Using case-based reasoning and situated activity to write geometry proofs[1]

Thomas F. McDougal
Department of Computer Science
University of Chicago
1100 E. 58th Street
Chicago, IL 60637
mcdougal@cs.uchicago.edu

Abstract

As models of human cognition, previous geometry theorem-proving programs were inappropriately influenced by the ease with which computers manipulate syntactic formulae. The failure of those programs to pay attention to human perception doomed them as models of how humans solve geometry proof problems. Just as the study of theorem-proving once evolved into the study of planning, it is time now for theorem-proving to incorporate current ideas in the planning community. A close examination of what humans do when they try to solve geometry proof problems, and of how geometry is taught, reveals an emphasis on chunks of problem-solving knowledge derived from examples, retrieved on the basis of visual cues. These ideas are characteristic of the case-based reasoning and situated activity approaches in planning. This paper concludes with a brief description and trace of a computer program, POLYA, which does reactive, memory-based geometry theorem-proving.

Introduction

The computer makes a handy tool for exploring ideas about how people think. It forces us to define those ideas concretely, and provides us with feedback as to whether those ideas are adequate to describe human behavior. As we discover what a computer must know in order to perform a particular task, we gain insights into what people must know.

One danger of working with computers, however, is that it tempts us to do things in ways which are easy to program though not cognitively plausible. The computer is a very good symbol manipulator. It is easy, for example, to program a computer to test inference rules against a database of facts and goals.

The capabilities of the computer as a symbolic inference engine led to early research in automatic deduction and resolution theorem-proving [Newell & Simon, 1956; Robinson, 1965]. Though the early programs were not necessarily intended to model how people think, they evolved into more ambitious models of human problem-solving and planning [Newell & Simon, 1963, McCarthy, 1968, Fikes & Nilsson, 1971].

One characteristic common to the early planning systems is that they had to think hard to solve simple tasks. This was not a defect in their construction; [Chapman, 1985] showed that planning for conjunctive goals, as it was then conceived, is fundamentally intractable.

Yet people do amazing things without thinking very hard at all. They understand stories, get themselves fed at restaurants, and cross the street. If constructing plans is intractable, then people must be doing something else.

There are two apparently contradictory theories about what that something else might be. [Agre, 1987] showed that complex behavior could arise from simple situation-action rules driven by a changing world. People don't need to plan because they can rely on the world to tell them what to do [Chapman & Agre, 1986; Agre,

[1]This research is supported by the Office of Naval Research under contract N00014-91-J-1185, by the Defense Advanced Research Projects Agency monitored by the Air Force Office of Scientific Research under contract F30602-91-C-0028, and by the University of Chicago School Mathematics Project Fund for Support of Research in Math Education.

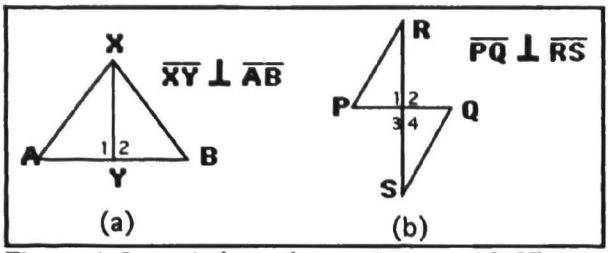

Figure 1: In an informal experiment with 27 subjects, six people marked both angles 1 and 2 in (a); eight people marked both angles 1 and 4 in (b). Only three people marked angle 2 in (b).

1988]. This theory is now called *situated activity;* it is a theory of reasoning by reacting.

Proponents of case-based reasoning (CBR), on the other hand argue that people don't need to plan for most tasks because they *already know* how to do it, having performed the task or similar tasks many times before [Schank, 1977; Schank, 1982; Kolodner, 1980; Hammond, 1989]. CBR is a theory of reasoning by remembering.

An ironic characteristic of theorem-provers and generative planners is that the more rules or operators they have, the longer it takes them to find a solution. The ideal situation for a CBR system is to have a lot of experiential knowledge in memory to increase the odds of having an appropriate plan for a particular task.

This creates a new problem, memory indexing and retrieval. The more plans in memory, the richer the feature vocabulary necessary to discriminate between them. Yet it should be easy to recognize a problem for which one knows the solution. The features to be used for indexing should be readily perceivable in the world.

Thus we see that a CBR reasoning system, like a situated action system, needs to be closely connected to a complex world. Whereas situated activity tries to make decisions only about the next immediate action, CBR tries to make decisions about the next plan.

As theorem-proving evolved into planning, we would like to see whether the CBR and situated activity ideas in planning can be used to build a better model of how people prove geometry theorems.

How people solve geometry proof problems

To see whether the CBR or situated activity views of planning might apply to geometry theorem-proving, we look first at the behavior of people.

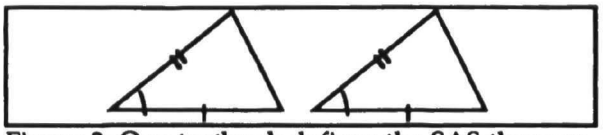
Figure 2: One textbook defines the SAS theorem of triangle congruence with this picture.

One interesting thing that people do when they solve geometry proof problems is *they mark the diagram.* Despite the pervasiveness of this behavior, no computer models of geometry theorem-proving have modelled it.

The usual explanation of why people mark the diagram is that the marks reduce the cognitive load of remembering which objects are congruent. This explanation is inadequate for two reasons. First, the same information is usually written immediately alongside the diagram. Presumably the same cognitive assistance could be had by checking the information there. Second, the "cognitive load" explanation fails to explain consistencies in the way people mark diagrams. Given the situation in figure 1(a) many novices and experts placed right-angle marks in both angles 1 and 2, even though only one mark is sufficient to record the information. Likewise, in figure 1(b), people are far more likely to place a right angle mark in angle 1 or in both angles 1 and 4 than to place a mark in angle 2, despite a tendency otherwise to place marks in angles to the right of the vertical perpendicular line.

The real reason for marking a diagram is found by looking at how a good teacher or textbook presents the formal concepts of geometry. In [Rhoad et al., 1988], several triangle congruence theorems (SAS and ASA) are defined *only* by the patterns of marks in diagrams like figure 2. Furthermore, the sample problems and the easiest problems in the problem set ask the student to identify the relevant theorem solely on the basis of tick marks in the diagrams.

Apparently, the authors believe that patterns of marks on the triangles is the easiest way for the student to recognize the applicability of these theorems. This is why it is so helpful for the student to mark congruences on the diagram. A mark on an angle or segment becomes also a mark on every object of which that angle or segment is a part.

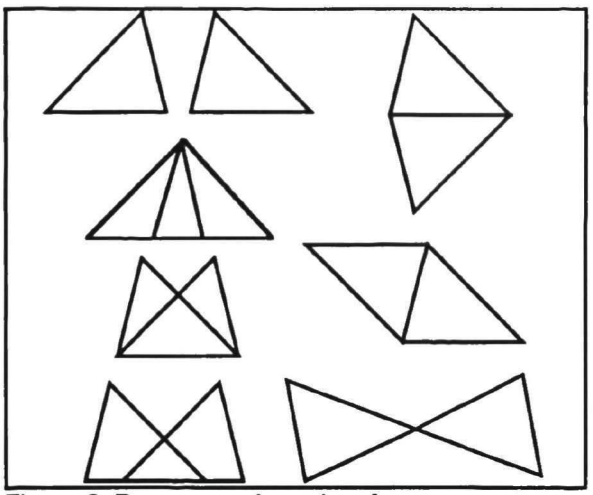

Figure 3: By rearranging triangles on an overhead projector, a teacher provides a quick preview of many of the congruent triangle patterns that his students will later encounter.

As with the triangle congruence theorems, the good textbook or teacher builds the students knowledge of significant visual patterns. As with triangle congruence, formal concepts are always accompanied by one or more illustrative diagrams, as are many informal ones. A teacher[2] shuffles congruent triangles around on an overhead projector, thereby introducing his students to patterns of congruent triangles they will encounter later (figure 3). This exposure enables his students to recognize such configurations as containing pairs of congruent triangles.

The behavior of people when they get stuck shows how much they rely on visual patterns. When people get stuck, they study the diagram intently, grouping and regrouping objects, their pencil frequently following their gaze as they look for a familiar and meaningful pattern.

This may sound equivalent to mental search for an appropriate rule, but it is not. Mental search would be characterized by staring off into space, tapping a pencil, asking "What rule could I apply here?" On the contrary, when people realize what they need to do, they exclaim, "Oh, yes, I see it," indicating that they have just seen something they did not see before, *not* that they have successfully matched a rule to something previously seen.

From the examples in texts, from the way good teachers teach, and from observations of humans solving problems, geometry theorem-proving turns out to be more like situated activity and case-based reasoning than like resolution theorem-proving or deductive search. People seem to solve problems by recognizing patterns they have seen before and making the inferences that were useful in the past.

We now introduce our computer program, POLYA, a recognition model of geometry theorem-proving.

An overview of POLYA

Our model of geometry theorem-proving calls for a memory of well-rehearsed proof-writing knowledge indexed by readily perceivable features. The reason this does not trivialize the task is because the diagram contains many more features than can be perceived at once; what you see depends on where you look. Furthermore, the diagram changes during the problem-solving task, as inferences are made and marks are placed on the diagram to record those inferences. Therefore, in addition to proof-writing knowledge, the problem-solver must also have knowledge about where to look to find the features which will reveal the solution.

As a computer program, POLYA is very simple. The input to POLYA is a list of givens, a goal, and a diagram. Its output is a proof. In between, POLYA operates as a plan interpreter and retrieval mechanism for accessing plans in memory.

POLYA currently has on the order of seventy plans for geometry problem-solving, of which there are two types: plans for directing visual search and plans for writing proofs. Both types of plans consist of sequences of actions and predictions of what should result from executing those actions. Most of the actions shift the focus of attention or compare two objects, but POLYA can also mark the diagram. An example of an action is LOOK-AT-BASE-ANGLE-1, which directs attention to one of the base angles of an isosceles triangle.

The diagram contains labelled points and lines described by cartesian coordinates. It also contains segment marks and angle marks. Segment marks are symbols such as SINGLE-TICK stored in a list associated with the coordinates of the segment endpoints. Angle marks are handled similarly.

When POLYA shifts its focus to an object, it uses a simulated visual system to compute a

[2]John Benson, at Evanston Township High School, Evanston, IL.

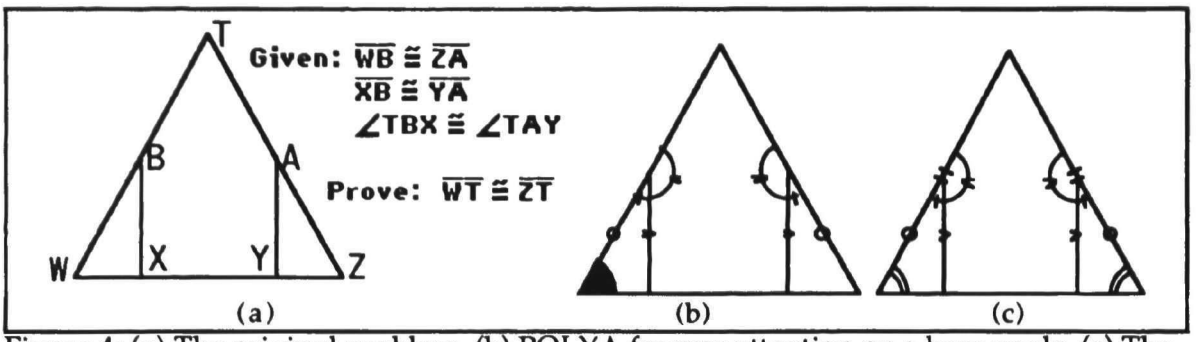

Figure 4: (a) The original problem. (b) POLYA focuses attention on a base angle. (c) The diagram, marked up, after POLYA has finished.

description of that object based on the diagram. This visual component is designed to return the features needed to support memory retrieval and inference without providing more information than a person could reasonably perceive at a glance. POLYA can focus attention on any of the basic geometry objects: points, angles, segments, triangles. The descriptions of those objects include such information as whether the object has a congruency mark on it, its approximate orientation, and its approximate size. In addition, POLYA can make spatial comparisons of two objects to find out, for example, whether two triangles share a side or whether two segments overlap.

The descriptions returned by the visual system are called *features*, and are used to retrieve plans from memory. POLYA's indexing scheme is based on the marker-passing scheme of the DMAP language understanding system [Martin 1990]. Usually it takes a sequence of several features to retrieve a plan.

Thus POLYA operates in a cycle. It executes the actions called for by a plan. The actions generate features, which are used to retrieve other plans from memory. The cycle stops when POLYA completes the proof or when no more plans have been triggered.

Example

One interesting problem that POLYA can solve is shown in figure 4 (a); the proof POLYA produces is in figure 5. Note that POLYA annotates the proof with the plans that wrote each section of the proof.

What is important, however, is not the final solution, but how POLYA found that solution. We summarize POLYA's reasoning in terms of the plans it executed. (By convention, we use the prefixes S- and P- to denote search and proof plans, respectively.)

S-READ-GIVENS
  Reads the given information.
P-CONGRUENT-SEGMENTS-GIVEN
  Marks the pairs of congruent segments in the diagram (runs once for each pair of segments).
P-CONGRUENT-ANGLES-GIVEN
  Marks the pairs of congruent angles.
S-CORNER-TRIANGLES
  Looks at the small corner triangles.
S-SIDE+SIDE
  Looks at the unmarked angles between the marked sides of the corner triangles; also looks at the third, unmarked sides; adds subgoals to prove those angles and those sides congruent.
S-PIER
  Looks at the angle adjacent to the unmarked angle in the corner triangle.
P-LINEAR-PAIR-PAIR
  Proves angles WBX and ZAY congruent. Runs a second time, starting on angle YAZ, but quickly halts because the angle has been marked.
S-SIDE+SIDE-1
  Angle congruence goal has been satisfied. Looks again at corner triangle.
P-SAS-SIMPLE
  Proves triangles WXB and ZYA congruent.
S-READ-GOAL
  Reads the goal.
S-CONGRUENT-SEGS-GOAL
  Looks at the segments (WT and ZT) which are supposed to be congruent.
S-ADJACENT-SEGMENT-ADDITION
  Looks at the subsegments of WT; if they are both marked (they aren't), it might be

(PLAN LINEAR-PAIR-PAIR)
angle XBT = angle TAY
Therefore, angle WBX = angle YAZ because of SUPPLEMENTS-SUBSTITUTION-RULE

(PLAN SAS-SIMPLE)
XB = YA
WB = ZA
angle WBX = angle YAZ
Therefore, ΔWXB = ΔYZA because of SAS-RULE

(PLAN CPCTC-ANGLES)
ΔWXB = ΔYZA
Therefore, angle XWT = angle TZW because of CPCTC-ANGLE-RULE

(PLAN ANGLES->ISOSC-LEGS)
angle XWT = angle TZW
Therefore, WT = ZT because of BASE-ANGLES->ISOSC-LEGS-RULE

Figure 5: The final product of POLYA's problem-solving activity: a proof, annotated with the proof plans that instantiated each step. (The givens are omitted for brevity.)

reasonable to use the segment addition property. The script halts as soon as it sees that BT is unmarked.

S-CPCTC-ANGLES-2
Runs twice: on angle WBX and angle TBX, and halts prematurely because they are already marked.

S-ISOSCELES
Looks at apex, legs, and base angles of the large triangle.

S-CPCTC-ANGLES-1
Compares base angle W with triangle WBX.

P-CPCTC-ANGLES
Proves corresponding angles W and Z congruent.

S-ISOSC-1
Looks again at large triangle.

P-ANGLES->ISOSC-LEGS
Proves the legs WT and ZT congruent, completing the proof.

Notice that POLYA took a tentative step down a dead-end path: since WT was subdivided, POLYA decided to check subsegments WB and BT. If those subsegments were already congruent to corresponding subsegments on ZT, it could prove WT and ZT congruent by adding the congruent subsegments. BT was not marked, however, so POLYA abandoned that idea.

This example shows a high degree of interaction with the diagram similar to the behavior of humans. POLYA uses features from the diagram bottom-up to index into a memory of plans. Those plans further direct visual search, making it possible eventually for POLYA to recognize how to write the proof.

Related work

Several researchers have worked on geometry theorem-proving from a cognitive modelling perspective. Several systems have sought to use the diagram as a source of heuristic information to control deductive search [Gelernter, 1959; Nevins, 1975; Greeno, 1983]. More closely related to POLYA is the Diagram Configuration model (DC) described in [Koedinger & Anderson, 1990].

In the DC model, diagram configuration schema associate patterns in the diagram with the rules most likely to apply to them. By parsing a diagram into schema, a restricted space of relevant rules is generated, and a solution is easily found by searching this restricted space.

DC was a significant advance in terms of using diagram configurations to organize formal geometry knowledge. This work extends the ideas in DC in two respects. First, POLYA models diagram parsing as an integral part of the problem-solving task. Second, POLYA uses more specific schema which allow it to make concrete decisions about which rules to instantiate, without knowing in advance how that rule will fit into the complete proof. In contrast, each of DC's schema may have several rules to choose from. Thus DC must fall back on traditional search within this restricted space to decide which rules are the appropriate ones.

By using a single mechanism and representation to model the entire problem-solving task, from diagram parsing to rule instantiation, POLYA is a complete model of what experts know and how they use what they know to solve geometry proof problems.

Conclusion

Our interest in modelling how humans solve geometry proof problems reflects a basic curiosity about how people think. We felt that existing models of geometry theorem-proving were too faithful to the representations and processes of

the original theorem-provers, whose design reflected what was easy to program rather than particular theories of human cognition. Those models of geometry theorem-proving failed to acknowledge the pattern-recognition abilities of people.

Ideas from the planning community, case-based reasoning and situated action, seemed to fit the behavior people exhibit when they try to solve geometry theorem. By combining these planning ideas, we have built a computer program which models the visual searching and pattern recognition of people doing geometry proof problem-solving.

References

Agre, P. & Chapman, D. 1987. Pengi: An Implementation of a Theory of Activity. In *The Proceedings of the Sixth National Conference on Artificial Intelligence,* 268-72. Menlo Park, Calif.: AAAI/M.I.T. Press.

Agre, P.E. 1988. The Dynamic Structure of Everyday Life. Ph.D. diss., Artificial Intelligence Laboratory, MIT.

Chapman, D. 1985. Nonlinear Planning: A Rigorous Reconstruction. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* International Joint Conference on Artificial Intelligence, 1022–24.

Chapman, D., and Agre, P.E. 1986. Abstract reasoning as emergent from concrete activity. In *Reasoning about Actions and Plans, Proceedings of the 1986 Workshop, Timberline, Oregon,* Georgeff, M.P., Lansky, A.L., eds. Los Altos, CA.: Morgan-Kaufmann.

Fikes, R. & Nilsson, N.J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189-208.

Gelernter, H. 1959. Realization of a geometry theorem proving machine. In *The Proceedings of the International Conference on Information Processing,* UNESCO, Reprinted in E.A. Feigenbaum & J. Feldman (Eds.) (1963), *Computers and thought.* McGraw-Hill.

Greeno, J.G. 1983. Forms of understanding in mathematical problem solving. *Learning and Motivation in the Classroom.* Paris, S.G., Olson, G.M., and Stevenson, H.W., eds. Erlbaum.

Hammond, K. 1989. *Case-Based Planning: Viewing Planning as a Memory Task.* Academic Press, San Diego, CA , Vol. 1, Perspectives in Artificial Intelligence.

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

Kolodner, J. 1980. Retrieval and organizational strategies in conceptual memory: A computer model. Ph.D. diss., Yale University.

Martin, C.E. 1990. Direct Memory Access Parsing. Ph.D. diss., Yale University.

McCarthy, J. 1963. Programs with common sense. In Minsky, M., *Semantic Information Processing.* MIT Press.

Nevins, A.J. 1975. Plane geometry theorem proving using forward chaining. *Artificial Intelligence* Vol. 6.

Newell, A. & Simon, H.A. 1956. The logic theory machine. *IRE Transactions on Information Theory* 2:61-79.

Newell, A. & Simon, H.A. 1963. GPS, a program that simulates human thought. In Feigenbaum, E.A. & Feldman, J., *Computers and Thought.* McGraw-Hill.

Rhoad, R., Whipple, R., & Milauskas, G. 1988. *Geometry for enjoyment and challenge,* McDougal, Littell.

Robinson, J.A. 1965. Automatic deduction with hyper-resolution. *International journal on comutational mathematics* 1:227-234.

Schank, R.C. & Abelson, R. 1977. *Scripts, Plans, Goals and Understanding,* Hillsdale, New Jersey: Erlbaum Associates.

Schank, R.C. 1982. *Dynamic Memory: A Theory of Reminding and Learning in Computers and People,* Cambridge University Press.