Representing and using procedural knowledge to build geometry proofs[1]

Thomas F. McDougal and Kristian J. Hammond

Department of Computer Science
University of Chicago
1100 E. 58th St.
Chicago, IL 60637
mcdougal@cs.uchicago.edu, hammond@cs.uchicago.edu

Abstract

What is the nature of expertise? This paper posits an answer to that question in the domain of geometry problem-solving. We present a computer program called POLYA which makes use of explicit planning knowledge to solve geometry proof problems, integrating the processes of parsing the diagram and writing the proof.

Introduction

This paper describes a computer program called POLYA that solves high school geometry proof problems. High school geometry first attracted our interest when the first author was student teaching geometry in a public high school as part of a teacher certification program. We were curious about what kinds of knowledge enabled him and other experienced mathematicians to solve geometry proof problems very quickly, in contrast to his students, who solved the same problems only with great effort. That knowledge had to involve more than the formal rules of geometry (the theorems, axioms, definitions), since, by virtue of their ability to solve the problems at all, the students clearly knew those rules. We conjectured that geometry expertise involves an ability to recognize when those rules *should* be used, in contrast to when the rules *can* be used. Such expertise generally arises from exposure to and experience with a large number of problems. The problem for us then was to define that knowledge concretely and to build a computer model of geometry problem-solving that made use of that knowledge. We call our computer program POLYA.

Although POLYA's task is to write geometry proofs, our desire to model human expertise led us away from some of the traditional concerns of automated theorem-proving. We are not concerned, for instance, with solving hard problems; rather, we are concerned with capturing the knowledge that allows experts to solve easy problems easily.

Our research has led us to address a broad range of important AI issues: visual reasoning and the use of diagrams in problem-solving; representation of planning and problem-solving knowledge; how to store, efficiently retrieve, and apply plans; how to integrate planning and action; how to use the world as a memory aid; how to direct a limited focus of attention to gather information; and what it means to know *how* to solve a problem as distinct from knowing the solution.

This paper describes our representation for geometry problem-solving knowledge and the computer program, POLYA, which uses that knowledge to write proofs.

Recognizing when rules should apply

Central to our model of human geometry theorem-proving expertise is a distinction between when a rule *may* be applied (as determined by its preconditions) and when a rule *should* be applied. A novice with complete understanding of the rules and their preconditions can still have trouble with a relatively easy problem. The novice may get lost in a large number of legitimate but useless inferences, or she may be reluctant to make a single inference without knowing how it will contribute to the final proof. The expert, on the other hand, shows a remarkable ability to make exactly those inferences relevant to the solution without knowing *a priori* what that solution is. [Koedinger & Anderson 1990] documents the tendency of geometry experts to make inferences from the given information without regard to the goal; [Larkin et al. 1980] documents analogous forward reasoning by physics experts.

We hold that most of the expert's decision-making is based on cues in the diagram. This thesis, so broadly stated, is not new; [Koedinger & Anderson 1990] presented a model of geometry problem-solving called DC in which the diagram is parsed into *configuration schema,* each of which defines a restricted subset of applicable rules. Their model contrasts with earlier systems [Gelernter 1959, Nevins 1975, Greeno 1983] which used the diagram primarily as a source of heuristic search control information.

[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.

We believe that our model addresses two shortcomings in the DC model. First, while the DC model makes a significant contribution in terms of knowledge representation for geometry problem-solving, it still says very little about the problem-solving process. The authors note that once the diagram has been parsed, finding the inference chain is trivial, which suggests that most, if not all, of the problem-solving task involves recognizing the relevant schema. Yet they consciously side-step the question of how people do this, with only a brief argument that perhaps the diagram parsing and schema search processes could be coordinated.

We also disagree with DC's model of schema application. Although DC's configuration schema significantly reduce the rule space, the model still falls back on inference chaining to decide, for each schema, which rule or rules should apply. The problem is that DC's schemas are overly general. Just as human experts are able to commit to specific inferences early in the problem-solving process, more specific schema would make it possible to decide exactly which rule should apply without a second phase of inference chaining.

In contrast, POLYA builds up the proof at the same time that it builds up its understanding of the diagram. It uses schema-like knowledge to parse the diagram on demand, and it recognizes highly specific configurations in the diagram which enable it to make concrete inferences likely to contribute to the final proof. The next sections provide an overview of POLYA's operation and a detailed description of its geometry problem-solving knowledge.

An overview of POLYA

POLYA comprises three basic modules: a memory retriever, a plan interpreter, and a module for simulated vision (figure 1). The memory retriever takes a steady stream of features and uses them to trigger plans in memory. The plan interpreter selects a plan and executes the steps in the plan. The vision module computes features in response to the actions called for by the plan. The next sections discuss these modules in detail.

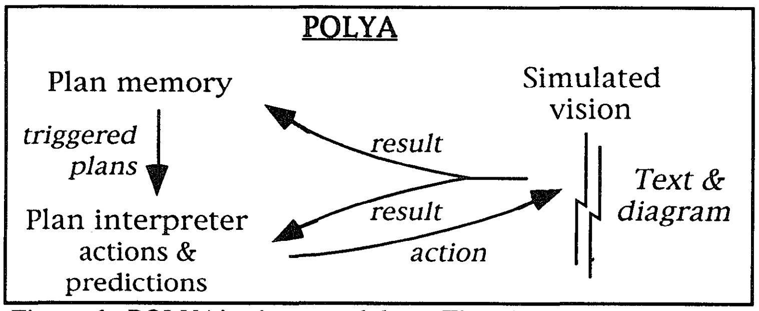

Figure 1: POLYA's three modules. The plan interpreter sends commands to the vision module, which computes a description of some part of the problem and sends that description both to the memory module and back to the plan interpreter.

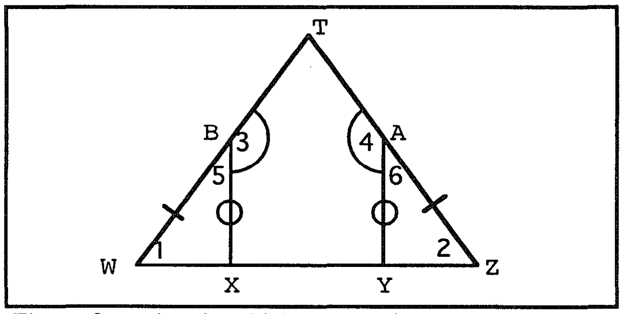

Figure 2: a triangle which appears isosceles. Angles 1 and 2 are the base angles. Angles 3 and 4 are marked congruent, as are two pairs of segments. This represents the initial conditions for one problem POLYA can solve.

The input

As in textbooks, a geometry proof problem for POLYA consists of givens, a goal, and a diagram. The givens and goal are predicates such as (congruent-segments (segment s x) (segment t y)). The diagram is a composite of lines and labelled points. Each line is defined by the coordinates of its endpoints, and labelled points are listed by their coordinate locations, as: (x y <label>), where <label> is a letter (A, B, etc.). The labels are irrelevant to the problem-solving process; they are used only to define the givens and the goal and to generate the proof in human-readable form.

Simulated vision

POLYA accesses the diagram by way of a simulated visual system. The visual system provides over 120 operators by which POLYA can specify which object or objects in the diagram it wishes to inspect; the visual system returns a description of that object including its exact location and a list of aspects.

The operator LOOK-AT-LEFT-BASE-ANGLE, for example, takes as its argument the vertices of a triangle which *appears* isosceles[2] and returns a description of one of its base angles (e.g. angle 1 in figure 2). That description includes the (*x*, *y*) location of the vertex, the compass directions of the rays, a symbolic description of the approximate size of the angle (e.g. ACUTE>45), a symbolic description of the pattern of rays at the vertex (SIMPLE-ANGLE), a count of the number of rays interior to the angle (zero), and, if there are no interior rays, a description of the space unto which the angle opens (TRIANGLE). The description also includes whether the angle is marked (e.g. angles 3 and 4 in figure 2 are each marked with SINGLE-ARC congruency marks).

POLYA can itself make annotations such as angle marks on the diagram. There are several benefits from such annotations. Looking at angles 3 and 4, it

[2]Based on euclidean distances between the vertices.

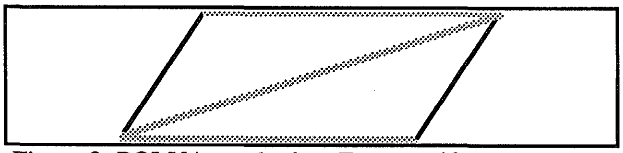
Figure 3: POLYA can look at Zs to see if the angles are congruent or the lines parallel.

is apparent from the marks that the angles are congruent to each other. Thus POLYA, like a person, need not remember which angles are congruent, since it can always get that information directly from the diagram. Furthermore, having a mark associated with an individual angle streamlines some inferencing. Looking at angles 3 and 5, POLYA can guess that angle 5 is probably congruent to some other angle (angle 6).

In addition to angles, POLYA can focus on and describe any of the visual objects involved in geometry proofs: points, segments, triangles, quadrilaterals. It can compare pairs of objects—two triangles, for example, or a segment and an angle—to see how they relate to each other spatially. When POLYA looks at a triangle, the visual system computes its shape (RIGHT, ISOSCELES, EQUILATERAL) and counts the number of sides and angles annotated with congruency marks. Thus marks on segments and angles are reflected in the descriptions of all larger objects which contain those segments or angles. So, for example, POLYA can recognize when all sides of a triangle are marked, suggesting the applicability of the side-side-side triangle congruency theorem. This is the other benefit of angle and segment annotations.

POLYA can also look at composite shapes sometimes mentioned in textbooks but never mentioned in proofs. One textbook, for example, explicitly teaches students Z and F patterns involving parallel lines cut by a transversal [Rhoad, Whipple & Milauskas 1988]. POLYA can look at Zs to see if the angles are marked congruent or if the lines are marked parallel (figure 3).

Operators can sometimes fail. FIND-ADJACENT-MARKED-ANGLE, for instance, can fail if no such angle exists. Such operators are nonetheless useful for efficiently locating objects of interest.

Geometry plans

To parse the diagram and write a proof concurrently, POLYA needs to gather information in an organized way while still responding flexibly to what it sees. POLYA has a large memory of geometry plans which structure visual search and instantiate rules; plans are triggered and executed on the basis of configurations in the diagram.

A geometry plan defines a sequence of actions which can be directly executed by the plan interpreter. A typical action looks like this:

```
((COMPARE-TRIANGLES ?tri1 ?tri2)
 :predict (tri-pair (extents shared-side))
 :unbind (?tri1 ?tri2)
 :bind ?tri-pair)
```

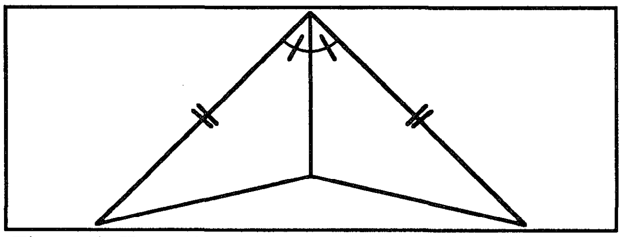
Figure 4: Two triangles share a side, and each triangle has one side marked and one angle marked. This pattern triggers the P-SAS-SHARED-SIDE proof plan, which checks that the marked angle is between the marked side and the shared side.
isosceles.

The first item is an action for the visual system whose arguments are defined by local plan variables. The prediction partially specifies what the result of that action should look like; if the result fails to match the prediction, the plan aborts. A typical use of predictions is to check preconditions of a rule. Each step may bind the result of the action to a local plan variable and may unbind variables no longer needed.

There are two types of plans: search plans and proof plans. *Search plans* direct the focus of attention to potentially relevant parts of the diagram, gathering information for the memory module. *Proof plans* instantiate formal rules of geometry and add them to the proof. Search and proof plans are structured and handled in the same way, except that POLYA runs proof plans preferentially. (See **plan selection,** below.)

S-ISOSCELES[3] is a typical search plan. It directs attention to the legs and base angles of a triangle which looks isosceles. This reflects knowledge that if the triangle is actually isosceles, the legs and base angles are more likely to provide useful information, and more likely to play a role in the proof, than the apex angle or the base side. P-SAS-SHARED-SIDE[3] is a typical proof plan: if POLYA detects two triangles each of which has one angle marked and one side marked such that the triangles also share a side (figure 4), the plan verifies that the angle between the marked side and the shared side is marked, then adds to the proof a statement that the triangles are congruent.

Search plans gather features which trigger proof plans and other search plans in an interacting cycle of plan selection and execution. To build a proof for an easy-to-moderate geometry problem, POLYA may make use of 12 to 28 plans, causing between 50 to 120 actions to be performed.

POLYA's geometry plans define what it means to know how to solve geometry problems. Once a plan has been triggered, execution is straightforward. This is consistent with our view of geometry problem-solving as being primarily a task of recognizing when a rule or other piece

[3]By convention, we use prefixes S- and P- to distinguish search and proof plans.

of knowledge should apply. Next we consider how POLYA recognizes which plans are relevant.

Plan memory and retrieval

We have said that plan execution generates visual results which in turn trigger other plans. In this section we describe specifically how plans are triggered.

POLYA's plan indexing scheme is based on the marker-passing scheme used in DMAP, a case-based natural language understanding system [Martin 1990]. Each plan has a plan index having two parts: an index pattern and index constraints. An *index pattern* consists of a sequence of partially-specified visual results. A typical index pattern looks like this one for the P-SAS-SHARED-SIDE proof plan:

```
( (triangle (angle-mark-count 1)
            (seg-mark-count   1))
  (tri-pair (extents shared-side)) )
```

This pattern detects a triangle with one marked side and one marked angle, and a triangle pair sharing a side. The index constraints specify that the triangle must be part of the triangle pair:

```
( ((?1) = (?2 triangle-1)) )
```

The memory module matches visual results against the index pattern using DMAP's marker-passing scheme. A marker is placed on a node representing the first element of the pattern. When a visual result matches that node, the marker is advanced to the next element in the pattern. When the last element has been matched, the plan is *triggered,* posted to a list as eligible for execution.

We wish to emphasize that a plan index should not be thought of as the antecedent of a rule. Relative to most rules, plan indices are both over- and under-specified: over-specified in the sense that they seek to recognize not only when a rule *may* be applied but when it *should* be applied; under-specified in that they do not always guarantee the applicability of the rule. In the case of the P-SAS-SHARED-SIDE proof plan, for instance, the marked angle must lie between the marked side and the shared side in each triangle, a constraint which is not easily captured by the raw descriptions of the triangle and the triangle pair. It would be possible to design the visual system to capture that information as one aspect of the triangle pair. That would subsume an important part of geometry knowledge in the vision module. We prefer, however, to represent explicitly as much geometry knowledge as possible in the plans.

As stated earlier, an important part of geometry expertise is the ability to make the inferences which are most likely to contribute to the final solution. POLYA makes an assumption which seems to work well for easy to medium problems: If some features in the diagram cause a search plan to focus attention on an object, then the object is likely to be relevant to the proof. For the P-SAS-SHARED-SIDE proof plan to be triggered, some other script must have directed POLYA's attention to the triangles. This does not guarantee that it will be useful to prove the triangles congruent, but it seems to be true most of the time.

Plan selection

When the plan interpreter module finishes a plan, POLYA chooses another from the list of triggered plans. Typically there are several search plans to choose from, and perhaps one proof plan. Proof plans are chosen ahead of search plans, since after all the task is to write a proof; and plans triggered more recently are chosen ahead of plans triggered less recently. This plan-selection algorithm is too simple to work in the long run, and we are experimenting with ways to incorporate knowledge about which plans should run first. In some cases one plan subsumes another; by annotating the larger plan we can ensure that POLYA runs it and not the other.

Issues

One of our objectives with POLYA was to integrate diagram parsing with the process of constructing a proof. In so doing, we have had to address issues associated more with planning, robotics, and vision than with traditional theorem-proving.

Planning and action

POLYA treats planning as memory retrieval, in the spirit of case-based planning [Hammond 1989]. Because feature extraction is a major part of POLYA's task, POLYA cannot solve problems by retrieving and adapting a single case, as a prototypical CBR system does. Instead, POLYA accesses memory constantly, retrieving and using tens of plans over the course of a single problem. POLYA plans and re-plans in response to what it sees.

This responsiveness to visual features is characteristic of situated activity [Chapman & Agre 1986, Agre 1988]. Yet POLYA is not purely reactive; its behavior is better characterized as *reactive planning* [Firby 1989]. It executes each plan to completion so long as its predictions are met. The plans provide necessary structure for apprehending complex relationships in the diagram and for writing a formal mathematical argument. POLYA strikes a balance between top-down memory-based planning and bottom-up reactivity.

Active sensing

In the computer vision community, researchers are acknowledging that it is both intractable and unnecessary to identify everything in an image [e.g. Ballard 1991, Clark & Ferrier 1988, Swain 1991]. Particular tasks require attention to only particular aspects of the image. One may be interested only in the object at the center of the image, or one may be concerned only with detecting rapid motion. Similarly, a robot may have a ring of sonars, but for moving forward across mostly-empty space it makes sense to ignore the sonars pointing aft.

Geometry diagrams are simpler and more constrained than an arbitrary image or a cluttered room, but the idea for POLYA is the same: different parts of the diagram are relevant for different tasks, and some parts of the diagram can be ignored altogether.

Furthermore, we suspect that POLYA's problem of coordinating multiple sensing operations is relevant to vision and robotics as well, and that the solution of using short plans for information-gathering, with top-down predictions, may apply.

Example

Figure 2 above is one example of a problem POLYA can solve. Here we show the diagram with all given information already marked on the diagram; one of the first things POLYA does is annotate the diagram to reflect the given information. The goal in this problem is to prove that the large triangle is isosceles, i.e. that its left and right sides are congruent (have equal lengths).

At startup, POLYA computes a general description of the diagram as a whole, capturing the left/right symmetry of the diagram and the basic shape: a triangle with corner triangles. Two search plans are immediately triggered based on this description: S-ISOSCELES, described above, and S-CORNER-TRIANGLES.

After marking the given information, POLYA executes the S-CORNER-TRIANGLES search plan, which focuses attention on the corner triangles. The descriptions of those triangles triggers S-SIDE+SIDE, which looks at angles 5 & 6 and at sides WX and YZ. The pattern of rays at the vertex of angle 5 triggers S-PIER, which compares angle 5 with its adjacent angle, angle 3. Because angle 3 is marked, plan P-LINEAR-PAIR-PAIR is triggered, which proves that angles 5 and 6 are congruent. Shortly thereafter, POLYA proves the corner triangles congruent using P-SAS-SIMPLE, which instantiates the side-angle-side theorem.

At this point POLYA makes the equivalent of a leap of reasoning. A very specialized plan, P-CORNER-TRIANGLES->ISOSCELES, is triggered by the combination of the shape of the diagram (isosceles with corner triangles) and the assertion that the triangles are congruent.[4] Though POLYA has not yet read the goal statement, this plan represents the knowledge one might have from having seen this problem before: the large triangle is almost certainly isosceles, and, furthermore, that is probably a key conclusion (though not necessarily the goal) in this problem. The plan steps through the argument, adding inferences to the proof: because the corner triangles are congruent, the corner angles are congruent (1 and 2), and therefore the sides of the large triangle are congruent.

Finally, POLYA reads the goal, discovering that the proof is complete.

Discussion

Theorem-proving is usually done from scratch, using a minimalist representation of the problem and the rules. In light of that tradition, what POLYA does in the problem above might seem like cheating.

In the foregoing example, POLYA makes use of two very specialized plans: S-CORNER-TRIANGLES and P-CORNER-TRIANGLES->ISOSCELES. If we remove the P-CORNER-TRIANGLES->ISOSCELES proof plan from memory, POLYA is still able to solve the problem, though it takes longer. (It cannot solve it without S-CORNER-TRIANGLES.) But to remove that plan would be contrary to the main point of this research.

The point is not to build a system which knows very little but can solve problems by working very hard. That has been done many times before. The point is to build a system which knows a lot and which can solve problems easily. The point is to model what an expert knows, including his memory of problems he has solved before. This is what POLYA's plans represent.

Conclusion

At current writing, POLYA has 68 search and proof plans. These plans constitute POLYA's knowledge about how to solve geometry proof problems, covering roughly one-fifth of the simple triangle congruence problems in a textbook and a very few examples involving parallel lines. Expanding POLYA's knowledge to cover quadrilaterals, other parallel line examples, and other aspects of geometry will at least double or treble the number of plans. Rapid growth of plan memory is not a bad thing, provided that we can continue to index the plans efficiently.

While many of POLYA's plans correspond fairly directly to general rules, some of POLYA's plans relate to specific problems in much the same way that a person might remember a specific problem. We believe that it is both necessary and appropriate that geometry expertise comprise knowledge at many levels of generality. When POLYA has a specific plan for a particular problem, fewer plans are needed and fewer actions are performed in solving that problem. Thus it is the case with POLYA, as with an expert, that the more knowledge it has, the more easily it can solve problems.

References

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

Ballard, D. H. 1991. Animate vision. *Artificial intelligence* 48.

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.

Clark, J. J. & Ferrier, N. J. 1988. Modal control of an attentive vision system. In Proceedings of the International Conference on Computer Vision.

Firby, R.J. 1989. Adaptive execution in complex dynamic worlds. Ph.D. diss., Yale University.

[4]Inferences are passed to memory in the same way that visual descriptions are, and can be used for indexing.

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. In Paris, S.G., Olson, G.M., and Stevenson, H.W., 1983. *Learning and Motivation in the Classroom.* 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. and Anderson, J.R. 1990. Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science* 14:511-550.

Larkin, J.H., McDermot, J., Simon, D.P., and Simon, H.A. 1980. Models of competence in solving physics problems. *Cognitive Science* 4:317-348.

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

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

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

Swain, M. J. 1991. Low resolution cues for guiding saccadic eye movements. *SPIE advances in intelligent robot systems.*