(s(X,Y):- recognizable(tr(X,Y,Z)).
recognizable(s(X,Y):- recognizable(X), recognizable(Y), exist(s(X,Y)).
recognizable(s(X,Y):- recognizable(X), recognizable(Y), collinear(X,Z,Y).
recognizable(a(X,Y,Z):- recognizable(s(X,Y)), recognizable(s(Y,Z)).
recognizable(tr(X,Y,Z):-
recognizable(s(X,Y)), recognizable(s(Y,Z)), recognizable(s(Z,X)).
where s(X,Y) -- segment XY, tr(X,Y,Z) -- triangle XYZ, a(X,Y,Z) -- angle XYZ
The literals underlined are additional conditions.

Fig. 2. The set of recognition rules in geometry. Each rule describes the necessary conditions for a domain object to be “visually recognizable”. These rules are as a whole a semantic representation of how we human beings see domain objects.

for each control decision node³ of the given problem-solving traces. For the purpose of chunking diagram elements that can be visually recognizable and grouped together for each control decision node, we need to provide a criterion for determining what is “visually recognizable”. The use of recognition rules determines it in PCLEARN.

Recognition rules themselves are a semantic representation of how human experts visualize domain objects. More precisely speaking, each rule itself describes the necessary conditions for a domain object to be “visually recognizable”, consisting of the recognizability of other related domain objects and some additional conditions.

Fig. 2 is the set of recognition rules we provided in the domain of geometry. Points, segments, triangles and angles are the domain objects in this domain. The first rule states that a point X is always recognizable when a segment XY is found recognizable because X is a constituting member of XY . In general, when an object is already found recognizable and we want to prove the recognizability of another object which is a structurally constituting member of the former object, we do not need any additional conditions. The first three rules in Fig. 2 belong to this category. On the other hand, when we prove the recognizability of an object from the other objects which structurally compose that target object, we need some (sometimes no) additional conditions. For example, when we prove the recognizability of segment XY from the recogni-

zabilities of the two points X and Y , an additional condition is needed, i.e., the segment XY actually has to exist in the problem space (corresponding to $exist(s(X,Y))$ in Fig. 2), or two segments $s(X,Z)$ and $s(Z,Y)$ have to be on the same line for another point Z (corresponding to $collinear(X,Z,Y)$ in Fig. 2). The last two recognition rules are the examples where no additional condition is needed by chance, although they belong to this category.

2.2. Chunking by Use of Recognition Rules

For the purpose of chunking perceptual-chunks for each control decision node, PCLEARN first (1) identifies all the recognizable domain objects when the objects included in the domain rule⁴ that has been successfully applied to the control decision node are supposed to be recognizable, and (2) enumerates all the recognizable features of the above objects, and (3) finally obtains a perceptual chunk which is the assembly of the recognizable objects with the recognizable features.

2.2.1. Step 1: Picking up Recognizable Objects

The first step is to enumerate all the recognizable objects relevant to a control decision node. The procedures are

1. to assert that all the objects which appear as the arguments of the literals in the SAR are recognizable, and
2. to enumerate all the objects which can be proved as recognizable, using *recognition rules*.

³A control decision node is a node which is a member of the successful proof tree with more than one tested domain rules, some of which have been successfully applied.

⁴This rule is denoted as SAR (Successfully Applied Rule) in this paper.

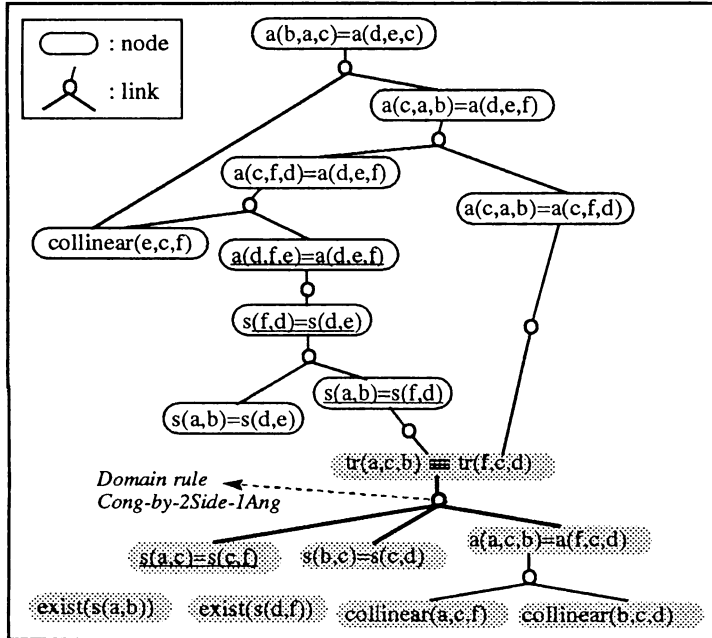


Fig. 3. The successful proof tree of Problem 1 of Fig. 1. A proof tree consists of nodes and links; nodes are statements including the given conditions and the features that have been produced as a result of inference, and each link connecting some nodes to an upper node represents the domain rule that has been used for deriving the upper node.

Fig. 3 is a successful proof tree of the Problem 1 in Fig. 1. The underlined nodes are the control decision nodes. Here, the learning process for the control decision node, $AC=CF$, is illustrated. The SAR for this control decision node is *Cong-by-2Side-1Ang*. First, the objects appearing in this SAR, $s(b,c)$, $s(a,c)$, $s(c,d)$, $s(f,c)$, $a(b,c,a)$, $a(d,c,f)$, $tr(a,b,c)$ and $tr(c,d)$, are asserted to be recognizable⁵. Then, by use of the recognition rules, the following objects, a , b , c , d , f , $s(a,b)$, $s(d,f)$, $s(b,d)$, $s(a,f)$, $a(b,a,c)$, $a(b,a,f)$, $a(a,b,c)$, $a(a,b,d)$, $a(d,f,c)$, $a(d,f,a)$, $a(f,d,c)$, $a(f,d,b)$, $a(b,c,f)$ and $a(a,c,d)$ are justified to be recognizable⁶.

2.2.2. Step 2: Enumerating Recognizable Features

The second step is to derive and pick up from the problem-solving traces all the recognizable fea-

tures of the above recognizable objects. The procedures are

1. the literals appearing in the SAR are picked up as recognizable,
2. the literals of the *additional conditions* which appeared in the recognition rules used successfully for proving the recognizability of objects in Step 1 are picked up as recognizable, and
3. all the features that can be derived from the above recognizable literals using domain rules are regarded as recognizable.

After the third step of Step 2, we get a derivation tree consisting of the enumerated recognizable features. Importantly, the derivation tree itself represents a piece of macro-operator information that can be applied to the same control decision node in future problems; the lowest nodes of the tree are the IF-part of the macro-operator and the other nodes are the THEN-part. If we notice that the macro-operator has been derived only from the recognizable features that have been determined by use of recognition rules, the significant role of recognition rules in chunking the macro-operator may be clear.

⁵These are the core objects from which all the recognizable objects will be enumerated using recognition rules.

⁶There is a heuristic in using recognition rules; it is to first use the rules of the first category mentioned in Section 2.1 to prove the recognizability of the more basic and lower objects in terms of object hierarchy, and then to use the rules of the second category to enumerate the recognizability of the upper objects.

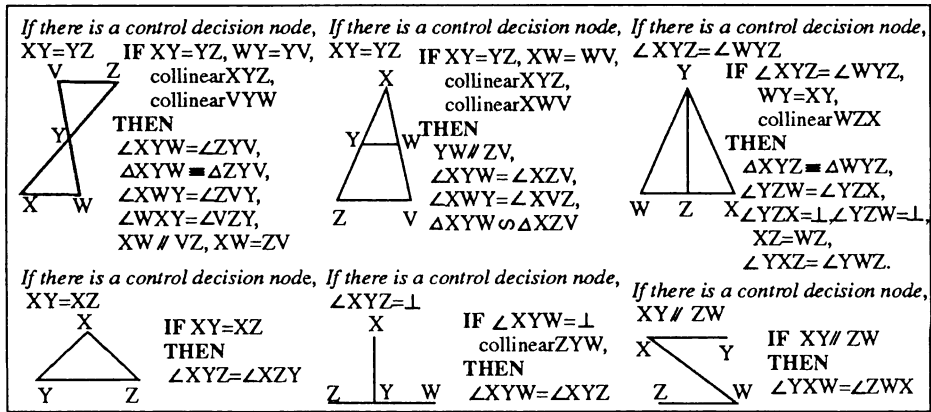


Fig. 4. The perceptual-chunks learned by PCLEARN and their macro-operator information. Each one includes information about (1) the control decision node to which the macro-operator should be applied, (2) the set of antecedent conditions and (3) the statements that will be produced when successfully applied.

Let us look at the example case of learning from the control decision node $AC=CF$ in Fig. 3. The recognizable literals to be picked up before the derivation process are shown in Fig. 3 as the nodes colored grey, out of which the literals that have been incorporated as a result of using recognition rules (the 2nd of Step 2) are $collinear(a,c,f)$, $collinear(b,c,d)$, $exist(s(a,b))$, $exist(s(d,f))$. The first two, $collinear(a,c,f)$ and $collinear(b,c,d)$, have been picked up because they appeared in the recognition rules used for proving the recognizability of the object $s(a,f)$ and $s(b,d)$ respectively. Out of these four, the last two will not be used in the derivation process and therefore removed from the macro-operator.

Notice here that due to the existence of some additional conditions in the set of recognition rules, the learned macro-operator becomes more specific⁷ than the SAR itself. In case of the above example, incorporating the two features of collinearity has been significant in obtaining a perceptual-chunk of the two congruent triangles located in a completely point-symmetry (the upper left of Fig. 4), which is more specific than the two merely congruent triangles.

2.2.3. Step 3: Generalizing

The final step is to generalize each node of the acquired derivation tree by dissolving the bindings of the variables of the used domain rules. The generalized tree itself represents a macro-operator that has been learned for the control decision node. In case of the above example, the one in the upper left of Fig. 4 is acquired.

We call this sort of macro-operator a perceptually-chunked macro-operator because the recognition rules work as a perceptual criterion for determining the area to be chunked out, just as human experts might do visually. Also in this sense, we can say that chunking by use of recognition rules has a connotation of visually controlling learning processes.

3. Experimental Results

In this section, we describe some experiments for showing the feasibility of PCLEARN perceptual-chunking in comparison with the EBL system that learns SCK.

The typical EBL system that learns SCK is PRODIGY [11]. It implements selective learning by providing four kinds of meta-level target-concepts, "succeeds", "fails", "sole-alternative", and "goal-interference". However, in the domain of geometry problem-solving, learning from "fails", "sole-alternative", and "goal-interference" will not lead to useful knowledge, because in this domain there

⁷This specificity directly influences much the operatinality of the learned perceptual-chunks. In this sense, recognition rules play a crucial role in determining the chunked area.

Table 1
The frequencies of macro-operators being learned from the 20 different problems, for both learners. The number of those macro-operators which have been learned more than once is an index showing the capability of the learner's acquiring general knowledge

Frequency	The number of macro-operators	
	PCLEARN	EBL
1	43	86
2	9	7
3	5	0
4	3	1
more than 4	4	0
total	64	94

may be no positive reason why a choice leads to a failure, and there may be no problem-solving phenomenon corresponding to sole-alternative and goal-interference. Therefore, we compare the PCLEARN system with the EBL system that learns from "succeeds" in which the goal-node of the problem is specified as the target-concept and the learner acquires a so-called *preference rule* by explaining why the selection of a domain rule at each control decision node contributed to achieving the goal.

Two experiments have been done for investigating generality and utility of the learned knowledge.

3.1. Generality of the Learned Knowledge

Whether or not the learned macro-operator represents a general and meaningful perceptual chunk depends upon how frequently it is learned from various problems. We carried out an experiment in which each of the 20 geometry problems (shown in Appendix A) were separately solved and learned by both learners i.e., the PCLEARN and EBL systems.

Table 1 shows how many times the same macro-operator has been learned from the 20 problems for both learners. PCLEARN learns the same macros much more frequently (i.e., more than twice), while it is quite rare that the EBL learner acquires the same macro more than twice. This is because the macro-operators learned by EBL tend to be more specific to the goal-structure of the original problems. It suggests that *goal-orientedness* of the EBL chunking criterion does not suit the nature of

the geometry domain where there is little consistency in goal-structure across problems but rather more consistency in perceptual chunks across problems. In this respect, PCLEARN is superior to EBL systems as a method of learning domain-specific perceptual chunks in this domain.

3.2. Cost-Effective Utility of the Learned Knowledge

In order to evaluate the feasibility of PCLEARN as a learner for speeding up problem-solving performance, we evaluated the cost-effective utility of each of the learned macro-operators throughout its use over many problems according to the following formula,

$$Utility = TotalSavings - TotalMatchCosts. \quad (1)$$

where *TotalSavings* is the cumulative cpu-time benefit that results from applying the macro-operator frequently, so-called *re-ordering effect* [10], and *TotalMatchCosts* is the cumulative time cost spent in testing to apply⁸ the macro-operator in vain over frequent testings during many problems. Every time a problem is solved, the above costs and benefits are measured for each macro-operator, as a result of which the utility of each macro-operator is updated.

The cpu-time benefit of a macro-operator is calculated, when it was applicable to a problem, by subtracting the costs relevant to the use of the macro-operator from the corresponding costs in solving the same problem without any macro-operators⁹. The former costs tend to become large by the existence of macro-operators because they potentially cause the solver to try more matchings in searching for applicable knowledge, while the latter may include the costs spent in vain by producing some irrelevant branches of nodes at the decision node before applying the successful domain rule, which the solver would have avoided by use

⁸The solver first tries to use the available macro-operators, and after finding it impossible to apply any of them the solver turns to use the domain knowledge. During this process also, the matching costs in "testing to apply" the macro-operators are surely spent. So *TotalMatchCosts* includes all this sort of matching costs for each macro-operator.

⁹The results of solving all the problems without macro-operators are provided in advance.

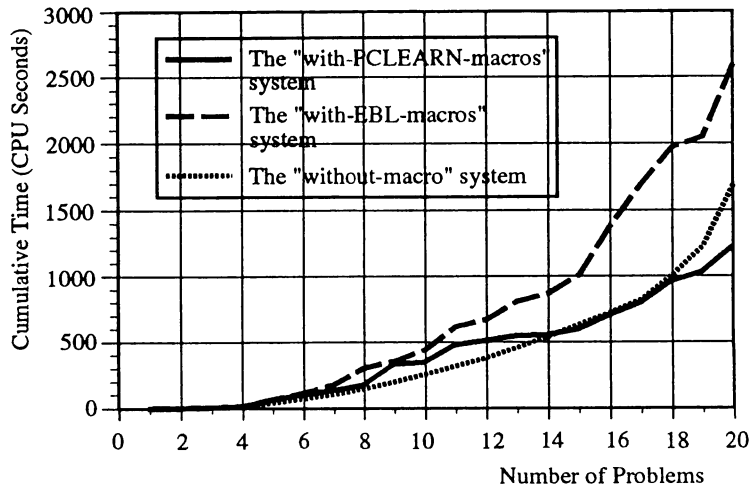


Fig. 5. The learning curves obtained for PCLEARN, the EBL learner and the solver that does not use any learned knowledge. Since the vertical axis indicates cumulative costs, the more gradual increase of learning curves means that there are some speed-up effects due to using learned knowledge.

Table 2
The costs of testing to apply learned knowledge during solving 20 problems, in both learners

	Total (msec)	Frequency of testing	Cost per one testing(msec)
PCLEARN	288,733	1,146	252
EBL	976,963	1,910	511

of the macro-operator. The tradeoff between these two effects influences the benefit of each macro-operator.

We do not separate a training session from a testing one. Instead, going through the 20 problems, each one is solved using only a limited number¹⁰ of macro-operators that have been already learned and have scored high utility values *up to* that point. The problem order is assigned so that the problems become progressively larger.

In Fig. 5, the cumulative problem-solving costs are plotted against the number of problems solved up to that point, for both cases of using PCLEARN macro-operators (“with-PCLEARN-macro” sys-

Table 3
The frequencies of learned knowledge being applied to other problems, and its percentage against all the frequencies of testing to apply them

	Frequency of testing	Frequency of applying	Percentage (%)
PCLEARN	1,146	39	3.4
EBL	1,910	12	0.6

Table 4
The average and standard deviation of the sizes of learned knowledge; comparison the case of calculating over the applied knowledge with the case of calculating over all the learned knowledge

	All macros		Applied macros	
	Average	Standard Deviation	Average	Standard Deviation
PCLEARN	3.3	1.4	2.7	1.3
EBL	5.0	1.8	3.2	1.2

tem) and the EBL macro-operators (“with-EBL-macro” system), together with the case of the system without any macro-operators (“without-macros” system) for reference. The costs of the “with-PCLEARN-macro” system become worse for the

¹⁰In this experiment, the currently best 25 macro-operators in terms of utility values are selected as available for the next problem, every time each problem is solved.

first several problems than the “without-macro” system due to the increased matchings by use of the macro-operators, but after going through eleven problems, the slope of the cumulative costs gradually begins to become smaller than that of the “without-macro” system, i.e., learning effects by use of macro-operators begin to appear. On the other hand, the costs of the “with-EBL-macro” system are larger by the costs of using macro-operators than that of the “without-macros” system, which indicates that only a few macro-operators were useful out of the learned ones.

The results of Tables 2, 3 and 4 can be an explanation of why the EBL macro-operators can exhibit little learning effect in the geometry domain. Table 2 shows, for both the PCLEARN and EBL macro-operators, the total costs of testing to apply macro-operators at control choice nodes. The average costs per one testing of the EBL macro-operators are about double the costs per one testing of the PCLEARN macro-operators. Table 3 shows the applicability of both macro-operators. The PCLEARN macro-operators scored a higher percentage of successful applications, i.e., higher applicability, in this domain than the EBL macro-operators. The lower applicability of the EBL macro-operators may be an explanation of their lower learning effect. Table 4 shows why the EBL macro-operators cost much more in testing and have lower applicability. The average sizes¹¹ and the standard deviations of the applied macro-operators weighed with the frequency of applications are compared with those of all the learned macro-operators. For the PCLEARN macro-operators, the distributions of the sizes of the learned macro-operators and the applied ones are not very different from each other, which tells that PCLEARN produces macro-operators with appropriate size in terms of applicability. On the other hand, however, for the EBL macro-operators the distribution of the sizes of the learned macro-operators is extremely shifted towards larger size than that of the applied ones, which indicates that the EBL method tends to produce too large macro-operators in terms of applicability.

To sum up the above observations, the principle

¹¹For simplicity, we define that the size of a macro-operator is equal to the number of its preconditions, reflecting the ease of finding appropriate instantiations of the preconditions.

of EBL techniques to chunk the path from a control choice node to the goal of the problem tends to make the size of the learned knowledge too large, causing too many costs in using it, and also tends to make the learned knowledge too specific to the goal-structure of the problem, causing low applicability and therefore little speed-up effect in future problems. On the contrary, use of the “recognition rules” in PCLEARN is effective in the domain of geometry problem-solving in the sense that it contributes much to acquiring macro-operators with appropriate size and better cost-effective performance. Fig. 4 shows the set of perceptually-chunked macro-operators which have scored high utility values after going through all the problems.

4. Discussion

4.1. Difference from the Operability Criterion

Since the PCLEARN system chunks a partial structure of the problem-solving traces, some readers may associate its chunking criterion with the concept of the operability criterion [12] that has been discussed with EBL. Here we have to discuss that both have different connotations from each other; both ideas come from different purposes for chunking, different semantics, and the different ways in using chunking criteria.

First, the purpose of PCLEARN chunking is to use the learned knowledge for parsing the whole problem space into some meaningful subparts, which helps a solver avoid traversing irrelevant paths of inference as shown in Koedinger’s CD model [7]. On the other hand, the purpose of learning by use of the operability criterion is to acquire operational and usable knowledge, e.g., in recognition tasks.

This difference in purpose brings about the difference of semantics; the semantic of PCLEARN chunking is to chunk out what is “visually grouped” in the light of human being’s way of visualizing domain objects, while the semantic of learning by use of the operability criterion is to chunk out such a portion of the obtained explanation tree that is “described only by operational predicates (i.e., descriptive predicates) and not by functional predicates”, in recognition tasks. Here one may regard what is represented by descriptive predicates as equal to what is chunked by use of

recognition rules, and also think that PCLEARN is an extension of operability criterion. But it is not correct; distinguishing functional and descriptive concepts (e.g., “liftable” vs. “with-hand” in the example of CUP [16]) is one thing and grouping the represented concepts into some chunks visually is another. The difference between both learners may not yet be clear enough in the above example, because the concept of a single cup is not so complicated as to make humans feel like parsing it into sub-concepts. But suppose an object with more complicated configurations which a human reasoner tries to parse into smaller parts when he thinks of the object. In this example, the explanation tree of the whole object would be represented by many predicates, some of which are functional and others descriptive. Here, the most important thing is to parse the whole object into several subparts from a viewpoint of “visual grouping”, not from a viewpoint of differentiating between functional and descriptive predicates. In this example, it is clear that the criteria of both chunking methods will bring about different learning results.

In order to implement the above purposes and semantics of learning, both chunking methods employ different criteria in different ways. EBL systems are provided in advance with a list of operational (descriptive) predicates that will work in target tasks, and chunks such a portion of the obtained explanation tree that is represented by those operational predicates. However, we cannot determine in advance what predicates in the explanation tree are visually grouped at each of the local control decision nodes, i.e., in other words, it must be dependent on each control decision node. Thus, the PCLEARN system dynamically determines it at each control decision node, using knowledge about how humans visualize each domain object in relation to other objects.

4.2. Comparison with Other Chunking Methods

Other kinds of chunking methods to be mentioned are SOAR [8] and ACT theory [1], in which the learners chunk the problem-solving processes relevant to satisfying the subgoals the solvers have established. We call these methods, including EBL, *goal-oriented* chunking because they share the view that the learner should chunk the goal-oriented problem-solving processes toward accomplishing subgoals and/or target-concepts. Com-

pared to these works, the PCLEARN’s chunking criterion is new in the sense that it is not *goal-oriented* but rather a bottom-up perception-oriented chunking mechanism. In this respect, we call it *perceptual* chunking.

4.3. Applicability of Perceptual Chunking to Other Domains

The characterization of the PCLEARN system is to acquire a useful set of perceptual chunks which help the solver parse problem diagrams into several subparts, contributing much to focusing only on some key inference steps to generate a solution plan. Geometry is a class of problem domains in which such inference is crucially required.

The perceptual-chunking technique can be straightforwardly applied to those domains in which proof or diagnosis is performed concerning a complex structured object that consists of several unit objects, because

- there are a lot of *part-whole* relationships in that sort of structured objects, and
- the *recognition rules* relate one object to another (or other objects) in terms of recognizability, when the former object has a part-whole relation to the latter. For example, the functor s in Fig. 1 represents a part-whole relation between a segment XY and the constituent points X and Y .

In the above characterization, we exclude such a class of domains in which problem-solving operators are applied (1) to add or delete something to and from the current state of the target object or (2) to transform the structure/shape of the target object, like the domain of designing; in such a domain part-whole relationships appear and/or disappear as inference goes on, until a goal state of the target structured object is obtained. It is an open problem to be investigated whether recognition rules can be described in advance in an environment where part-whole relations are not fixed, and whether, even if they can be, they may work well to parsing all inference steps into subparts.

4.4. The Structure of Recognition Rules

Perceptual chunks are a kind of tacit knowledge. Anderson [2] argues that the knowledge underlying expertise is often tacit due to the process of

knowledge compilation; as experts learn problem-solving strategies from experience in a domain, they tend to become unaware of the individual inference steps for deriving the useful associations between situations and actions which they were aware of when they were novices. Perceptual chunks are the product of grouping some diagram constructs together via *visual scanning on problem diagrams* [7,9], accompanied by applications of problem-solving operators. They are tacit in two ways; firstly even experts cannot articulate why a perceptual chunk is useful because they have almost forgotten the problem situations from which they acquired it in the past as well as the individual steps of visual grouping which they did to acquire it. Secondly, more importantly, experts cannot easily articulate what kind of perceptual chunks they have if they are asked independently of problems, because they can associate the chunks only with some problem elements they are faced with.

We will discuss how the PCLEARN system addresses the issue of acquiring perceptual-chunks which are tacit in the above senses, from a view point of relationships between knowledge acquisition and machine learning. The PCLEARN system basically employs machine learning techniques of automating the process of acquiring chunks from experiences. But the automated process is controlled by the knowledge that governs how we human beings recognize diagram elements in problems, and the issue on how to elicit the knowledge itself from experts belongs to the field of knowledge acquisition.

However, the knowledge about how to recognize diagrams is also too tacit for experts to easily articulate. The commonly used technique in knowledge acquisition is to explicitly design the structure of the knowledge we want to elicit so that experts may have only to "fill in" the structure [13] in articulating knowledge. We designed the structure of the knowledge about how to recognize diagrams such that it is represented as a relation among recognizability of discrete diagram units that have a part-whole relation to each other (i.e., domain objects); experts have only to articulate the necessary conditions for a diagram unit to be recognizable, independently of problem-solving processes and it will work well as a chunking criterion in learning. Although the recognition rules in the geometry domain are comparably easy to be elicited, the proposed structure is general enough to be ap-

plied to other domains and thus may work well as a *medium of articulating*¹² a plausible set of recognition rules in those domains.

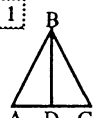

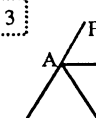

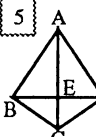

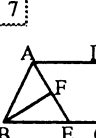

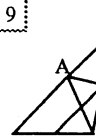
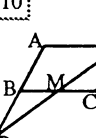
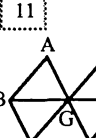
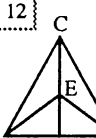

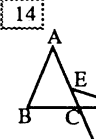
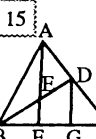


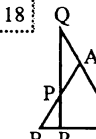
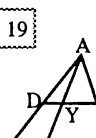
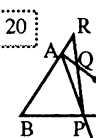
5. Conclusion

We proposed a new mechanism of learning domain-specific perceptual-chunks from experience that can be used as search control knowledge. Its basic concept is that the learner acquires, for each control decision node in the problem-solving traces, a chunk which is an assembly of diagram elements *that can be visually recognizable and grouped together* with the control decision node. Its chunking criterion is quite different from that of the other learners that chunk the goal-oriented problem-solving processes of the solver, e.g., explanation-based learners. *Recognition rules*, domain-specific knowledge relating one object to other objects in terms of "recognizability" when the former has a *part-whole* relation to the latter, implement this chunking criterion in PCLEARN. The set of rules can be regarded as a *semantic representation of how human beings see objects* and its effective use as a guide of learning processes is demonstrated here. The use of the perceptual-chunks obtained *semantically* guides problem-solving processes toward relevant paths of inference just as human beings do visually, although PCLEARN syntactically does sentential searches in logic for applicable perceptual-chunks.

In the domain of geometry problem-solving, a typical domain where diagrammatic reasoning is effective, we made some experiments of measuring the utility of the learned perceptual chunks and showed that the chunks learned by PCLEARN can work as search control knowledge with higher applicability and better cost-effective utility, compared to conventional learners like EBL systems. The results suggest that the chunking criteria proposed are acceptable and useful in this domain.

The results of this paper suggest that the basic notion of the PCLEARN system is general enough to be applied on a large scale to those domains in which proof or diagnosis is performed concerning a structured object that consists of some unit objects.

¹²We used this terminology in the same sense as Gruber does in his book [5, p. 232].

<p>1</p>  <p>Givens: ADC collinear $\angle ADB = \perp$ $\angle ABD = \angle CBD$</p> <p>Goal: $AD = CD$</p>	<p>2</p>  <p>Givens: BDC collinear $AB = AC$ $\angle BAD = \angle CAD$</p> <p>Goal: $BD = CD$</p>	<p>3</p>  <p>Givens: $AB = AC$ $AE \parallel BC$ BAF collinear</p> <p>Goal: $\angle FAE = \angle CAE$</p>
<p>4</p>  <p>Givens: $AB = AC$ $\angle ABD = \angle CBD$ $\angle ACE = \angle BCE$ AEB collinear ADC collinear</p> <p>Goal: $BD = CE$</p>	<p>5</p>  <p>Givens: AEC collinear, BED collinear, $AB = AD$ $BC = DC$</p> <p>Goal: $\angle BEA = \perp$</p>	<p>6</p>  <p>Givens: BDHC collinear, $DH = HC$ $\angle DHA = \perp$ $\angle DBA = \angle DAB$</p> <p>Goal: $BD = AC$</p>
<p>7</p>  <p>Givens: BEC collinear, AFE collinear, $AD \parallel BC$ $\angle ABF = \angle EBF$ $\angle BAE = \angle DAE$</p> <p>Goal: $\angle BFA = \perp$</p>	<p>8</p>  <p>Givens: AFC collinear, BGD collinear, $AF = FC$, $BG = GD$ $AB = CD$ $\angle AFE = \perp$, $\angle BGE = \perp$</p> <p>Goal: $\triangle ABE \cong \triangle CDE$</p>	<p>9</p>  <p>Givens: $BD = DC$ $AF = AC$ $DE \parallel BF$ BAF collinear, BDC collinear, CEF collinear</p> <p>Goal: $\angle AEC = \perp$</p>
<p>10</p>  <p>Givens: $AB \parallel CD$, $AD \parallel BC$ $BM = MC$ ABP collinear, PMD collinear, BMC collinear</p> <p>Goal: $AB = BP$</p>	<p>11</p>  <p>Givens: $AG = GD$, $BG = GE$ $CG = GF$ AGD collinear, BGE collinear, CGF collinear</p> <p>Goal: $\angle ABC = \angle DEF$</p>	<p>12</p>  <p>Givens: ADB collinear, CED collinear, $AC = BC$, $AE = BE$</p> <p>Goal: $\angle ADC = \perp$</p>
<p>13</p>  <p>Givens: AEB collinear, MADC collinear, BDM collinear, CEN collinear, $AE = EB$, $AD = DC$ $BD = DM$, $CE = EN$</p> <p>Goal: $AN = AM$</p>	<p>14</p>  <p>Givens: $BC = CD$, $AC = CF$ $AB = DE$ BCD collinear, ACF collinear</p> <p>Goal: $\angle BAC = \angle DEC$</p>	<p>15</p>  <p>Givens: ADC collinear, AEF collinear, BED collinear, BFGC collinear, $AD = DC$, $BE = ED$ $AF \parallel DG$</p> <p>Goal: $BF = CG$</p>
<p>16</p>  <p>Givens: BNMC collinear, AEN collinear, ADM collinear, $\angle ABD = \angle CBD$ $\angle ACE = \angle BCE$ $\angle ADB = \perp$ $\angle AEC = \perp$</p> <p>Goal: $DE \parallel MN$</p>	<p>17</p>  <p>Givens: ADB collinear, ACE collinear, DME collinear, BFMC collinear, $BD = CE$, $CM = MF$ $DM = ME$</p> <p>Goal: $AB = AC$</p>	<p>18</p>  <p>Givens: RPQ collinear, APB collinear, CAQ collinear, BRC collinear, $AB = AC$ $\angle BRQ = \perp$</p> <p>Goal: $AP = AQ$</p>
<p>19</p>  <p>Givens: AEC collinear, ADB collinear, DYE collinear, BXC collinear, AYX collinear, $AD = DB$, $AE = EC$</p> <p>Goal: $AY = YX$</p>	<p>20</p>  <p>Givens: BAR collinear, BPC collinear, APE collinear, AQDC collinear, PQR collinear, $AB = CD$, $BP = PC$ $AP = PE$, $AQ = QD$</p> <p>Goal: $AQ = AR$</p>	

Appendix A: Geometry Problems for Experiments

“Givens” are the problem statements provided as given conditions and “Goal” is a statement to be proved when “Givens” exist in the target structured object whose diagrammatic information is shown.

References

- [1] Anderson, J.R. (1983): *The Architecture of Cognition*, Harvard University Press, Massachusetts, London, England.
- [2] Anderson, J.R. (1986): Knowledge Compilation: The General Learning Mechanism. *Machine Learning: An Artificial Intelligence Approach*, Volume 2. Morgan Kaufmann.
- [3] Gelernter, H. (1963): Realization of a Geometry-Theorem Proving Machine. *Computers and Thought*. MacGraw-Hill Book Company.

- [4] Greeno, J.G. (1983): Forms of understanding in mathematical problem-solving. *Learning and Motivation in the Classroom*. Erlbaum.
- [5] Gruber, T.R. (1989): *The Acquisition of Strategic Knowledge*. Academic Press (Perspectives in Artificial Intelligence series).
- [6] Kim, M.Y. (1989): Visual Reasoning in Geometry Theorem Proving. *Proceedings of IJCAI-89*, 1617–1622.
- [7] Koedinger, K.R. and Anderson, J.R. (1990): Abstract planning and perceptual chunks: elements of expertise in geometry. *Cognitive Science* 14: 511–550.
- [8] Laird, J.E., Newell, A. and Rosenbloom, P.S. (1987): SOAR: an architecture for general intelligence, *Artificial Intelligence* 33: 1–64.
- [9] McDougal, T. and Hammond, K. (1992): A recognition model of geometry theorem-proving. *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 106–111.
- [10] Minton, S. (1990): Quantitative results concerning the utility of explanation-based learning, *Artificial Intelligence* 42: 363–391.
- [11] Minton, S., Carbonell, J.G., Knoblock, C.A., Kuokka, D.R., Etzioni, O. and Gil, Y. (1989): Explanation-based learning: a problem solving perspective, *Artificial Intelligence* 40: 63–118.
- [12] Mostow, D.J. (1983): Machine Transformation of Advice into a Heuristic Search Procedure. *Machine Learning: An Artificial Intelligence Approach*. CA: Tioga Press.
- [13] Musen, M.A., Fagan, L.M., Combs, D.M. and Shortliffe, E.H. (1987): Use of a domain model to drive an interactive knowledge-editing tool. *International Journal of Man-Machine Studies*, 26(1): 105–121.
- [14] Suwa, M. and Motoda, H. (1989): Acquisition of associative knowledge by the frustration-based learning method in an auxiliary-line problem, *Knowledge Acquisition* 1: 113–137.
- [15] Suwa, M. and Motoda, H. (1992): Learning perceptually-chunked macro-operators. Presented at the international workshop on Machine Intelligence, to appear in *Machine Intelligence*, Volume 1, Oxford University Press.
- [16] Winston, P.H., Binford, T.O., Katz, B. and Lowry, M. (1983): Learning Physical Descriptions from Functional Definitions, Examples and Precedents. *Proceedings of AAAI-83*, 433–439.