How to Teach it – Polya-Inspired Scenarios in ACTIVEMATH

Erica Melis and Carsten Ullrich
DFKI and Saarland University, 66123 Saarbrücken, Germany,
melis,cullrich@activemath.org

Abstract. We adopt Polya's heuristics to scaffold problem solving and learning. In particular, in ACTIVEMATH adopting Polya's framework provides structure (and subgoals) for certain presentation-scenarios and prompts for scaffolding problem solving.

Keywords: course generation, meta-cognition, problem solving heuristics

1 Introduction

Building on ideas reaching back to Descarte and Bolzano and even earlier, Polya suggested a framework for teaching meta-reasoning in mathematical problem solving. In his famous book "How to Solve It" [9] Polya made explicit a set of strategies for solving mathematical problems. While Polya's work was mostly concerned with the challgenge of *finding* a proof, his heuristics can also be interpreted for *presenting* a proof – more general a problem solution – in a way that includes context and other relevant information rather than the pure proof or solution.

But why stop here? Can't we generalize Polyas ideas and use them for teaching the overall mathematical subject matter? Furthermore, why not investigate a user-adaptive and dynamic version of Polya's heuristics, i.e., ask which suggestions are suited for which user under which circumstances.

In this paper, we suggest several possibilities of adopting Polya's strategies for intelligent tutoring systems. The paper starts with a overview on Polya and on the ACTIVEMATH system. It continues with adoptions of Polya for various purposes in ACTIVEMATH and finally, discusses related work.

2 ActiveMath

ACTIVEMATH is a web-based learning environment, currently used for mathematics.

Its learning materials are represented as a set of related learning objects rather than as (a sequence of) predefined (HTML) pages. From these objects the course presentations are dynamically assembled. The learning objects are representend in a semantic (XML) knowledge representation annotated with metadata. The presented courses can vary depending on the learning scenario, the learner's knowledge and her preferences. For instance, for content she knows pretty well, less learning materials is presented than for content she barely knows. Available scenarios are, for instance, "Guided Tour"or "Exam Preparation".

That is, the authors write the learning materials and a *course generator* assembles courses using pedagogical rules. By this separation of content and pedagogical knowledge, the learning materials can be reused in ways an author has not even thought of.

ACTIVEMATH also contains several tool to support interactive learning, for instance, a dictionary that displays the dependencies between the learning objects and concepts, and a suggestion mechanism.

ACTIVEMATH presents content by adaptive hypermedia[1] in a way that makes it simple to provide information on demand because students can interact (e.g., fold/unfold) presentations.

Knowledge Representation In ACTIVEMATH, the learning objects that can be represented are concepts such as definition and theorem, and additonal elements for these concepts, such as proof, proof method, example, exercise, motivation, introduction, or elaboration (for an exhaustive description of ACTIVEMATH knowledge representation see [13]).

All learning objects can be annotated by pedagogical metadata, such as `abstractness` and `difficulty`, as well as several kinds of relations between concepts (`for`, mathematical `dependency`, pedagogical `prerequisites`, `references`, `similar`). Additional metadata serve to describe exercises (e.g., the `pedagogical goal` such as knowledge, comprehension, application, or transfer [1]). Furthermore, if an author wishes, he can represent the internal structure of the learning objects. Consider the following example:

- *Let's take a look at the set of real numbers with the addition operation.* – situation descriptions.
- *This structure is a monoid.* – assignment of a property
- *Indeed, the addition operation is associative and possesses a unit, the number 0.* – Proof, i.e., problem solution

By sharing (parts of) the situation description, other examples (i.e. the real numbers with addition operation) can reuse the same object description. Similarly, exercises can share the problem-statement (e.g., "Proof the triangles are congruent."). More examples of the use of sharing structure will be shown later in the description of the scenarios.

3 The Polya-Guided Problem Solving

Polya's books are a rich source of inspiration for teaching problem solving. 'How to Solve It' [9] has the form of a how-to manual. It is a formulation of a set of heuristics cast in form of brief questions and commands within a frame of four problem solving stages:

1. Understand the Problem
2. Devise a Plan
3. Carry out the Plan
4. Look Back at the Solution

Some questions and commands Polya uses in the respective phases are

1. What is the unknown? What are the data?
2. Do you know a related problem? Did you use all the data?
3. Can you see/prove that each step is correct?
4. Can you check the result? Can you use the result for some other problem?

[1] The content is alternatively available in a print format which is, however, not relevant here.

The activities surrounding the actual problem solving process (i.e., Carry out the plan) are typical meta-reasoning activities for problem solving.

Polya's stages augmenting the actual problem solving process (which is essentially captured in 'Carry out the plan') model typical meta-reasoning activities for problem solving and can also be interpreted as a structure for learning. Therefore, they can serve as a basis for principles of instructional design.

Reception of Polya's Heuristics in Artificial Intelligence In AI-research, Polya's heuristics became a challenge for automated problem solving. However, as Newell [8] summarized, these heuristics are too general to be implemented and automatically applied.

This is, however, no argument for disregarding Polya in a learning environment, because here structures and prompts are made for a human student who will interprete the heuristic cues in order to find a solution rather than for a machine. So the structure and cues can still scaffold the student's learning even if not representing exactly one formal step. Moreover, Polya's stages can be supported by a learning system that offers related information which the user can pick. And finally, already solved worked-out examples can be enriched by such phases in order to provide a big picture and in order to teach meta-reasoning.

Reception of Polya's Heuristics in Psychology of Mathematics The literature of mathematics education is full of heuristic studies. Most of these, while encouraging, have provided little concrete evidence that heuristics have the power that was promised, see e.g.,[15, 12, 4]. The attempts to teach these strategies have been met with mixed success because: (1) heuristic strategies are labels for *classes* of strategies whose elements may not be all available to a student. (2) Training in the use of strategies must involve training of all phases, has to be precise and rigorous. (3) Although heuristic strategies can serve as guides to relatively unfamiliar concepts or problems, they do not replace subject matter knowledge. The success heavily depends on the resources available to the student such as knowledge about domain, facts, definitions, algorithmic procedures, routines, competencies.

Extending Polya's ideas and building on a firm empirical ground, Schoenfeld [10, 11] stresses the importance of meta-cognition that includes not just planning but also chosing subgoals, monitoring partial solutions, and revising a plan if necessary. Certainly, such meta-cognition is not just crucial for mathematics but generally for problem solving and learning.

4 Application of Polya in ACTIVEMATH

Polya's structured presentation and scaffolding is not yet included in any of today's mathematics tutoring systems. His ideas can be realized in different ways by intelligent tutoring systems among them

- *Polya-Proof-Scenario* principled structured presentation of augmented proof examples
- *Polya-Example-Scenario* more general: structured presentation of augmented problem solutions
- *Polya-Exercise-Guidance* structured guidance for problem solving inquiry cycle
- *Polya-Course-Scenario* structured presentation of (static) learning material
- *Polya-Suggestions* in dynamic suggestions for the learning process.

Since proof is central in mathematics, one of the first scenarios we built is a Polya-Proof/Example-Scenario in which Polya's stages are interpreted and assembled for the presentation of worked-out solutions. This is described in Section 4.1. In Section 4.3 we describe how we adopt Polya for problem solving exercises. There the stages can structure the inquiry activities of the student and Polya's (or similar) prompts can guide the student in (mathematical) problem solving. The adoption of Polya's structure also serves as a model for generating material for a particular learning scenario in ACTIVEMATH as described in Section 4.2.

Polya's suggestions build a framework around the actual object-level problem solving with the goal to guide and restrict the search and to support later usage of the problem experience (learning). A solving service system that helps students to solve a problem works at the object-level. Therefore, it would be difficult and not natural to merge such a tool with Polya's framework. However, appropriate service systems, e.g. a proof planner, can support the student in the phases *Devise a Plan* and *Carry out the Plan*. They can act as a cognitive tool and additionally make relevant information explicit, such as the collection of constraints on a mathematical object, methods, and expansions of plans [5].

4.1 Polya-Proof/Example-Scenario

This scenario presents proof *examples* in a Polya-framework. That is, this scenario augments actual worked-out proof and puts them into a larger learning context. Here, the stages can be either explicitly displayed for structure and is a model for exercises or can be kept implicit in the presentation.

As the following Table displays, ACTIVEMATH' scenario adopts Polya's stages (Understand the Problem, Devise a Plan, Carry out the Plan, and Look Back at the Solution) by assembling certain types of learning objects (*italic font*) and considering certain metadata (`typewriter font`). Which objects are actually assembled (e.g., the difficulty of objects may vary, the user may have seen some objects and not others previously) also depends on the user model and therefore, the scenario is user-adaptive as most other scenarios of ACTIVEMATH.

Problem	The *theorem* to be proved.
Understand	*motivation, figure, situation description* `for` the theorem *theorem* and *concept* the proof `requires`
DevisePlan	`similar` *proofs* `lemma-for` the problem (`proofPlanner`), `abstract` *proof*
CarryOutPlan	expand *method* (`proofPlanner`), `concrete` *proof*
Examine	*method* `for` other proofs different *proof* `for` the theorem

The dynamic aspects of the presentation cannot be shown in the table. The stages can be displayed one by one and inside the single stages a dynamic presentation is possible too. For instance, DevisePlan can dynamically be presented in a way that, among others, first shows just a skeleton of the proof that only contains the conjecture and the given assumptions and then stepwise introduces further lemmas and methods into the plan (and maybe even more information). Similarly, CarryOutPlan can be dynamic, e.g., by fold/unfold facilities. In case the proof planner integrated into ACTIVEMATH is used to demonstrate the planning

(DevisePlan) or expansion (CarryOutPlan) its output is dynamically added anyway. We have a prototypical implementation of a dynamic presentation of proof in ACTIVEMATH which is, however, not subject of this paper.

Not just proofs examples can be presented in such a scenario but also other kinds of (augmented) problem solving examples. This requires authors to provide certain information and augment instructional elements with metadata in the first place.

No cues or questions are generated and used in this scenario yet because currently, ACTIVEMATH just demonstrates the structure of a worked-out solution and provides the relevant information. However, similar to self-explanation prompts [3], we could introduce requests prompting the student to actively construct the context information (e.g., with the help of ACTIVEMATH' dictionary which elicits dependencies etc.) Such requests include: check each plan step (in CarryOutPlan), mark the known/unknown (in DevisePlan).

4.2 Polya-Course-Scenario

This scenario presents (mathematical) concepts/a course in a Polya-inspired problem-oriented way.

1. **Understanding the Problem/Context:** Present a problem that illustrates the relevance of the to be learned concepts. Let student compare it to related already known problems. Provide missing prerequisites.
2. **Devising a Plan:** Present the concepts necessary to solve the problem. Present (worked-out) examples and counter-examples and relevant methods.
3. **Carry out the Plan:** Present exercises.
4. **Looking Back:** Promote integration by linking back/out and by presenting analysing and evaluating exercises. Present similar concepts and let student compare them. Prompt the learner to use ACTIVEMATH's dictionary and the concept map.

The mapping of these general principles into ACTIVEMATH' metadata is realised as follows:

Understanding the Problem	*theorem*/*exercise* with `type` equal `problem`. `Similar`, already solved problems. Unknown or barely known *definition*s and *theorem*s that are `prerequisites` of the new concepts.
Devising a Plan	*Concepts* the problem is `for`. `(counter)` *examples* `for` the concepts.
Carry out the Plan	*excercises* `for` the new concepts.
Looking Back	Already known `similar` *theorem*s. *examples* with/without new property. Important *theorem*s and *definition*s that `depend-on` learned material. *excercises* `for` new concepts with `type` analyse/evaluate. Hyperlinks to concept map and dictionary.

As an example, let's take a look at a course that ACTIVEMATH generates for Anton, a beginner in the domain of probability theory. He just started to learn the basics and now comes to solve concrete problems. The course generator generates the following scenario for him, taking his current knowledge into account:

1. **Understanding the Problem:**
   - Problem: The German lottery draws 6 numbers ouf of 49. How big are your chances of winning?

- Already solved problems: Horse racing (underlying related mathematical concept: “ordered pairs without replacement”)
- Barely known definition: “Factorial”

2. **Devising a Plan:** Concepts “binomial coefficient” and “unordered pairs without replacement” and examples.
3. **Carry out the Plan:** Solve the given problem. Additional exercises such as “How many different five card hands are possible in the game of poker?”
4. **Looking Back:**
   - Already known similar theorems and definitions: “ordered pairs without replacement” and “ordered pairs with replacement”
   - Important definitions that depend on learned material: “unordered pairs with replacement”, “mean of a distribution”

4.3 Polya-Exercise-Guidance

Similar to other inquiry learning contexts, mathematical exercises can be supported by a scaffolding structure that includes questions and promts. This is a first step towards a meta-cognitive support and can later be extended to a more flexible and user-adaptive scaffolding. To begin with, we devised the fill-in form displayed in Figure 1 that is presented with exercises.

We would also like to introduce an activity revisePlan into the problem solving cycle. It is missing in Polya’s stages. A problem is that revising a plan fits into several stages of active problem solving: in DevisePlan (during planning it might turn out that an original partial solution plan will not solve the problem), in CarryOutPlan (when carrying out the plan it might turn out that something does not work out as expected and therefore, the plan has to be revised), as well as in Examine (when trying to generalize, it might turn out that the original plan or method has to be revised to be applicable elsewhere).

4.4 Polya-Suggestions

The next step will be to include Polya-like suggestions into ACTIVEMATH’ global feedback which aims at supporting the learner’s meta-reasoning anyway [6].

These suggestions will issue prompts (with suggestions) similar to those in Figure 1 but not just targeted at solving one problem but at the current learning situation more generally, i.e., the learning problem. That is, instead of asking “Can you derive the result differently?” the question would be “Can you find a different way to explain concept xxx?”. Instead of “Find the connection between the given and the unknown!”the prompt would be “Find the connection between the previous concept xxx and the current concept yyy”.

5 Related Work

ThinkerTools [14] offers a somewhat different framework with its structured guidance for experimentation in science.

Cairns [2] adopts Polya’s stages through his interpretation that gives a fix hypertext structure surrounding the actual example proof. [2] The presentation variance is restricted to foldable

[2] `http://www.uclic.ucl.ac.uk/topology/`

- **Understand the problem**:
  - What is the unknown? What are the data? What is the condition? Write down the different parts of the condition. Is it possible to satisfy the condition?
  - Are there contradictions or redundancies in the problem?
  - Draw a figure if possible!
- **Devise a plan**:
  - Find the connection between the given and the unknown!
  - Have you seen such a problem before or maybe in a slightly different form? Do you know a similar or related problem? Could you use this related example, its result, its method?
  - Do you know a theorem or method that could be useful?
  - Did you use all the data?
  - In case you cannot solve the problem: Find a (simpler) related problem (a more general or more special one or an analogous problem)! Keep one part of the condition and drop another part! How can you vary the problem such that a plan can be found? You may be obliged to consider auxiliary problems or auxiliary variables! Could you restate the problem? Go back to the definitions!
  - You might be able to use the Proof Planner.
- **Carry out the plan**:
  - Check whether each step is correct!
  - Can you prove the correctness? If not, how to revise the plan?
  - You might be able to use the Proof Planner.
- **Examine the obtained solution**:
  - What is the general method or idea?
  - Can you use the result or method for some other problem?
  - Can you derive the result differently?

Figure 1: A Polya-outline for scaffolding the inquiry cycle of exercises.

proofs. It is not user-adaptive and not automatically assembled. Moreover, Cairns does not use questions or prompts in the presentation.

Instructional design offers a multitude of frameworks that structure teaching and learning. For instance, Merrill identifies the following stages necessary for successful learning: *Problem*, *Activation*, *Demonstration*, *Application*, and *Integration* [7]. However, most of these frameworks serve as a guidance for the creators of learning materials and therefore are to abstract to be implemented in an ITS.

6 Conclusion

Polya suggested a framework for teaching meta-reasoning in mathematical problem solving. This has been extended, e.g. by Schoenfeld, but not been implemented in an learning system for mathematics, previously.

Now, ACTIVEMATH offers the first Polya-like exercises scaffolding outline as well as proof- and course-scenarios. Soon it will be extended to dynamically scaffold students with a Polya-like suggestion strategy. The context of the user-adaptive system ACTIVEMATH responds to the needs for a success of those heuristics: (1) heuristics can be made available for a student; (2) the training in the use of strategies can involve training of all phases; (3) subject matter knowledge is taught by the system rather than exercises only.

References

[1] B.S. Bloom, editor. *Taxonomy of Educational Objectives: The Classification of Educational Goals: Handbook I, Cognitive Domain*. Longmans, Green, Totonto, 1956.

[2] P. Cairns and J. Gow. On dynamically presenting a topology course. In *First International Workshop on Mathematical Knowledge Management (MKM 2001)*, September 2001.

[3] C. Conati. *An Intelligent Computer Tutor to Guide Self-Explanation While Learning from Examples*. PhD thesis, University of Pittsburgh, 1999.

[4] M.G. Kantowski. Processes involved in mathematical problem solving. *Journal for Research in Mathematics Education*, 8:163–180, 1977.

[5] E. Melis. Why proof planning for maths education and how? In D. Hutter and W. Stephan, editors, *Festschrift in Honor of Jörg Siekmann*, number - in Lecture Notes in Artficial Intelligence, pages –. Springer-Verlag, 2003.

[6] E. Melis and E. Andres. Evaluators and suggestion mechanisms for ACTIVEMATH. Technical report, DFKI, 2002.

[7] M. D. Merrill. First principles of instruction. *Educational Technology Research & Development*, 2001.

[8] A. Newell. The Heuristic of George Polya and its Relation to Artificial Intelligence. Technical Report CMU-CS-81-133, Carnegie-Mellon-University, Dept. of Computer Science, Pittsburgh, Pennsylvania, U.S.A., 1981.

[9] G. Polya. *How to Solve it*. Princeton University Press, Princeton, 1945.

[10] A.H. Schoenfeld. What's all the fuss about metacognition? In A.H. Schoenfeld, editor, *Cognitive Science and Mathematics Education*, pages 19–216. Erlbaum, GHillsdale NJ, 1987.

[11] A.H. Schoenfeld. *Learning to Think Mathematically: Problem Solving, Metacognition, and Sense Making in Mathematics*, chapter 15. McMillan Publ.Company, New York, 1992.

[12] J.P. Smith. *The Effect of General versus Specific Heuristics in Mathematical Problem-Solving Tasks*. PhD thesis, Columbia University, 1973.

[13] The ActiveMath Group. Knowledge representation and management in ACTIVEMATH. To appear in the Proceedings of MKM'01 in Annals of Mathematics and Artificial Intelligence, 2003.

[14] B.Y. White and J.R. Frederiksen. *Inquiry, Modeling, and Metacognition: Making Science Accessible to all Students*. Lawrence Erlbaum, 1998.

[15] J.W. Wilson. *Generality of Heuristics as an Instructional Variable*. PhD thesis, Stanford University, 1967.