THE NATURE OF GEOMETRY EXPERTISE: A CHALLENGE FOR SPEEDUP LEARNING

Kenneth R. Koedinger
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213 USA
koedinger@psy.cmu.edu

Abstract

While speedup learning mechanisms are typically evaluated in terms of their ability to improve the efficiency of a problem solving system, another source of evaluation is to check the results of learning against the knowledge content and organization of expert problem solvers. Investigating the knowledge organization of human or computer experts can not only provide an independent evaluation of speedup learning mechanisms, it may also suggest new approaches to speedup learning. Following this strategy, this paper reviews our cognitive model of expertise in geometry theorem proving (Koedinger & Anderson, 1990) and discusses its implications for speedup learning. In particular, we found the organization of expert knowledge in Geometry to be distinctly different from the knowledge organization that results from traditional speedup learning mechanisms. While the knowledge resulting from such mechanisms usually reflects the goal-structure organization of problem solutions, expert geometry knowledge is organized in terms of the perceptual structure of geometric objects. It is suggested that perceptual structure may provide a useful alternative to goal-structure as a source of guidance in the formation of efficient knowledge during speedup learning.

1 INTRODUCTION

Koedinger and Anderson (1990) presented a cognitive model of geometry expertise (called DC) that is not only a more accurate account of the abstract planning abilities of human experts, but is also a more efficient problem solver than previous models. The knowledge in DC is organized around "perceptual chunks" that provide a way to cue clusters of relevant geometric knowledge from prototypic images in geometry diagrams. This knowledge is represented in what we call "diagram configuration schemas". The model presents a challenge for theories of skill acquisition and speedup learning in that current theories may be inadequate to achieve the particular kind of highly organized knowledge we have found geometry experts to have.

Speedup learning mechanisms, like chunking in Soar (Newell, 1990), compilation in ACT* (Anderson, 1983), or explanation-based learning in general, can be viewed as methods of composing and/or specializing the basic operators in the domain theory to come up with macro-operators that lead to more efficient problem solving. While the diagram configuration schemas that make up the DC model can certainly be described as macro-operators of the basic domain operators, it is unclear how they can be derived or learned by composing basic operators (and whether human experts learn them in this way). Macro-operators resulting from traditional speedup learning mechanisms will typically reflect the goal-structure organization in problem solutions. In contrast, diagram configuration schemas reflect the perceptual organization of object-structures in the domain.

This paper presents the DC model and poses the challenge for speedup learning. I also describe an analysis of the geometry learning environment that puts constraints on potential solutions to the challenge. Finally, I propose a partial solution to the challenge and frame the characteristics of a more satisfying solution.

2 THE DIAGRAM CONFIGURATION MODEL (DC)

In Koedinger and Anderson (1990), we described a study of geometry expertise in which we found the approach of human problem solvers to be distinctly different from the typical approach taken in previous cognitive models and expert systems for geometry theorem proving (Gelernter, 1963; Goldstein, 1973; Anderson, Boyle & Yost, 1985). Most previous models found proofs by performing a heuristic search in a problem space made up of formal geometry rules (i.e., definitions, postulates, or theorems) – the same rules that appear in the "reasons" column of a textbook proof (see Figure 1). In contrast to this rule-by-rule approach, we found that human experts take an abstract planning approach (Newell & Simon, 1972; Sacerdoti, 1974) – they initially make leaps of inference

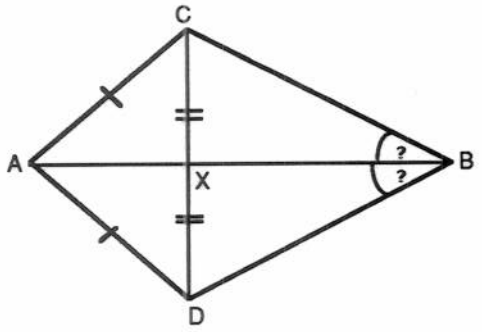
Appears in the proceedings of the Machine Learning workshop on Knowledge Compilation and Speedup Learning, Aberdeen, Scotland, July 4, 1992.

to identify the key steps in a solution. In the process they skip detailed formal steps that they later fill-in to produce a complete proof solution. In developing a plan for the problem in Figure 1, an expert would mention the key steps 5, 7, and 10 and tend to skip the others.

There was a real regularity in the kinds of steps experts were skipping and we argued that simple macro-operator or abstraction methods could not provide a good explanation for this step-skipping regularity. Instead we found that experts' knowledge appeared to be organized around certain prototypical geometric configurations. These configurations group together clusters of geometric knowledge that can be cued by the diagram and can lead to numerous inferences in a single step. These configurations were the basis of the DC model and we showed how they provided an accurate account of our step-skipping data and how they lead to a drastic reduction in the size of the geometry search space.

2.1 DIAGRAM CONFIGURATION SCHEMAS

Figure 2 shows two example configuration schemas. The configuration slot contains recognition knowledge that can pick out potential instances of the configuration in geometry diagrams. These configurations are perceptual chunks (Chase & Simon, 1973) that experts have acquired from considerable experience in solving geometry problems. Examples of instances of the CONGRUENT-TRIANGLES-SHARED-SIDE configuration (3) and the PERPENDICULAR-CROSS configuration (1) can be seen in the top row of Figure 3. The whole-statement slot indicates the geometry statement usually associated with this configuration (if there is one – not all configurations have labels in the conventional curriculum). The names of the next two slots have been generalized from Koedinger and Anderson (1990) to facilitate application of configuration schemas to other domains (see Koedinger, 1992 for an example). The part-properties slot indicate properties of parts of this configuration. They are true (i.e., can be proven) whenever the whole-statement is true or whenever one of the sufficient-conditions can be satisfied. These part-properties are shown in the bottom row of the network in Figure 3 and may occur in other configuration-schemas. The sufficient-conditions slot indicates subsets of the part-properties that are sufficient to prove the whole-statement and the remaining part-statements. This pattern completion nature of these configuration schemas explains one way in which experts can skip steps.



GIVEN: $\overline{AC} \cong \overline{AD}$
$\overline{AXB}$ bisects $\overline{CXD}$

PROVE: $\angle CBX \cong \angle DBX$

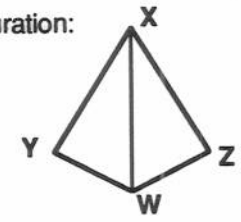
PROOF SOLUTION:

Statements:	Reasons:
1. $\overline{AC} \cong \overline{AD}$	1. Given
2. $\overline{AXB}$ bisects $\overline{CXD}$	2. Given
3. $\overline{CX} \cong \overline{DX}$	3. Def-bisector (2)
4. $\overline{AX} \cong \overline{AX}$	4. Reflexive
5. $\triangle ACX \cong \triangle ADX$	5. Side-Side-Side (1 3 4)
6. $\angle AXC \cong \angle AXD$	6. Corresponding-Parts (5)
7. $\overline{AB} \perp \overline{CD}$	7. Congruent-Adjacent-Angs (6)
8. $\angle BXC \cong \angle BXD$	8. Congruent-Adjacent-Angs (7)
9. $\overline{BX} \cong \overline{BX}$	9. Reflexive
10. $\triangle BCX \cong \triangle BDX$	10. Side-Angle-Side (3 8 9)
11. $\angle CBX \cong \angle DBX$	11. Corresponding-Parts (10)

Figure 1. A geometry proof problem and example solution.

CONGRUENT-TRIANGLES-SHARED-SIDE :

Configuration:



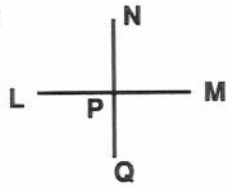
Whole-statement: $\triangle YXW \cong \triangle ZXW$

Part-properties:
1. $\overline{XY} = \overline{XZ}$
2. $\overline{YW} = \overline{ZW}$
3. $\angle Y = \angle Z$
4. $\angle YXW = \angle ZXW$
5. $\angle XWY = \angle XWZ$

Sufficient-conditions:
{1 2} {1 4} {2 5}
{4 5} {3 4} {3 5}

PERPENDICULAR-CROSS

Configuration:



Whole-statement: $\overline{LM} \perp \overline{NQ}$

Part-properties:
1. rt $\angle LPN$
2. rt $\angle MPN$
3. rt $\angle LPQ$
4. rt $\angle MPQ$
5. $\angle LPN = \angle MPN$
6. $\angle MPN = \angle MPQ$
7. $\angle MPQ = \angle LPQ$
8. $\angle LPQ = \angle LPN$

Sufficient conditions:
{1} {2} {3} {4}
{5} {6} {7} {8}

Figure 2. Two examples of diagram configuration schemas. The numbers in the sufficient-conditions indicate part-properties. Thus, in the congruent-triangles-shared-side schema {1 2} means that if the part-properties XY=XZ and YW=ZW are proven, all the properties of the schema can be proven.

2.2 RECOGNITION AND SEARCH IN DC

DC's processing combines a *recognition* process that identifies configuration instances in a problem diagram (see Koedinger, 1992 for a discussion of how the diagram is used to achieve efficient problem solving) and a *search* process that looks for a chain of schema instances that link a problem's givens to its goal in a logically sound way (as specified in the other slots of the schema). In theory the schema recognition and search processes are interleaved, but in practice the computer simulation of DC does all the recognition first – producing a network like the one shown in Figure 3 – and then searches this network for a problem solution.

In solving the problem in Figure 1, DC recognizes 14 configurations and, in the process of recognition, connects them with other configurations based on overlapping part-properties. The result is a network, most of which is shown in Figure 3 (the remainder is not relevant to the solution of this particular problem). The task of the search process is to find a path through this network that connects the problem givens to the problem goal subject to the sufficient-conditions of the schema traversed. For example, the given $\overline{AXB}$ **bisects** $\overline{CXD}$ leads to $\overline{CX} \cong \overline{DX}$ (not shown in Figure 3). $\overline{CX} \cong \overline{DX}$ and the other given $\overline{AC} \cong \overline{AD}$ are enough to satisfy a sufficient-condition of the CONGRUENT-TRIANGLES-SHARED-SIDE schema $\triangle ACX \cong \triangle ADX$. As a result, the remaining part-statements of $\triangle ACX \cong \triangle ADX$ are known. One of these, $\angle AXC \cong \angle AXD$, satisfies a sufficient-condition of the PERPENDICULAR-CROSS schema $\overline{AB} \perp \overline{CD}$ and so it's part-properties are known. (Alternatively, at this point $\angle CAX \cong \angle DAX$ and $\overline{AC} \cong \overline{AD}$ satisfy $\triangle ACB \cong \triangle ADB$ which has the goal $\angle CBX \cong \angle DBX$ as a part-property – leading to an alternative solution.) Lastly, $\angle BXC \cong \angle BXD$ and $\overline{CX} \cong \overline{DX}$ satisfy $\triangle BCX \cong \triangle BDX$ so its part-properties are known, one of which is the problem goal $\angle CBX \cong \angle DBX$.

2.3 EVALUATION OF DC

A summary of our evaluation of DC as both a computational system and a psychological model appears below – for more detail see Koedinger and Anderson (1990).

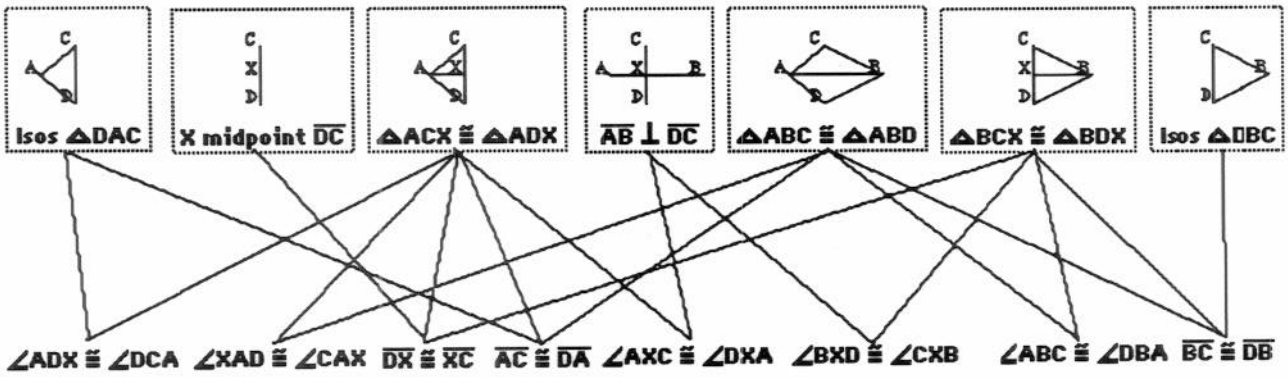


Figure 3. A portion of the network formed by DC's diagram parsing and configuration conjecturing components for the problem in Figure 1.

2.3.1 Relative Search Space Size

A task analysis was performed on a problem of about similar difficulty to the one in Figure 1 to compare the size of the domain theory search space with that of the space of DC schemas. The final domain theory solution to this problem involves 7 steps. There are 45 possible domain rules inferences that can be made from the givens of this problem, from these inferences another 563 inference can be made, from these greater than 100,000 can be made. The search space explodes yielding more than 10^6 rule applications in a breadth-first search to the solution. In contrast, the final solution in DC schemas involves only 3 steps and the breadth-first search explores 8 schema applications.

2.3.2 Modeling Human Experts

We established the psychological accuracy of DC by showing that it does a good job of accounting for the steps that subjects mention or skip in their verbal reports of proof planning. In the verbal reports of 5 experts' proof planning on a total of 12 problems, we found that less than half of the steps were mentioned (37/98) and more were skipped (61/98). In proof planning on these same problems, DC wound mention 29 of these steps and skip 69. Of the 29 steps that DC predicts will be mentioned, 23 were actually mentioned and only 6 were not. Of the 69 that DC predicts will be skipped, 55 were skipped and only 14 mentioned. Clearly there is a regularity in the steps being skipped and DC captures a lot of this regularity.

2.4 COMPARISON TO TRADITIONAL APPROACHES

It is worthwhile to consider whether the step-skipping behavior of geometry experts could be explained in terms of an alternative abstract problem space to the diagram configuration space. Two possible approaches to generating an abstract problem space are abstraction of domain rules and composition of domain rules into macro-operators. An abstract space can be created from the domain rules by an *abstraction* process where the conditions (if-part) of domain rules are generalized, for example, by dropping a clause which, ideally, refers to some detail that can be temporarily ignored (e.g., Sacerdoti, 1974; Unruh, Rosenbloom, & Laird, 1987). Such "minor" clauses in the domain rules of geometry are rare – dropping clauses most often results in operators that can propose future states which cannot be proven. Such incorrect plans can cause significant efficiency problems, however, this is not the only criticism of the adequacy of this abstraction method. In addition, this method is inconsistent with the observation that the abstract plans of DC and our expert subjects as well were always correct. That is, the abstract inferences they made don't produce unprovable statements. Thus, a "clause-dropping" type abstraction process is unlikely to yield an expert-quality knowledge representation.

A second approach to building an abstract problem space is by *composition* of consecutively applicable domain rules. This general approach has received numerous instantiations, for example, ACT*'s composition (Anderson, 1983), Soar's chunking (Newell, 1990), Korf's macro-operator learning (Korf, 1987). Although most of these approaches have some stipulations of the appropriate context in which composition can occur, there is little in them that indicates whether or when some pairs of consecutively applicable operators are more likely to be composed than other pairs. Thus, we would not expect any regularity in the kinds of steps that would be skipped in an abstract problem space of composed domain rules. However, such a regularity is exactly what we observed of subjects.

One way this regularity might be achieved, for example, in the the Soar architecture, is by having the initial problem solver/domain theory contain a hierarchy of problem spaces corresponding DC schemas. However, this approach simply puts off the question of how would this hierarchy be learned in the first place.

Below I elaborate on the argument that traditional speedup mechanisms cannot explain the acquisition of DC's schemas. The problem results from the fact that the basic domain rules are organized in a very specific way within the DC model. Traditional approaches provide no clear reason why this organization would evolve in contrast to the numerous other possibilities. This problem of lack of constraint in macro-operator formation is similar to the "utility problem" (e.g., Minton, 1990; Tambe & Newell, 1989). In that case the problem was identified by finding that existing mechanisms may not always lead to speedup as expected. In this case we have recognized this problem in considering whether speedup learning could achieve a specific expert representation. It seems likely that if standard speedup techniques were applied to geometry they would not only fail to reach a state of knowledge that corresponds with DC, they would also run into severe efficiency problems along the way.

3 THE CHALLENGE

The challenge posed by the DC model is to find a learning mechanism that can acquire DC schemas from the initial domain theory of geometry rules. Since most speedup mechanisms are rule-based, an initial problem is to find a way to express DC schemas as rules. Section 3.1 addresses this problem. A more serious problem is that the set of macro-operators that correspond with DC's schemas is a small and particular subset of the possible ways to combine the domain theory operators into macros. This problem is discussed in Section 3.2.

3.1 DC SCHEMAS AS RULES

This can be done by having a rule for each of a schema's sufficient-conditions whose left-hand side contains the configuration and one of the sufficient-conditions and the right-hand side contains the whole-statement and the part-properties that are not a part of this sufficient-condition. For example, the schema in Figure 1 would need 6 rules for each of the sufficient-conditions. The rule corresponding with the first sufficient-condition would be:

If triangles $\triangle YXW$ and $\triangle ZXW$ share a side and
look congruent in the diagram,
and $\overline{XY} \cong \overline{XZ}$ is known,
and $\overline{YW} \cong \overline{ZW}$ is known,
then conclude $\triangle YXW \cong \triangle ZXW$
and $\angle Y \cong \angle Z$
and $\angle YXW \cong \angle ZXW$
and $\angle XWY \cong \angle XWZ$.

This is a composition of five domain rules as shown in Figure 4.

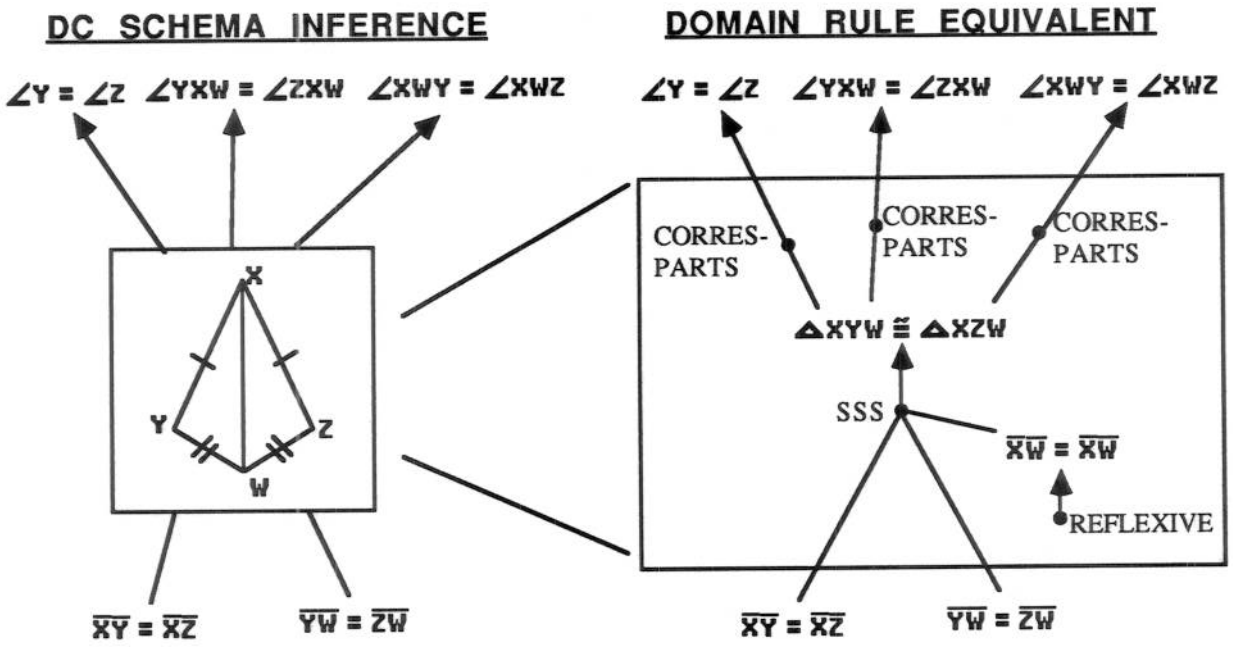


Figure 4. Illustration of a how a DC schema can be thought of as a macro of five domain rules.

3.2 DC SCHEMAS AND THE SPACE OF MACRO-OPERATORS

The collection of macro-operators that correspond with DC schemas, call it S, is a restricted subset of the space of possible macro-operators. S is restricted in two ways. First, S does not contain any of the possible macro-operators which could make inferences between statements which are whole-statements of schemas, for example, it doesn't contain an operator that could infer perpendicularity directly from triangle congruence in a problem like the one in Figure 1. Second, S does not contain any of the macro-operators with fewer then-part actions than macro-operators like the ones corresponding with the TRIANGLE-CONGRUENCE-SHARED-SIDE schema that have 4 then-part actions.

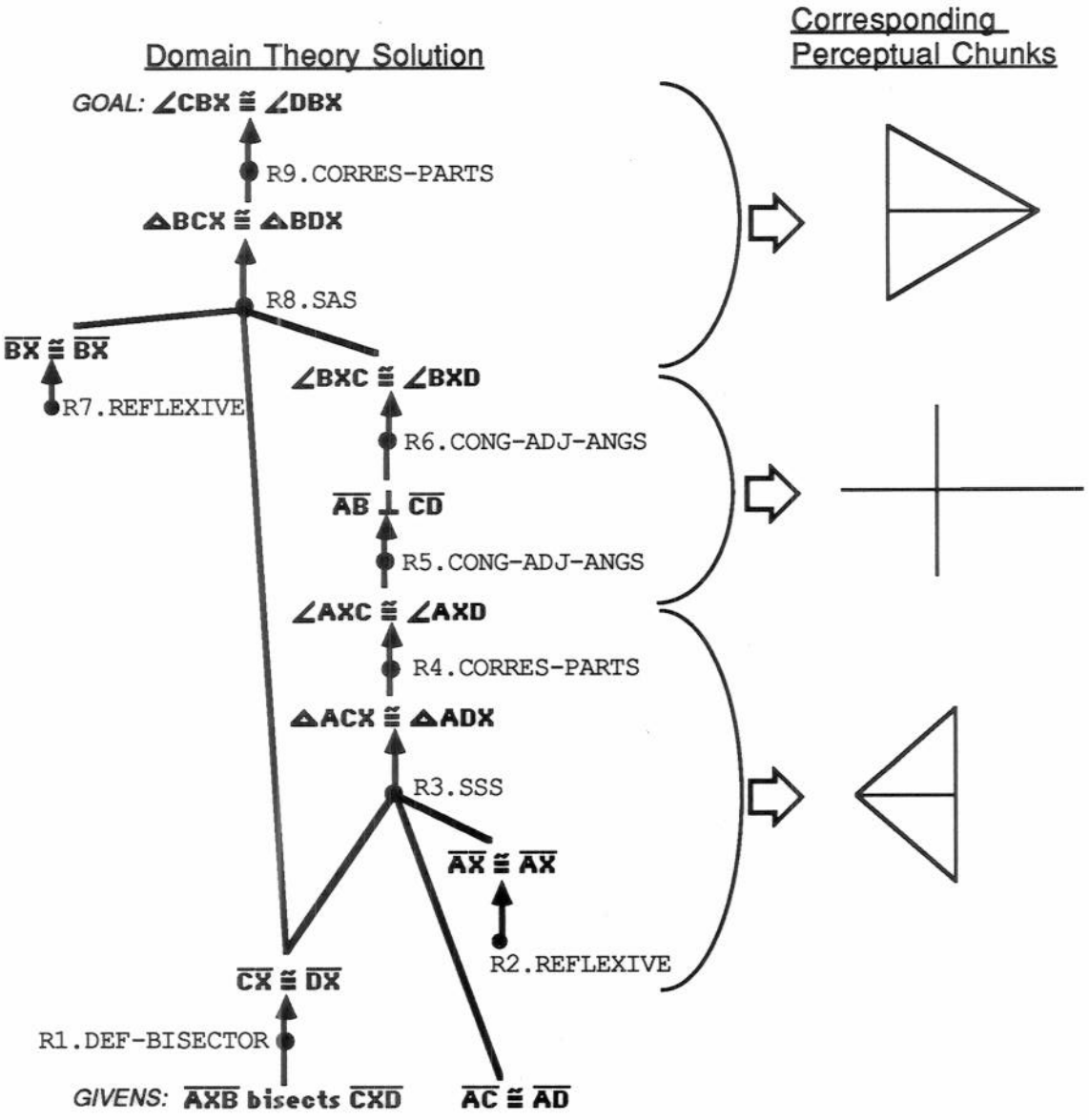


Figure 5. Perceptual chunks break up solutions in a particular way corresponding to only a small set of the potential macro-operators for the problem.

Figure 5 and Table 1 provide an illustration of this reduction. On the left in Figure 5 is a proof-graph solution to the problem in Figure 1. On the right are the perceptual chunks (configurations) that correspond with subsets of domain rule applications in the solution. For example, the first TRIANGLE-CONGRUENCE-SHARED-SIDE configuration **△ACX ≅ △ADX** corresponds with the rules R2, R3, and R4. Forming a macro-operator of these rules, call it R234, would have the same effect in this problem as this schema. However, forming a macro-operator R45 of R4 and R5, for example, would not correspond with a schema.

Table 1 shows the possible macro-operators that could be formed from the proof-graph in Figure 5 and whether they correspond or not with DC schemas. Of the 51 possible macro-operators, 3 correspond with DC schemas while 48 do not.

The key point is that adding "non-schema" macro-operators to a problem solver that already has "schema" macros would have negative utility. It would not increase the system's ability to solve problems in fewer steps and at the same time it would increase the matching time necessary to decide on the next operator to apply.

Table 1. Expert-like macro-operators are a small subset of the possible macro-operators.

	Total	Macro-operator content
Macro-operators corresponding to DC schemas.	3	R234, R56, R789
Remaining macro-operators.	48	R13, R18, R23, R34, R45, R68, R78, R89, R123, R134, R189, R345, R456, R568, R678, R689, R789, R1234, R1345, R2345, R3456, R4568, R5678, R6789, ..., R123456789

While most variations of traditional speedup mechanisms have some stipulations on the appropriate context in which macro-operators are formed, there is little in them that indicates precisely which sequences of consecutively applicable operators are more likely to be composed than others. Thus, it is unclear how such mechanisms would yield the highly organized and small subset of possible macro-operators that constitutes DC.

To be more precise, many speedup mechanisms (e.g., Soar's chunking or ACT*'s composition) stipulate that macro-operator formation occurs within a goal structure, that is, macro-operators are formed of consecutive operators applied to achieve the same goal. Thus, the clustering of operators into macro-operators will reflect the organization of a problem solver's goals and subgoals and to the extent that this goal structure is consistent across many problems in the problem solver's environment, a small and highly organized set of macro-operators might emerge. Whether there is such a consistency in goal structures across geometry problems is an empirical question (and one that demands a characterization of the goal structure that doesn't presuppose DC-like operators).

I've done some preliminary analysis of a database of geometry problems to try to address this empirical question -- that is, to see if there is any strong regularity in operator composition opportunities across problems.

3.3 PRELIMINARY ANALYSIS OF LEARNING ENVIRONMENT

It is clear that unguided macro-operator formation will not yield expert-like knowledge nor efficient problem solving. One straightforward approach is to use the number of times operators are used in a sequence while problem solving as an empirical guide to whether they should be composed into a macro-operator. For example, the composition mechanism of the ACT* theory takes such an approach (Anderson, 1983). The analysis below is intended both to show to what extent such an approach can work and to identify some characteristics of the learning environment that might constrain other proposed speedup mechanisms.

3.3.1 Method

The analysis involved counting the number of times a pair of domain rules occur in a two-rule sequence (i.e., when the conclusion of the first rule matches a premise of the second) in the solutions to a set of geometry proof problems. For example in Figure 5 the rules R8.SAS and R9.CORRES-PARTS appear in a sequence as do R6.CONG-ADJ-ANGS and R8.SAS. In all there are 9 possible two operator sequences in that solution. The solutions to 427 problems were analyzed. The problems were drawn from the curriculum of an Intelligent Tutoring System (ITS) for geometry (Anderson, et. al., 1985) and are representative of the problems a high school student would be given in a traditional geometry course. The solutions were generated from the expert system component of the ITS. The analysis was focussed on the 33 rules that overlap with the scope of the DC model (this eliminates rules corresponding with quadrilaterals). The two-rule sequences or "compositions" were categorized as either being *within-schema* compositions (e.g., R89 is within schema **△BCX ≅ △BDX**) or *between-schema* compositions (e.g., R68 is between schemas $\overline{AB} \perp \overline{CD}$ and **△BCX ≅ △BDX**).

3.3.2 Results

The analysis provided evidence that there is a reliable statistical difference in the number of within-schema compositions (about 30 occurrences on average) versus between-schema compositions (about 20 occurrences on average). However, the distributions are largely overlapping and many within-schema compositions occur quite rarely (once or twice) while a number of between-schema compositions occur quite often (> 30 times). Many between-schema compositions occur more often than within-in schema compositions – enough so that a system relying only on this subtle difference to select macro-operators would continue to get bogged down in creating unnecessary macros. On the other hand, using this statistical regularity might provide a useful supplement to other heuristics for macro formation.

3.4 SUMMARY

Many speedup mechanisms form macro-operators by composing domain rules that are used in achieving a particular goal. Thus, these macro-operators will reflect the organization of goal structures in solutions to problems in that domain. However, in geometry at least, it appears more likely that macro-operator-like knowledge is not primarily organized around goals but is organized around objects and aggregations of objects in the domain. DC's schemas are not really macro-operators in the sense of being derived from the basic domain operators. Rather, they seem to derive from perceptual chunking of domain objects and they merely bear a macro-operator relation with the domain rules.

4 POTENTIAL SOLUTIONS

4.1 HAVE GOAL-STRUCTURE REFLECT OBJECT-STRUCTURE

If it is true that expert knowledge is organized in terms of perceptual chunks rather than goal-based chunks as appears to be the case in geometry, this would seem to be a serious blow to unified theories like Soar or ACT* that are based on a goal-based learning mechanism. However, one potential fix to this apparent inconsistency is to find a way to frame geometry problem solving in such a way that the goal-subgoal structure becomes analogous to the perceptual or object structure. For example, this might be possible if an attention mechanism drove goal setting and this attention mechanism was responding to the perceptual structure in geometry diagrams.

4.2 OBJECT-STRUCTURE AS MACRO-OPERATOR SELECTION HEURISTIC

Relaxing the constraint of finding a solution consistent with a unified theory, there are other possible approaches to developing a learning model capable of producing DC-like knowledge. One promising approach is to use the perceptual structure of geometry diagrams (or, for example, inclined planes in physics) as a guide to macro-operator formation. This idea is an extension of Suwa and Motoda (1991) and came about in discussions with Suwa. In addition to the domain rules, the learning system would require a has-part hierarchy of objects in the domain (called "recognition rules" in Suwa & Motoda's system). For example:

triangle ABC has-parts {segment AB, segment BC, segment AC}

angle ABC has-parts {segment AB, segment BC}

These has-part rules can be used to simulate the process of drawing a particular figure (e.g., a triangle QRS and segment ST) and determining what other objects are or are not visible in that figure (e.g., segment QR, angle RQS, angle RST ..., but not angle SRT). Using these rules, the system can take advantage of the structural constraints in geometry diagrams to guide how to chunk together domain rules into macro-operators. The system inputs a proof graph like the one in Figure 5 and outputs a particular set of macro-operators selected from that proof graph as guided by the has-part rules. The algorithm is essentially as follows:

For each domain rule in the proof graph:

1. "Draw" a *focus-figure* that includes just the objects in the premises and conclusion of that rule.
2. Let *connected-statements* be all statements that are directly connected to the premises and conclusion.
3. For each connected-statement determine, using the has-part rules, which ones are visible in the focus-figure. Add any rules that connect to the visible statements to a set of rules that will form the macro.
4. Let connected-statements be all statements directly connected to the visible statements identified above. Go to step 3.

The focus-figure need not be drawn as described in step 1, rather the has-part rules can be used as described above to perform the analogous process. Consider how these steps apply to R9 in Figure 5:

1: The focus-figure is drawn including $\triangle BCX$, $\triangle BDX$, $\angle CBX$, and $\angle DBX$. It looks like the perceptual chunk on the top right of Figure 5.

2: Connected-statements = {$\overline{BX} \cong \overline{BX}$, $\overline{CX} \cong \overline{DX}$, $\angle BXC \cong \angle BXD$}

3: All 3 connected-statements are visible and are all reached by R8, so R8 is added to {R9} as the set of rules to be composed.

4: Connected-statements = {NIL, $\overline{AXB}$ bisects $\overline{CXD}$, $\overline{AB} \perp \overline{CD}$}

3': The empty premise of R7, NIL, is visible by convention so R7 is added to {R8, R9}. The other 2 connected-statements are not visible – both because $\overline{AX}$ is not part of the focus-figure – so no other rules are added.

4': Connected-statements is empty, so the iteration stops and returns {R7, R8, R9} as the rules to be composed into a macro.

The resulting macro R789 corresponds with one of the rules in the rule-based version of the CONGRUENT-TRIANGLE-SHARED-SIDE schema (see Section 3.1). Applying the algorithm to R8 yields the same result: R789. Applying it to R6 yields R56 which corresponds with the PERPENDICULAR-CROSS schema and to R4 yields R234 which is another instance of the CONGRUENT-TRIANGLE-SHARED-SIDE schema (R1 is not included because the $\overline{AX}$ in **$\overline{AXB}$ bisects $\overline{CXD}$** is not a part of the focus-figure). Applying it to R1 yields nothing more than R1.

This algorithm limits a proliferation of macro-operators by only forming macros out of consecutive rules that refer to parts within the same overall whole and not across wholes. For the most part these macros are isomorphic to DC schemas, however, they do deviate from them in a couple of ways. First, there needs to be a way to merge macros which refer to the same abstract configuration, like R789 and R234 in the example, but which are acquired at different times and may involve different part-properties and/or sufficient-conditions. Without such a mechanism, the pattern completion nature of DC schemas will not be achieved. Second, under certain circumstances this algorithm will produce macros that are larger than those in DC. For example, if the solution in Figure 5 had another step at the end concluding **$\overline{XB}$ bisects $\angle$CBD** from **$\angle$CBX $\cong$ $\angle$DBX** by R10.DEF-BISECTOR, the resulting macro would have included R10 since **$\overline{XB}$** and **$\angle$CBD** would both be a part of the focus-figure. Such a macro does not correspond with a single DC schema (in this case it's a combination of the CONGRUENT-TRIANGLE-SHARED-SIDE schema and the BISECTED-ANGLE schema), however, it is not clear at this point whether such macros would have a positive or negative effect on the efficiency of the resulting problem solver.

One interesting question that this approach raises is how the whole-part relations expressed in has-part rules might be learned. A mechanism like Servan-Schreiber and Anderson's (in press) CC model provides at least one possible answer. There is evidence to suggest that such learning is "implicit" or non-deliberate indicating a non-goal-based learning mechanism. Supporting this VanLehn (1989) found that certain learning events in human acquisition of strategies for the Tower of Hanoi puzzle are not marked by an impasse. The *perceptual chunking* of disks into "pyramids" account for a number of such events.

4.3 AN INDUCTIVE APPROACH TO DC SCHEMA LEARNING

The discussion above has taken as a premise that the initial domain theory will contain rules corresponding with the definitions, postulates, and theorems that appear in geometry textbooks. Learning such rules is knowledge level learning (Dietterich, 1986) and not typically the realm of speedup learning. However, given that DC schemas are much more like categories than rules, it seems plausible that inductive learning mechanisms might lead more directly to a DC-like representation than the indirect route of first learning the textbook domain rules and then deductively combining them into schemas as discussed in the previous solutions. Rather than simply interpreting textbook geometry rules and learning how to better syntactically apply them, geometry experts come to understand the semantic relationship between textbook formalisms and the classes of objects to which these formalisms refer. One possibility for capturing some of this is to take a more empirical approach like that of Yoo and Fisher (1991). Much like Yoo and Fisher's system categorizes algebra word problems, it appears expert knowledge may derive more from categorizing geometry diagrams and inducing general formal properties than it does from compiling and composing basic domain rules. One limitation of the Yoo & Fisher system is that it categorizes whole problems. DC's schemas are essentially subproblem categories so there is a need for a way to identify subproblems. Again something like knowledge of perceptual chunks or has-part structure might help guide this.

References

Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985). The geometry tutor. In *Proceedings of the International Joint Conference on Artificial Intelligence-85*. Los Angelos: International Joint Conference on Artificial Intelligence.

Dietterich, T. G. (1986). Learning at the knowledge level. *Machine Learning*, *1*, 287-316.

Gelernter, H. (1963). Realization of a geometry theorem proving machine. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and Thought*. New York: McGraw-Hill Book Company.

Goldstein, I. (1973). Elementary geometry theorem proving. MIT AI Memo 280.

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

Koedinger, K. R. (1992). Emergent properties and structural constraints: Advantages of diagrammatic representations for reasoning and learning. In the working notes of the AAAI Spring Symposium on Reasoning with Diagrammatic Representations, Stanford University, March 27-29.

Korf, R. E. (1987). Macro-operators: A weak method for learning. *Artificial Intelligence*, *27*, 35-77.

Minton, S. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *44*, 1469-1481.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, *5*, 115-136.

Servan-Schreiber, E. & Anderson, J. R. (in press). Learning artificial grammars with competitive chunking. *Journal of Experimental Psychology: Learning, Memory, and Cognition.*

Suwa, M., & Motoda, H. (1991). Learning abductive strategies from an example. Ohio State University technical report 91-JJ-WORKSHOP.

Tambe, M. & Newell, A. (1988). Some chunks are expensive. In *Proceedings of the Fifth International Conference on Machine Learning*, pp.451-458.

Unruh, A., Rosenbloom, P. S., & Laird, J. E. (1987). Dynamic abstraction problem solving in Soar. In *Proceedings of the AOG/AAAIC Joint Conference*, Dayton, OH.

VanLehn, K. (1989). Learning events in the acquisition of three skills. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: LEA.

Yoo, J. & Fisher, D. (1991). Concept formation over problem-solving experience. In *Concept Formation: Knowledge and Experience in Unsupervised Learning*. San Mateo, CA: Morgan Kaufman.