A recognition model of geometry theorem-proving

Tom McDougal
Kristian Hammond
University of Chicago AI Lab
1100 E. 58th Street
Chicago, IL 60637
(312) 702-1571
mcdougal@cs.uchicago.edu
hammond@cs.uchicago.edu

Abstract

This paper describes POLYA, a computer program that writes geometry proofs. POLYA actively collects features from a geometry diagram on the basis of which it recognizes and applies knowledge from known examples. We present a vocabulary of visual targets, results, and actions to support incremental parsing of diagrams. We also show how scripts can be used to organize visual actions into useful sequences. We show how those sequences can be used to parse diagrams and instantiate proofs. Finally, we show how scripts represent the implicit spatial knowledge conveyed by examples.

Introduction

Anyone who has struggled in a math class knows the difference between understanding the solution to a problem and knowing how to come up with that solution. In *How to Solve It* [1957], mathematician and educator George Polya gives advice to the student having trouble deriving solutions. He breaks the problem-solving process into four steps:

1. Understand the problem.
2. Devise a plan.
3. Carry out the plan.
4. Look back.

Most of the work occurs in the second step. There Polya recommends drawing on experience: "...It is often appropriate to start the work with the question: *Do you know a related problem?*" [p. 9, italics in original]

How to Solve It is peppered with examples from geometry. However, the history of geometry theorem-proving in AI and Cognitive Science contains little that resembles Polya's four steps (see [Koedinger & Anderson 1990] for a review). Instead, nearly all computer programs which construct geometry theorems do so using forward and backward chaining of if-then rules corresponding to the traditional theorems and axioms of plane geometry. The programs have differed from one another mostly in the heuristics used to make the chaining approach tractable.

This paper describes a computer program, POLYA, which constructs geometry proofs in the way George Polya suggests. It devises a plan by recognizing similarity to known examples; it carries out the plan by mapping significant features of the examples to the new case. POLYA addresses some of the limitations of Koedinger and Anderson's Diagram Configuration model (DC), resulting in a more cognitively plausible model of geometry theorem-proving.

This work was supported in part by the University of Chicago Department of Computer Science, AFOSR grant number AFOSR-91-0112, DARPA contract number F30602-91-C-0028, DARPA contract number N00014-91-J-4092 monitored by the Office of Naval Research, Office of Naval Research grant number N00014-91-J-1185, and an internal fellowship from UCSMP.

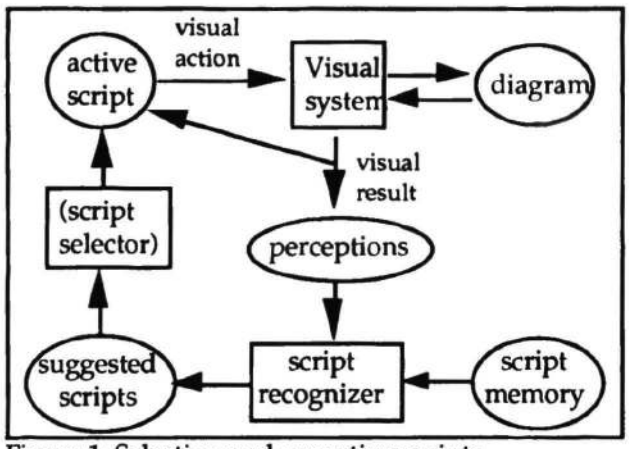


Figure 1: Selecting and executing scripts.

Overview of POLYA

POLYA accepts as input a problem statement and a diagram. The problem statement consists of a short list of "givens" and a goal. The diagram is a bitmap drawing. POLYA marks the diagram to reflect the givens, and then begins to parse the diagram.

POLYA parses the diagram incrementally, using *visual actions* to shift its focus of attention to areas of likely interest in response to what it sees. POLYA's knowledge of how to parse diagrams, as well as its knowledge of how to write proofs, is contained in *scripts*. *Search scripts* are sequences of visual actions for detecting a particular useful pattern or relationship in the diagram. *Proof scripts* contain a sequence of visual actions for verifying the relevance of an example, and another sequence for mapping the example to the current diagram.

POLYA's algorithm for selecting scripts and running them is shown in figure 1. The steps are:

1. Select a script.
2. Perform each action in the script, obtaining visual results. Add each result to a list of perceptions.
3. Compare the visual result to the prediction made by the active script. Also compare the list of perceptions to the triggering sets of inactive scripts.
4. Update the list of suggested scripts, adding ones which have had their triggering set satisfied.
5. Repeat until there are no more scripts or until the proof is complete.

The rest of this paper will describe POLYA's visual system, its visual search knowledge, and its proof-writing knowledge. We will illustrate with a working example how these interact to devise and carry out a plan.

POLYA's visual system

The design of POLYA's visual system is influenced by certain facts about the human visual system. The human visual field has a very small area (about 3 degrees) of high resolution at the fovea with much lower resolution elsewhere. This presents a computational advantage for processing retinal images, but presents challenges for gathering information. Much of the success of human vision is due to our ability to shift the fovea rapidly among areas of likely interest, with additional processing relating those narrow perceptions [Carpenter 1988]. These advantages and challenges have inspired new research in *active vision* [Ballard, Clark, Schwartz, Swain, Tistarelli, etc.].

Focus of attention

Matching configuration schema against a geometry diagram, as DC does in its first problem-solving phase, reduces to NP-complete subgraph isomorphism. The addition of a single irrelevant line segment to a diagram can double the time DC requires to parse it [Koedinger, personal communication]. Koedinger and Anderson [1990] acknowledge in their paper that an accurate model of human problem solving would integrate parsing and schema search.

This is what POLYA does. It parses the diagram opportunistically, using what it sees (*visual results*) to help it decide what to look at next (*visual targets*). Figure 2 contains excerpts from POLYA's vocabulary of visual targets and results. Note that POLYA can focus on pairs of objects, such as two triangles or a point and a segment, as well as on individual objects. In focusing on pairs of objects, POLYA loses specific information about the individual objects while gaining relational information. For example, looking at a pair of triangles yields no information about markings on the individual triangles; for that information POLYA must focus on one triangle at a time.

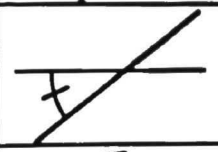
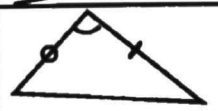
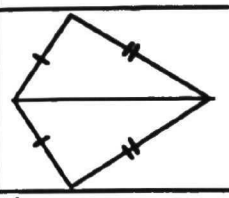
What POLYA can look at	What POLYA sees	Examples	Visual result
Single point	ray-pattern label no. of angle marks		X-HORIZ-UP LABEL-A 1-ANGLE-MARK
pair of segments	rel. lengths rel. extents rel. orientation		1<2 DISJOINT PARALLEL
triangle	number of sides marked no. of angles marked no. of interior lines basic shape		TWO-MARKED-SIDES ONE-MARKED-ANGLE 0-INTERIOR-LINES OTHER-TRIANGLE-SHAPE
triangle pair	symmetry rel. sizes rel. extents		T/B-SYMMETRIC APPROX-SAME-SIZE SHARED-SIDE

Figure 2: Some of POLYA's visual targets and visual results.

Visual actions

Currently, POLYA has three types of actions for shifting its focus from one target to another: FIND, LOOK-AT, and COMPARE. All three shift the focus and return a description (visual result) of the target object(s). FIND shifts the focus to a hypothesized object whose description is (partly) known but whose location is not known. FIND may return nil if no object matching the description exists in the diagram. LOOK-AT shifts the focus to an object whose location is known but whose description is not known. COMPARE shifts the focus to pairs of objects. FIND-MARKED-SEGMENT, LOOK-AT-LOWER-LEFT-VERTEX, and COMPARE-TRIANGLES are instances of the three action types.

As one reasonableness criterion, we intend that POLYA should be able to perform its visual actions directly on a bitmap diagram. Currently, however, POLYA computes its visual results from coordinate listings of points, lines, and segment and angle marks.

Geometric planning knowledge

POLYA's planning knowledge of where to look in a diagram, as well as its knowledge of how to write proofs, is contained in geometry *scripts*. These are modelled after the scripts described in [Schank & Abelson, 1977] and implemented in SAM for understanding newspaper stories [Cullingford 1978]. Scripts store routine actions and sequences of events so that, once a script has been selected, inferencing is tightly controlled.

POLYA has two kinds of scripts: *search scripts* and *proof scripts*. Both kinds of scripts have one or more *triggering sets*, sketchy lists of perceptual features which suggest the relevance of the script. A script becomes *suggested* when the accumulated perceptions contain all elements of the triggering set.

Search scripts

Search scripts direct the focus of attention to potentially salient parts of the diagram. They themselves do nothing with the visual results; their purpose is to gather the features needed to trigger proof scripts. The predictions usually serve only as a check on the success of the visual actions. ISOSC-LEGS search script (figure 3) directs the focus to the legs of a triangle which appears to be isosceles. POLYA has another, similar script which directs the focus to the base angles. The two scripts represent POLYA's knowledge of the important parts of isosceles triangles.

Proof scripts

Proof scripts may correspond to formal geometric facts—axioms, theorems, and properties—or to parts of complete proofs POLYA has seen. They

ISOSC-LEGS search script

Triggering set

DIAGRAM [left-right symmetry]

TRIANGLE [type = ISOSC-UP]

Sequence

1. Action = [look at left side of isosc. triangle]
   Prediction = [generic segment]
2. Action = [look at right side of isosc. triangle]
   Prediction = [generic segment]
3. Action = [compare the two segments]
   Prediction = [segments share common endpoint]

Figure 3: A script to look at isosceles triangles.

are similar to the Diagram Configuration schemas in DC. However, the proof-writing knowledge of POLYA's proof scripts is very specific and uni-directional, whereas DC's schema may package multiple rules for forward or backward inferencing.

Proof scripts have four parts:

1. A triggering set.
2. A *verification sequence.*
3. A *template-filler sequence* for locating objects needed for the proof.
4. A *proof template* .

Once activated by a match against the triggering set, proof scripts operate in three phases. First, the verification sequence checks that the relevant objects in the diagram are in the proper configuration for the proof script to be valid. Second, the template-filler sequence looks again at the diagram to determine variable bindings for the template. Third, the proof script instantiates its template with the variable bindings from the previous step.

The SSS-SHARED-SIDE proof script (figure 4) can prove that two triangles are congruent to each other if two pairs of sides are congruent and if they share a third side (see the triangle-pair example in figure 2). It is triggered on the basis of a triangle visual result in which two of the triangle's sides are marked, and a triangle-pair result in which the two triangles share a side. Because these results may be separated in time, there is no guarantee that they have anything to do with each other, so the first actions of the SSS-SHARED-SIDE script verify that each of the triangles which share a side have two sides marked. Additional actions locate the corresponding pairs of marked sides, and the shared unmarked side.

A complete proof generally requires more than one proof script, and is thus a composite of several instantiated templates. POLYA currently lacks a mechanism for organizing steps from multiple templates in logical order.

sss-shared-side proof script

Triggering set

DIAGRAM [some symmetry]

TRIANGLE [2 sides marked]

TRIANGLE-PAIR [shared-side]

Verification sequence

1. Action = [look at triangle1 of triangle-pair]
   Prediction = [2 sides marked]
2. Action = [look at triangle2 of triangle-pair]
   Prediction = [2 sides marked]
3. Action = [look at shared side of triangle-pair]
   Prediction = [unmarked segment]

Template-filler-sequence

1. Action = [look at triangle1 of triangle-pair]
   Prediction = [2 sides marked]
   Bind-to: ?TRIANGLE1
2. Action = [look at triangle2 of triangle-pair]
   Prediction = [2 sides marked]
   Bind-to: ?TRIANGLE2
3. Action = [find marked-side of ?triangle1]
   Prediction = [marked-segment]
   Bind-to: ?SIDE1A
4. Action = [look at symmetric partner of ?SIDE1A]
   Prediction = [marked-segment]
   Bind-to: ?SIDE2A
5. Action = [compare ?SIDE1A & ?SIDE2A]
   Prediction = [seg-pair similarly marked]
   Bind-to: —
   ....

Proof template

1. Statement = (?SIDE1A = ?SIDE2A)
   Reason = "As marked"
2. Statement = (?SIDE1B = ?SIDE2B)
   Reason = "As marked"
3. Statement = (?SHARED-SIDE = ?SHARED-SIDE)
   Reason = "Reflexive property (shared side)."
4. Statement = (?TRIANGLE1 = ?TRIANGLE2)
5. Reason = "SSS with shared side."

Figure 4: A script for proving two triangles congruent, if they have two sides congruent and share a third side.

Script selection

When more than one script is suggested, which is frequently the case, POLYA chooses a script essentially at random, except that preference is given to proof scripts. This is adequate for the current model, since the order in which perceptions are gathered does not affect the final solution.

Representing examples with scripts

Scripts capture the visual search and proof-writing knowledge implicit in examples and sample problems. Because this knowledge is rarely, if ever, explicitly taught, we have relied on a careful study of textbook examples to tell us what POLYA's scripts should contain.

When the triangle congruence theorems are introduced in one text [Rhoad et al., 1986], the diagrams in the examples and first several problems emphasize two types of patterns: triangles with two parts marked (sides, angles, or a combination), and triangles with three parts marked. The associated skills are, respectively, to identify what third side or angle would have to be marked for a particular theorem to apply (always two possible theorems), and to identify which one theorem (if any) applies to the given case.

To represent these skills, POLYA has search scripts whose triggering sets consist of triangle-visual-results with two angles marked, or two sides marked, or one side and one angle. They focus attention on the additional side or angle which would be needed for a particular theorem, and compare that object with its symmetric partner in the other triangle. The proof scripts for this section are triggered on the basis of three marks (or two marks and some significant object-pairing). Their verification sequences check that the marked objects are in the proper configuration.

An example

One problem POLYA can recognize is shown in figure 5, a textbook problem from a section which introduces isosceles triangles. To recognize this problem and write its proof requires four search scripts and two proof scripts. The search scripts are a script which looks at the dominant triangle, ISOSC-LEGS and ISOSC-ANGLES, discussed earlier, and CONGRUENT-TRIANGLES. CONGRUENT-TRIANGLES focuses POLYA's attention on the marked triangles in the lower-left and lower-right corners.

The features collected by those four search scripts complete the triggering sets for two proof scripts: SSS-SIMPLE and a proof script specific to this problem (3.6-PROBLEM-3). Of the two proof scripts, POLYA arbitrarily chooses to run 3.6-PROBLEM-3. As its verification sequence, this

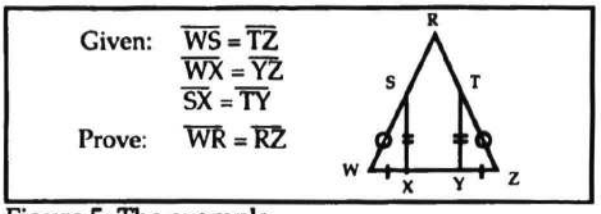


Figure 5: The example.

script COMPAREs the marked triangle with the dominant isosceles triangle, predicting (correctly) that they will share an angle.

The template contains two proof steps related to two geometry rules:

1. Statement = (?BASE-ANGLE1 = ?BASE-ANGLE2)
   Reason = "Corresponding parts of congruent triangles are congruent."
2. Statement = (?SIDE1 = ?SIDE2)
   Reason = "If two angles of a triangle are congruent, then the sides opposite them are congruent."

The sequence of steps for binding the template symbols is:

1. LOOK-AT-BASE-ANGLE1→ ?BASE-ANGLE1
2. (similarly for ?BASE-ANGLE2)
3. LOOK-AT-ISOSC-LEG1 → ?SIDE1
4. (similarly for ?SIDE2)

Step 1 in the template assumes that the smaller triangles are congruent. This has not yet been proven, but soon will be.

Now POLYA runs the SSS-SIMPLE proof script, which represents an iconic example of side-side-side triangle congruence. The template for SSS-SIMPLE looks like this:

1. Statement = (?SIDE1A = ?SIDE2A)
   Reason = "As marked"
2. Statement = (?SIDE1B = ?SIDE2B)
   Reason = "As marked"
3. Statement = (?SIDE1C = ?SIDE2C)
   Reason = "As marked"
4. Statement = (?TRIANGLE1 = ?TRIANGLE2)
   Reason = "SSS"

This template is filled in with the actions LOOK-AT-SIDE1, LOOK-AT-SIDE2, and LOOK-AT-SIDE3 on one of the corner triangles, and using FIND-SYMMETRIC-SEGMENT to locate the corresponding side in the other triangle. This template completes the proof for this problem.

Discussion

POLYA is not really "solving" the problem above, but merely recognizing it as a problem for which it knows the solution. This is the simplest type of reasoning from examples. Simple or not, however, the example shows that POLYA's visual vocabulary is adequate for representing both general patterns and specific solutions. The example also suggests that search scripts can be used for efficient diagram parsing. Finally, the example shows shows how multiple search scripts and proof scripts can interact to recognize and write the proof of a known problem.

Conclusion

This paper has presented a new model of geometry theorem-proving consistent with George Polya's steps of problem-solving. POLYA constructs a proof plan and carries it out through visual search and recognition. To support this behavior we have defined a visual system—a vocabulary of visual targets, results, and actions—for representing and interacting with diagrams. We have shown how search scripts and proof scripts can be used to direct a constrained focus of attention for efficient visual parsing and for writing proofs. We have described an algorithm that allows smooth interaction of search scripts and proof scripts.

POLYA's visual system is still evolving, especially the vocabulary of visual actions. As we add examples we find that new actions are required, or at least new ways of describing them. Generally the new actions stem directly from new geometric concepts. For instance, to handle isosceles triangles requires knowledge of how to focus on the base angles. A full domain knowledge of geometry will comprise a large set of such context-specific visual actions.

We are anxious to test our model on a much larger set of examples with a much larger set of scripts. We expect that additional examples will reveal gaps in our vocabulary of visual actions. We expect the set of visual actions to grow quickly for awhile, then level off to a stable vocabulary of the visual skills required to parse plane geometry diagrams.

Thus far the emphasis of our research has been on reasoning from the diagram; POLYA currently ignores the goal in the problem input. However, we plan to incorporate some goal-directed reasoning in POLYA. Certainly this is something people do, and POLYA will need it to solve more complicated problems.

Finally, we are interested in the expanding POLYA into a case-based system which can learn new cases in response to difficulties experienced during problem-solving.

Acknowledgements

Many thanks to Paul Schiffer for his fine-tooth editing of earlier drafts of this paper.

References

Ballard, D. H. (1991). "Animate vision." *Artificial intelligence*, Vol. 48.

Carpenter, R. (1988) *Movements of the eyes.* Pion.

Clark, J. J. & Ferrier, N. J. (1988). "Modal control of an attentive vision system." *Proceedings*, International Conference on Computer Vision.

Cullingford, R. (1978). *Script application: Computer understanding of newspaper stories.* Ph.D. Dissertation, Research Report #116. Computer Science Dept., Yale University.

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

McDougal, T. (1988) "A computational model for the structural comparison of secondary plane geometry problems." M.A.T. Thesis, University of Chicago.

Polya, G. (1957). *How to solve it: A new aspect of mathematical method, 2nd Ed.* Princeton University Press.

Rhoad, R., Whipple, R., and Milauskas, G. (1986) *Geometry for enjoyment and challenge.* McDougal, Littell.

Rojer, A.S. and Schwartz, E. L. (1990) "Design considerations for a space-variant visual sensor with complex-logarithmic geometry." *Proceedings*, Int'l Conference on Pattern Recognition.

Schank, R., Abelson, R. (1977). *Scripts, plans, goals, and understanding.* Lawrence Erlbaum.

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

Tistarelli, M. & Sandini, G. (1990). "On the estimation of depth from motion using an anthropomorphic visual sensor." *Proceedings*, European Conference on Computer Vision.