Machine Intelligence
Vol. 13

16

Learning Perceptually Chunked Macro Operators

M. Suwa and H. Motoda

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

Abstract

In previous studies search control knowledge has been acquired using explanation-based learning (EBL) techniques. These learn *goal-oriented* control knowledge by explaining how a decision at a control decision node leads to the goal. In the domain of geometry problem-solving, however, this leads to knowledge which is neither sufficiently general nor sufficiently operational. This paper addresses an alternative form of search control knowledge in which the search is controlled at each decision node in such a way that a problem solver can locally recognize relevant 'perceptual chunks'. Previously the effectiveness of perceptual chunks as control knowledge has been reported in a geometry domain. In this paper, we propose a new chunking technique, which acquires, as a chunk, an assembly of diagram elements that can be *recognized and grouped together with the control decision node.* In order to implement this chunking criterion a learner, PCLEARN, employs *recognition rules*, domain-specific knowledge describing necessary conditions for a domain object to be recognizable. Experiments in a geometry domain show that the set of learned knowledge exhibits higher operationality than EBL macro-operators. They also suggest that the PCLEARN chunking technique can be a powerful method for obtaining a small and highly organized set of domain-specific

perceptual chunks if augmented with a mechanism for dynamically managing the utility of each chunk.

1 INTRODUCTION

The ability to learn search control knowledge is critically important for problem solvers due to the exponential growth in size of the search spaces they confront. It has been shown that explanation-based learning (EBL), including macro-operator learning by simple EBL techniques (Fikes *et al.* 1972; Minton 1985) and a more sophisticated one that actively selects what to learn (Minton *et al.* 1989), is a powerful technique for learning search control knowledge. This research shares the common view that learners acquire 'goal-oriented'[1] search control knowledge by explaining why a choice taken at a control decision node eventually satisfies the target concept of the problem.

The objective of this paper is to pose and answer the following question. 'Is there any kind of effective search control knowledge which is not goal-oriented?' In the domain of geometry, one of the classical but typical domains with exponential growth in size of search spaces, it has been reported that use of 'perceptual chunks' in the Diagram Configuration (DC) model (Koedinger and Anderson 1990) drastically reduces search spaces. Perceptual chunks are regarded as search control knowledge which is not aimed at achieving a certain goal/subgoal, but at guiding the search process at a control decision node in such a way that problem solvers can recognize the chunks in the problem space. Suwa and Motoda (1989; 1991) have shown that use of 'figure-pattern strategies', a small set of macro-operators whose figurative patterns are the chunks meaningful in geometry domain, enables problem solvers to intelligently select appropriate construction-lines by adding a new point out of an indefinite number of candidate constructions. This research suggests that perceptual chunks also provide critically important search control knowledge in the geometry domain. Here, we pose a sec-

[1]EBL learns from a target concept and produces search control knowledge that is used for accomplishing a unifiable goal in future problems. In this chapter we say that such control knowledge is 'goal-oriented'.

ond question, 'Can the EBL technique simulate acquisition of perceptual chunks in geometry?' Koedinger (1992) suggested that macro-operator-like perceptual chunks in geometry are not primarily organized around the goal-structure explaining target concepts but, rather, are organized around objects or aggregations of objects in the domain of geometry. We will find an answer to the above questions and justify Koedinger's suggestion.

In this chapter, we will address two issues along the lines of the above questions, by illustrating experimental data in the domain of geometry. The first issue is about operationality of EBL macro-operators in the domain of geometry. It is a critical issue because unless learners provide a mechanism of acquiring a small and highly organized set of macro-operators with high operationality, problem solving performance degrades drastically with increasing numbers of macro-operators (Minton 1984, 1985). Operationality of EBL macro-operators depends upon the intrinsic nature of the geometry domain itself concerning whether there is a consistency in goal-structure across many problems or not, because EBL macro-operators can be applied only to those future problems which include the same goal-structure. It is an open empirical question (Koedinger 1992). We will examine it by collecting experimental data on the frequency at which EBL macro-operators are acquired from and applied to many problems, which is a simple measure of the utility of macro-operators (Minton 1985).

The second issue is the proposal of a new learning technique, which is based on another concept different from that of EBL; the learned concept is to be acquired as a chunk, *an assembly of diagram elements that can be recognizable and grouped together with each control decision node.* The learned chunk can be used as control knowledge which guides problem-solving search so that solvers can locally recognize a perceptual-chunk relevant to each of the control decision nodes. The distinguishing point is that there is no notion corresponding to target concepts of EBL. This requires us to provide a criterion for dynamically determining the range of chunking. Suwa and Motoda (1991) proposed the idea of '*recognition rules*', domain-specific rules describing necessary conditions for a domain object to be recognizable, as a guide to determine the area of problem-solving traces to be chunked out. In this chapter, we present a computer program PCLEARN, a domain-independent system of learning perceptual chunks, by use of recognition rules. We also investigate the utility of the rules using experimental data.

In the second section, we characterize the geometry domain and enumerate the problems in applying EBL to this domain. In the third section, we describe the details of the recognition rules themselves and their use in learning perceptual-chunks. In the fourth section, experimental results in the geometry domain are presented and comparisons are made between EBL and the proposed technique in terms of operationality of the learned knowledge. The current limitations of the PCLEARN system and future research issues are discussed in the fifth section.

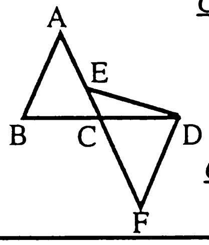
2 LEARNING SEARCH CONTROL KNOWLEDGE IN GEOMETRY

2.1 Geometry domain

A general characterization of geometric problem-solving is that it is the task of proving a fact holding among an assembly of geometrical objects when a set of other facts are known to hold as given conditions. Domain rules are used for deriving new facts from the set of already given facts. Once a fact is derived, it will never be undone in this domain, because it has been already proved to hold in the given environment. Therefore, the number of facts will increase monotonically as the proving process proceeds. As discussed later, these characteristics are major factors in bringing about difficulties in applying EBL in this domain.

The geometrical objects in this domain are points, segments, directed segments, angles, and triangles. A fact is a nature of an object or a relation between geometrical objects, which is represented in this chapter using the predicate symbols *eqs*, *eqa*, *cong*, *sim*, *para*, *collinear*, *exist*. These express equality of the lengths of two segments, equality of the sizes of two angles, congruence of two triangles, similarity of two triangles, the state of two directed segments being parallel, collinearity of two seg-

Problem 1



Givens:
BC=CD, AC=CF
AB=DE,
BCD collinear,
AECF collinear
Goal:
∠BAC=∠DEC

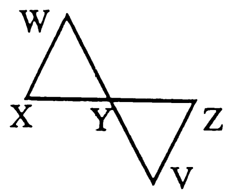
Domain Rules

Congruence of triangle
IF XY=PQ, YZ=QR, ZX=RP
THEN ΔXYZ ≡ ΔPQR

Isosceles
IF XY=YZ
THEN ∠YXZ=∠YZX

Figure 16.1. Geometry domain: examples of a problem and domain rules

If there is a control choice node, XY=YZ



IF XY=YZ, WY=YV, collinearXYZ, collinearVYW
THEN
∠XYW=∠ZYV, ΔXYW ≡ ΔZYV,
∠XWY=∠ZVY, ∠WXY=∠VZY,
XW // VZ, XW=ZV

Figure 16.2. An example of perceptual-chunk (which can be useful in solving Problem 1 of Fig.16.1)

ments and the existence of a segment respectively. Domain rules are general knowledge describing the natures of geometrical objects. Figure 16.1 shows an example of a geometry problem as well as two examples of domain rules. Figure 16.2 is an example of a perceptual chunk, which says, 'When there is a fact (control decision node) such as *segment* XY = *segment* YZ, try to apply the domain rule of Triangle-Congruence to the fact preferably, and subsequently apply the designated macro-operator if possible.'

2.2 EBL as a learner in geometry

Various versions of EBL systems have been proposed as a technique for learning search control knowledge. Experiments with a STRIPS-like pure EBL technique (Minton, 1985) confirmed that problem solving efficiency degrades remarkably as macro-operators are learned, which is caused by the two limitations of these earlier EBL systems. One is the limitation of the ways of selecting what to learn, i.e. their target concepts were essentially the same as the goals of the problem solving traces (as in

(Fikes *et al.* 1972; Mitchell *et al.* 1983)). The second is the lack or deficiency of utility measures for storing only useful macro-operators. Some methods had no measure (Fikes *et al.* 1972; Minton 1984) and others merely had a simple measure (Minton 1985). The PRODIGY system (Minton *et al.* 1989) addresses these two problems by providing four kinds of meta-level target concepts (i.e. 'succeeds', 'fails', 'sole-alternative' and 'goal-interference') and by evaluating the cost-effective utility of the learned control knowledge (Minton 1990) over a series of experiences of solving other problems.

However, in the domain of geometry problem-solving, learning from 'fails', 'sole-alternative', and 'goal-interference' will not lead to useful knowledge, because there 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 in this domain where facts increase monotonically in the problem space as reasoning proceeds. This is unlike task planning where applications of operators successively change the state of the reasoning target. Consequently, it is only 'succeeds' that may work well in the domain of geometry problem-solving. This means again that the target concept is essentially the same as the goal node of the problem. So, the EBL technique cannot go beyond the first limitation mentioned above in this domain.

In the fourth section, we will examine the operationality of the knowledge learned by EBL, based on the experimental data in solving geometry problems.

3 THE PCLEARN SYSTEM

3.1 The learning concept

In order to address the problems mentioned in the previous section for the purpose of learning a useful set of perceptual-chunks from problem-solving traces, we proposed a new learning concept quite different from the 'goal-orientedness' of EBL; PCLEARN acquires, for each control decision node in the problem-solving traces, *an assembly of diagram elements that are visually recognizable and grouped together with the control decision node* as a chunk. It then learns the macro-operator information in-

Input

Training Example : each control decision node

Domain Rules : knowledge represented as production rules

Chunking Criterion : "recognition rules"

Output

The pair of a control decision node and the relevant perceptual-chunk with macro-operator information.

Figure 16.3. The specification of PCLEARN chunking technique

cluded in the chunk as search control knowledge.

For that purpose, PCLEARN has a criterion for determining which portion of the problem diagram is recognizable and grouped together with each control decision node. The criterion is a set of '*recognition rules*', domain-specific knowledge which describes the necessary conditions for a domain object to be recognizable. These take the following form;

$$\begin{aligned}\text{recognizable}(Obj)\text{:-}\ & \text{recognizable}(Obj_1), \ldots, \text{recognizable}(Obj_N), \\ & \alpha(Obj_1,\ldots,Obj_N).\end{aligned}$$

In order for Obj to be recognizable, all the objects Obj_1 through Obj_N must be recognizable and also an additional condition $\alpha(Obj_1,...,Obj_N)$ has to hold. $\alpha(Obj_1,...,Obj_N)$ is a relation between the argument objects and/or the objects composing the argument objects. The precise procedure for acquiring perceptual-chunks by use of these recognition rules is shown in the Section 3.3.

The specification of PCLEARN perceptual chunking is summarized in Figure 16.3. PCLEARN selects training examples from the problem-solving traces according to the definition of control decision node. Control decision nodes are nodes that are members of the successful proof tree for which there is at least one tested domain rule which has been found to be applicable when the other tested ones were not. The output of PCLEARN is the pair of a control decision node and the perceptual chunk relevant to that node with macro-operator information telling what domain rules should be subsequently applied to that node. The learned knowledge can be regarded as a counterpart of 'preference rules' (Minton *et al.* 1989) of the PRODIGY system.

3.2 The overview of PCLEARN

The PCLEARN system includes the following modules;

- **A domain-independent problem solver.** This deals with the task of proving a fact that holds in a given assembly of domain objects when a set of other facts is known to hold, e.g. theorem-proving or diagnosis, using domain rules as well as search control rules. Domain rules are general domain knowledge. They are represented as production rules which have preconditions (sets of facts) in the IF part and a conclusion (a fact) in the THEN part.
- **Chunking facility.** PCLEARN's chunking method is explained in the previous section.

The problem solver's search is conducted by repeating the following decision cycle until the goal node is derived;

1. A node in the search tree is chosen. A node represents a fact which has been given or proved to hold in the problem space.
2. Domain rules (or search control rules) which can be applied to that node in a forward direction are searched for. The domain rule applicable in a forward direction to a node is the one which has an element of the IF part unifiable to that node and whose other elements in the IF part can also be unifiable to the already existing facts.
3. If there is no applicable domain rule, go back to 1 and select another node. If there is one, add a new parent node(s), representing the instantiated fact of the THEN part of the domain rule, whose children are the nodes representing the set of facts in the IF part. Unless the new node is unifiable to the goal node, go back to 1.

3.3 Algorithm for PCLEARN chunking

In creating a perceptual-chunk for a domain rule which is successfully applied to a control decision node, PCLEARN first identifies all the *recognizable* domain objects included in the rule.[2] It then enumerates all the *recognizable* features of these

[2]This rule is denoted as SAR (Successfully Applied Rule) in this paper.

```
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.
```

Figure 16.4. The set of recognition rules in geometry

objects. This produces a perceptual chunk which is the assembly of the objects with their features. Recognition rules are used in the first process.

Figure 16.4 is the set of recognition rules in the geometry domain. 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 to be recognizable because X is a constituent member of XY. In general, when an object is already found to be recognizable and we want to prove the recognizability of another object which is a structural constituent member of the former object, we do not need any additional conditions. The first three rules in Figure 16.4 belong to this category. On the other hand, when we prove the recognizability of an object from the other objects which compose that object, we need some (sometimes no) additional conditions. For example, when we prove the recognizability of segment XY from the recognizabilities 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 Figure 16.4), 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 Figure 16.4). The last two recognition rules are examples where no additional condition is needed by chance, although they belong to this category.

3.3.1 *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

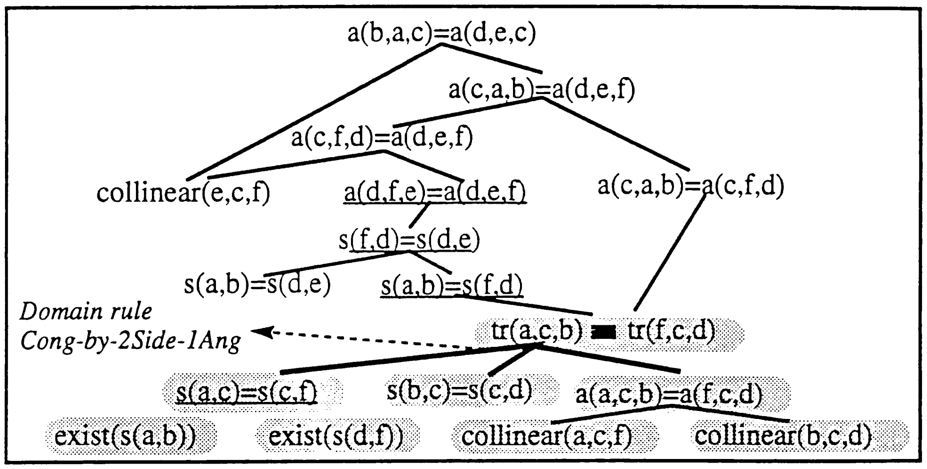


Figure 16.5. The successful proof tree of Problem 1 of Fig.16.1

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.**

Figure 16.5 is a successful proof tree of the Problem 1 in Figure 16.1. 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 $tr(f,c,d)$, are asserted to be recognizable. 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 recognizable.

3.3.2 *Step 2: Enumerating recognizable features.*

The second step is to derive from the problem-solving traces all the recognizable features of the above recognizable objects. The procedures are

1. the literals appearing in the SAR are 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 recognizable, and

3. all the features that can be derived from the above recognizable literals using domain rules are recognizable.

What we have obtained so far is the derivation tree (the third procedure of Step 2). Note that 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.

Let us look at the example case of learning from $AC = CF$ in Figure 16.5. The recognizable literals to be picked up before the derivation process are shown in Fig. 16.5 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 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 will be removed from the macro-operator.

Note that owing to the existence of some additional conditions in the set of recognition rules, the learned macro-operator becomes more specific[3] than the SAR itself. In case of the above example, incorporating the two collinearities has been significant in obtaining a perceptual-chunk of the two congruent triangles located in a completely point-symmetry (the one in Figure 16.2), which is more specific than the two merely congruent triangles.

3.3.3 *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

[3]This specificity directly influences the operationality of the learned perceptual-chunks. In this sense, recognition rules play a crucial role in determining the chunked area.

Table 16.1. Assignment of problems to training and test sessions

Sessions	Category 1	Category 2	Category 3
Learning	7	7	6
Test	3	4	3

domain rules. The generalized tree itself represents a macro-operator that has been learned for the control decision node. In the case of the above example, the one in Figure 16.2 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.

4 EXPERIMENTAL RESULTS

4.1 Method of experimentations

For simplicity, we divided the experiments into two sessions; a training session where problems are solved without using learned search control knowledge and learning is conducted for each problem, and a test session where problems are solved by use of search control knowledge obtained from the training session and no new learning is performed. We selected 30 geometry problems from some reference books on geometry, 20 of which are assigned to the training session and 10 are assigned to the test session. The problems we selected are limited to three problem categories; congruence (and/or similarity) of triangles (Category 1), natures of isosceles and right-angled triangles (Category 2) and natures of quadrilaterals (Category 3). The numbers of the problems selected for each category and assigned to the two sessions are shown in Table 16.1.

In selecting problems, we paid attention mainly to two issues. The first is that the numbers of training problems selected for the three categories should be approximately equal. This is in order to avoid problem selection in terms of categories for the training session which may cause the bias that certain perceptual chunks are learned more frequently, obscuring the issues of consistency in perceptual chunks across problems. The second is that the numbers of the test problems should also be approxi-

Table 16.2. Frequencies of the same perceptual-chunks being learned from many problems

Frequencies of being learned	The number of perceptual-chunks	
	PCLEARN	EBL
1	43	86
2	9	7
3	5	0
4	3	1
more than 4	4	0
total	64	94

mately equal in the three categories because the problems in the three categories should be equally tested using macro-operators.

4.2 Operationality of macro-operators

In this chapter, operationality of macro-operators is measured by the frequency at which each of the macro-operators is acquired from the problems in the learning session and applied to the problems in the test sessions. We investigated it in both cases of the EBL learner which learns from 'succeeds' and the PCLEARN system.

Table 16.2 shows the frequency at which macro-operators with the same diagram configuration are acquired during solving 20 problems in the training session. It seems to be quite a rare case that the EBL learner acquires the same set of perceptual chunks from different problems. On the other hand, PCLEARN learns several kinds of perceptual chunks more frequently in different problems.

Table 16.3 shows the results of the frequency at which those learned macro-operators are successfully applied to problems in the test session. The macro-operators of the EBL learner were applied 15 times, out of which 13 were successful, while the macro-operators by PCLEARN were applied 44 times, out of which 39 applications were successful. Success rate is about the same with both learners but the frequency is much larger in PCLEARN.

Table 16.4 shows the percentage of the nodes which were related to applications of macro-operators against all the nodes

Table 16.3. Frequencies of applications of the learned knowledge

Frequency	PCLEARN	EBL
Total applications	44	15
Successful applications	39	13
(ratio)	(89%)	(87%)

Table 16.4. The ratio of the nodes related to applications of macro-operators against all the nodes in a proof-tree (average over all the test problems)

PCLEARN			EBL		
Mean (%)	Max. (%)	Min. (%)	Mean (%)	Max. (%)	Min. (%)
67.5	86.0	38.0	30.0	80.0	0

in the successful proof tree, i.e. a measure of how much macro-operators contribute to constructing a proof-tree in the test session. The shown data (mean, maximum and minimum) are statistics over all the test session problems. The degree of the macro-operators' contributions to constructing proof-trees are larger in using PCLEARN macro-operators.

According to the data on cross-problem learnability (Table 16.2), successful applicability (Table 16.3) and the degree of contribution to proof-trees (Table 16.4), an answer to the empirical question mentioned in the first section is that there is *little* consistency *in goal-structure* across geometry problems while there is indeed cross-problem consistency *in perceptual chunks*, i.e. in the domain of geometry, 'perceptually-chunked' macro-operators have higher operationality than 'goal-oriented' EBL macro operators and hence PCLEARN is more appropriate to this domain than EBL.

Table 16.5 shows an explanation why goal-oriented macro-operators have low operationality. The average size of the applied macro-operators weighted with the frequency of applications is compared with the average size of all the macro-operators learned in the training session[4]. Expert-selected macro-

[4]We define that the size of a macro-operator is the number of its IF part elements, a measure reflecting the ease of finding appropriate instantiations of its preconditions.

Table 16.5. The average sizes of all the learned and applied macro-operators

Range of average	EBL	PCLEARN	Expert-selected
All the learned macros	5.1	3.9	3.2
The applied macros	2.9	2.5	3.1

operators in the third column are the ones which are carefully selected by a geometry expert from among the set of perceptual chunks PCLEARN has acquired. In general, a large difference in both quantities means that a group of macro-operators with a certain size is not applicable. However this may cause considerable costs in testing to apply them in vain. In the case of the EBL macro, the average size of all the macro-operators is much bigger than that of the others which were actually applicable. This is mainly because the EBL learner acquires a chunk from all the paths from each control decision node to the goal of the problem and hence the learned macro-operators tend to be too big in size to be applied to the control decision nodes of future problems. The experimental data suggest that more localized small macro-operators, around control decision nodes which are not always goal-oriented, would have higher operationality in geometry domain.

From all this discussion, we conclude that the PCLEARN chunking module is superior to the typical EBL technique as a method for learning perceptual chunks in the geometry domain, and also that *recognition rules* are effective as an operationality criterion for determining the area of problem-solving traces to be chunked out.

4.3 Learning performance results

Previous experiments have revealed that macro-operator learning has some distinct (both positive and negative) effects on the search process. These reflect two sides of the same coin. The major good effect is referred to as the 're-ordering effect' (Minton 1990); the domain rules encoded as macro-operators are tried before other rules which might be tried first if there were no macro-operators, and consequently unsuccessful search

Table 16.6. Reduction of the explored nodes by use of macro-operators as well as frequencies of macro-operator applications, in four macro modes

Problem	Without macros			EBL			PCLEARN			Expert-selected		
No.	Expl.	Suc.	Tot.	Expl.	Suc.	Tot.	Expl.	Suc.	Tot.	Expl.	Suc.	Tot.
1	33	–	–	33	0	0	14	7	7	12	4	4
2	10	–	–	10	1	1	5	4	4	5	4	4
3	10	–	–	23	2	4	7	2	2	7	2	2
4	19	–	–	18	2	2	15	8	8	19	2	2
5	12	–	–	11	2	2	9	4	4	9	2	2
6	17	–	–	15	2	2	15	3	3	15	2	2
7	17	–	–	17	0	0	7	2	2	7	2	2
8	14	–	–	11	2	2	12	3	6	10	3	3
9	8	–	–	4	2	2	8	2	4	4	1	1
10	15	–	–	15	0	0	8	4	4	7	3	3

Expl. -- The number of the explored nodes in the proof tree
Suc. -- Frequencies of macro-operators being applied successfully
Tot. -- Total frequencies of macro-operators being applied

will be put off later or sometimes left out. This reduces the search space.

Another negative effect is 'increased matching cost' (Minton 1990). As the number of macro-operators increases, the potential frequencies of testing domain knowledge (domain rules and macro-operators) at each control decision node also increases; if no macro-operators are applicable at a control decision node, the problem solver will have to resort to its domain rules, which means that the matching cost in considering the macro-operators was unnecessarily consumed. The number of bindings for each precondition of a domain rule is especially large in domains like geometry. Thus increased matching cost produced by macro-operators severely affects the performance.

The third effect, which also tends to degrade performance, is unsuccessful macro-operator application. The way PCLEARN applies macro-operators is not in goal-oriented search control but in a more local, opportunistic search control. This may sometimes guide the solution search in the wrong direction, which will increase the search space as a whole.

We investigated the above three effects in solving 10 test problems using the following four macro modes; with no macro-operators and with each of the three sets of macro-operators mentioned in Table 16.5. A simple measure of the search space explored by the problem solver is the number of nodes to which the problem solver applied domain knowledge (domain rules and macro-operators). Table 16.6 shows the numbers of the explored nodes in solving 10 problems in each of the four macro modes, together with the statistics about all the applications of macro-operators and successful applications. It is observed that in all three macro-modes the search spaces are reduced (i.e. 're-ordering effect'), compared to no macro mode. There are some exceptional cases of unsuccessful macro-operator applications. Especially the PCLEARN macro-operators contributed much more to reducing the search space than the EBL macro-operators did. In order to reduce the search space considerably, problem solvers need a set of macro-operators with operationality higher than a certain threshold. The PCLEARN macro-operators exhibit a relatively high percentage of success, 89%. The ideal value 100% is seen in the case of Expert-selected-macro mode (see Table 16.6). This shows that the third effect mentioned above is just a minor one in the PCLEARN system.

Table 16.7 shows the experimental data of 'matching costs' (the total cost, the cost of domain rule matchings and the cost of macro-operator matchings) when solving the test problems in the four macro modes. In EBL-macro mode, the cost of macro-operator matchings is extremely large compared to the total cost in No-macro mode. This is mainly because a large number of inapplicable macro-operators with relatively large IF sizes (refer to Table 16.5) were unnecessarily tested.

In PCLEARN-macro mode, the cost of domain rule matchings is smaller than that in No-macro mode, due to the reduction of the search space by the re-ordering effect using PCLEARN macro-operators which have relatively high operationality. However, the cost of macro-operator matchings still exceeds the reduction amount of the cost of domain rule matchings, and hence the total cost does not pay in all the test problems compared to No-macro mode. Expert-selected macro-operators are ones which have been obtained by eliminating some of the PCLEARN macro-operators (as mentioned before) which do not satisfy sim-

Table 16.7. Matching costs in solving the test problems in each of the four macro-operator modes

Problem	Without macros			EBL			PCLEARN			Expert-selected		
No.	Tot.	Rule	Macro	Tot.	Rule	Macro	Tot.	Rule	Macro	Tot.	Rule	Macro
1	477	477	0	3889	841	3048	629	278	351	329	199	130
2	67	67	0	1052	57	995	110	14	96	44	11	31
3	45	45	0	1060	162	898	131	32	99	59	26	33
4	87	87	0	1205	159	1046	198	35	163	267	121	146
5	33	33	0	660	53	607	112	34	78	35	21	14
6	170	170	0	621	100	521	305	75	230	169	67	102
7	81	81	0	541	117	424	103	32	71	57	31	26
8	98	98	0	941	173	768	318	130	188	137	86	51
9	54	54	0	92	5	87	150	52	98	15	6	9
10	106	106	0	1506	175	1331	168	79	89	66	39	27

Tot. -- The total cpu-time cost taken in solving the problem
Rule -- The cpu-time cost taken for domain-rule matchings
Macro -- The cpu-time cost taken for macro-operator matchings (unit: sec)

ple requirements such as successful applicability and cross-problem learnability. This elimination contributed to reducing the cost of macro-operator matchings considerably (ranging from 20% reduction to even 90%) and thereby reducing the cost of domain rule matchings marginally. Consequently, in some of the test problems, the total cost is less than the cost in No-macro mode.

However, there are still some problems in which even the use of the Expert-selected macros does not pay in terms of the total matching cost. This is an issue to be addressed. From the data of Tables 16.6 and 16.7, in all the problems (No. 1, 2, 7, 9, 10) where the use of macro-operators pays, the reduction ratio of search space is more than 50%. This data shows that since the cost of macro-operator matchings is inevitable, the only way to minimize cost using macro-operators is to use those macro-operators which are empirically promising and give great re-ordering effects. For that purpose, we need to manage the utility of each macro-operator dynamically (as in Minton 1990). This must be based on empirical data of how frequently each macro-operator can be successfully applied to problems and in how many nodes of each of the applied macro-operator sequences re-ordering effects are expected.

5 FUTURE WORK

The experiments show that PCLEARN provides a method of learning perceptual chunks with high operationality. However, in order to obtain a small and highly organized set of cost-saving perceptual chunks further study must be made on 1) the mechanism of collecting empirical data of operationality and re-ordering effects for each macro-operator and 2) managing the utility of macro-operators dynamically. PRODIGY addresses this problem (Minton 1990). This is one of the areas we intend to study with a perceptual-chunking learner.

The PCLEARN chunking mechanism must be compared with other goal-structure-based learning methods like SOAR (Laird *et al.* 1987) and compilation of ACT theory (Anderson 1983) in terms of operationality and dynamic utility of learned search control knowledge. The characterization of learning mechanism is that problem solvers learn knowledge of how to satisfy the subgoals the solvers have established during problem solving, i.e. chunking all the lower subgoal structures of the target subgoal. If we applied this method to the domain of geometry, the learning procedure would be as follows. Problem solvers establish a subgoal for finding a domain rule which can be successfully forward applied to a control decision node and then chunk all the necessary conditions for deriving each precondition of the domain rule which was actually applied to the decision node. This mechanism may chunk quite a different range of the problem solving traces from a chunking mechanism which uses 'recognition rules'.

Finally, we have to examine the generality of the PCLEARN perceptual chunking mechanism. Currently, domain independentness is assured if PCLEARN deals with tasks of reasoning within structured objects that satisfy the requirement of monotonicity of the derived facts. A candidate task in which the extension of this method has to be examined might be in design and/or planning tasks where operator applications will change the state or forms of the domain objects, producing domain objects unseen at the initial state of reasonings. The two key requirements that have to be retained even in this extension are the following. Firstly all the kinds of domain objects which will potentially appear in the reasonings have to be listed in advance. Secondly recognition rules holding between those objects can be explicitly described as domain-specific knowledge.

6 CONCLUSION

We proposed a new technique of learning search control knowledge from problem solving episodes by the perceptual chunking mechanism. This approach is quite different from the 'goal-oriented' principle in EBL. Our method learns control knowledge that guides problem-solving search at a control decision so that the solver can recognize locally a perceptual chunk relevant to the node. The learned knowledge consists of chunks which are *assemblies of diagram elements that can be recognizable and grouped together with the control decision node.* In order to implement chunking, PCLEARN employs *recognition rules*, domain-specific knowledge describing the necessary conditions for a domain object to be recognizable.

In this chapter, experimental results of solving and learning from 30 geometry problems were presented for comparing both the goal-oriented EBL technique and the PCLEARN technique in terms of operationality of the learned knowledge and performance improvement by use of them. In the domain of geometry, there is little consistency across many problems in goal structure, but rather a lot of cross-problem consistency in perceptual chunks primarily. Reflecting the intrinsic nature of the geometry domain, perceptually-chunked macro-operators, have higher operationality than goal-oriented EBL ones, which tend to be too large to be applied to problems. In this respect, the EBL technique does not work well in this domain.

'Recognition rules' are useful because they produce search control knowledge with high cross-problem learnability, a high percentage of successful applications, and high contribution to proof-trees construction. The learned knowledge has a re-ordering effect on the search process which reduces the search space considerably. However, the cost of macro-operator matchings, a negative effect, cannot be neglected because some of the learned knowledge is inevitably not operational. So, a key issue in future research is how to obtain a small and highly organized set of perceptual chunks by eliminating ones with low utility and selecting ones with large re-ordering effects. This requires that we add to the current framework a mechanism for empirically measuring the utility of each perceptual chunk.

REFERENCES

Anderson, J. R. (1983). *The Architecture of Cognition*, Harvard University Press, Massachusetts, London, England.

Fikes, R., Hart, P. and Nilsson, N. (1972). Learning and executing generalized robot plans, *Artificial Intelligence, 3*, 251–288.

Koedinger, K. R. (1992). Emergent properties and structural constraints: advantages of diagrammatic representations for reasoning and learning, *Working Notes of the 1992 AAAI Spring Symposium on Reasoning with Diagrammatic Representations*, Stanford Univ., March 27–29.

Koedinger, K. R. and Anderson, J. R. (1990). Abstract planning and perceptual chunks: elements of expertise in geometry, *Cognitive Science, 14*, 511–550.

Laird, J. E., Newell, A. and Rosenbloom, P. S. (1987). SOAR: an architecture for general intelligence, *Artificial Intelligence, 33*, 1–64.

Minton, S. (1984). Constraint-based generalization, *Proceedings AAAI-84*, 251–254.

Minton, S. (1985). Selectively generalizing plans for problem solving, *Proceedings IJCAI-85*, 596–602.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning, *Artificial Intelligence, 42*, 363–391.

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.

Mitchell, T., Utgoff, P. and Banerji, R. (1983). Learning by experimentation: acquiring and refining problem-solving heuristics, *Michalski, R. S., Carbonell, J. G. and Mitchell, T. M., eds., Machine Learning: An Artificial Intelligence Approach.* Tioga, Palo Alto, CA, 163–190.

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.

Suwa, M. and Motoda, H. (1991). Learning abductive strategies from an example, *Working Notes of AAAI-91 workshop on Towards Domain-independent Strategies for Abduction; also Tech. Report 91-JJ-WORKSHOP, Department of Computer and Information Science, The Ohio State University*, 72–79.