Why Proof Planning for Maths Education and How?

Erica Melis

DFKI Saarbrücken, 66123 Saarbrücken, Germany
melis@dfki.de, http://www.ags.uni-sb.de/~melis

1 Introduction

Artificial Intelligence techniques have massively been applied for Intelligent Tutor systems (ITS), e.g., user modeling, error diagnosis, user adaptation, knowledge representation, and dialog techniques. In this paper, I will argue in favor of the application of another AI-technique in ITS, namely of proof planning, a methodology from automated theorem proving.

The attempt to employ proof planning for educational purposes is not self-evident. It requires an understanding of what mathematics education needs and what proof planning offers.

When I tried to convince teachers or mathematics professors of using proof planning for teaching and, in particular, of using our proof planner ΩMEGA, then most of them were very sceptical to begin with. Only slowly, when they had seen several examples, they finally could understand the general advantages and agree that using proof planning can be a goal for supporting learning in a constructivist direction. [1] What are the main reasons for the skepsis? *First* of all, handling current proof planners is not made at all for students who learn mathematics or even for teachers used to mathematical argumentation and this heavily inhibits the usability and discourages an actual usage. In particular, the user interface and the support for interactive theorem proving by feedback and other functionalities is still insufficient.

Secondly, to learn a proof planning language to handle yet another system requires an additional effort for teachers and for students.

Third and maybe most importantly, many teachers and even mathematicians are not used to think about teaching mathematical proofs and problem solving in a non-traditional way. They themselves have learned mathematics traditionally, by listening lectures, following a sequential description of proof steps in textbooks, and finally somehow constructing the representation of knowledge in their minds from many examples *implicitly* containing structures, methods, and heuristics used in the proof processes. Although modern didactical and cognitive theories and empirical evidences show that (deep) learning is little supported by the traditional style of teaching, good examples for a modern teaching of theorem proving are rare. Geometrical proof is an exception because building hypotheses

[1] 'constructivist' used in the epistemological sense of *constructing* knowledge in a user's mind [27] rather than the logical meaning of the word.

and visual clues are supported by dynamic geometry systems such as Geolog [13] and Cinderella [29].

Fourth, many mathematics professors do not believe that learning heureka steps can be supported by a system.

Given these heavyweight opinions, it takes a lot to convince teachers of the benefits a proof planning approach can have for learning. However, my experience shows that teachers and maths professors *can* be convinced by good examples from different mathematical areas that show how learning can be supported at points, where the teachers experienced bottlenecks for understanding and transfer of mathematical proofs in their courses. Once convinced, they even accept the additional effort. This has been similar for other mathematical tools such as calculators and computer algebra systems (CASs) for which one has to learn an input language. These systems are widely accepted by now even in schools.

Since good examples are crucial for convincing the potential users, this paper will not only discuss the benefits of the proof planning methodology for learning mathematics but also illustrated them with examples, in particular some I have been using successfully in my 'conviction activities'.

A warning before we go into medias res: although I shall advocate a support that makes explicit the heuristics and the common structure of a class of proofs, I do not want to replace beautiful singular heureka ideas and creative acts in mathematical problem solving and proving that we love so much and that create some of the beauty of mathematics that I admire personally. In the opposite, this paper investigates why common methods, structures, meta-level knowledge, and heuristics uncovered by the proof planning methodology can support the learning of standard textbook mathematics. In addition, I shall argue how an interactive system can improve the fun, motivation, self-responsibility, and self-guidance of learners – pedagogic goals that are generally pursued by any kind of advanced learning.

For Jörg I want to re-phrase and summarize: Machines can think mathematically, at least together with humans and therefore, AI-machines can support meat-machines in learning mathematics. And, in order to have a real impact, this requires an interdisciplinary effort from AI, Cognitive Science, and empirical pedagogy. Therefore, this article has an interdisciplinary flavor.

2 Why is Improvement Necessary in the First Place?

To start with, let me reflect upon the current situation in mathematics teaching and upon the ways for improvement and new directions suggested by advanced educationalists and cognitive scientists. Subsequently, I will show *why* proof planning can contribute to some of the ingredients of a more appropriate learning and teaching; provide examples for *how* proof planning can be beneficial for learning mathematics; and finally discuss which improvements are necessary in order to make proof planning a methodology that real students and teachers will be able to use beneficially.

During the last decades, the mathematics pedagogy community recognized that students learn mathematics more effectively, if the traditional rote learning of formulas and procedures is supplemented with the possibility to explore a broad range of problems and problem situations [32, 18]. In particular, the international comparative study of mathematics teaching, TIMSS [4][2], has shown that teaching with an orientation towards active problem solving and an encouragement to find different solutions yields better learning results in the sense that the acquired knowledge is more readily available and applicable especially in new contexts. Moreover, several investigations show that a reflection about the problem solving activities and methods yields a meaningful learning [11, 2] that a training of meta-cognitive skills is a basis for meaningful and for lifelong learning [1].

There are at least four avenues to an improved mathematics learning: (1) the *active construction* of knowledge in the learner's mind, combined with a (2) systematic and structured teaching of *heuristic and systematic* knowledge for problem solving steps and *processes*, (3) the support of meta-cognitive reasoning, and (4) the appropriately structured presentation of problem solutions. The application of proof planning and its augmentation by a meta-cognitive cycle has the potential to contribute to all four ways of improvement as I shall show in the remainder of this article.

3 Active Learning

Active learning, as opposed to pure instruction, is one of the educational conclusions of the constructivist learning paradigm that goes back to Piaget [27] and Vygotsky [36]. The construction of knowledge in a student's mind requires to learn in a context, to be able to make mistakes and to learn from erroneous proof attempts, and to deeply understand the involved objects and their relationships. Moreover, the student has to experience "proving" as a process in which she engages during learning and "proof" as a (mere) product.

How can the learner's active participation be supported? As in other sciences, cognitive tools can provide a supporting machinery. The term *cognitive tool* was coined in [16] and denotes instruments explicitly supporting or representing cognitive processes and thus extending the limits of the human cognitive capacities, e.g., the working memory. When applied to learning, such tools can help, e.g., to

[2] The same seems to apply for the recent OECD study, PISA 2000, which looked at mathematical literacy of students measured in terms of student's capacity to

- recognize and interprete mathematical problems in everyday life
- translate these problems into a mathematical context
- use mathematical knowledge and procedures to solve problems
- interprete the results in terms of the original problem
- reflect on the applied methods and
- formulate and communicate the outcomes.

remember, to practice, to hypothesize, to solve a problem. In particular, when learning is difficult because it is too complex or because several things have to be done at the same time, these tools can help considerably. This is well-known for simulation tools [15], dynamic geometry systems, CASs, visualization by animations, and for tools that can impose a structure on a reasoning process [37, 34].

Let me summarize *why* proof planners and the integrated Computer Algebra Systems can support active learning of theorem proving and problem solving. They help the learner

- to *explore* a problem interactively and *directly experience* the result of a tentatively applied method or theorem. This is similar to what is possible in programming, where debugging is a most significant source of learning. The proof planner can *check the correctness* of the student's problem solving steps. It can help with feedback on where it is promising to explore and where dead ends are reached in a proof attempt;
- to *focus* on a particular subtask or skill in solving a problem. For example, the user of the system should not fail to calculate the limit of a function just because she cannot factorize polynomials but a CAS can perform this task. Similarly, the user should not fail to find the overall proof just because she cannot prove a minor subgoal;
- by presenting heuristics used, e.g. for intelligent backtracking, by explicating methods (see section 4), and also by providing an intelligently designed preselection and ordering of methods.
- to keep an overview of the proof attempt, e.g., list the not yet proved conjectures, present the resulting partial proof multi-modally.

4 Learning by Interactive Proof Planning

After the short answer on *why*, I want to explain *how* proof planning can help in learning.

A mere check of correctness could be performed by traditional automated theorem provers (ATPs), such as OTTER. However, there is more to proof planning which provides information that is essential for learning and for performing a mathematical proof: a sense of direction in searching for a proof plan, the theory-reasoning that helps to find instantiations for mathematical objects, and the use of mathematical methods which represent *mathematical* steps whose abstraction-level is usually different from the *logical* steps applied by an ATP.

An analysis of traditional textbook proofs gives rise to appreciate the information produced in proof planning. In such an analysis [17] Leron observed that the linear mathematical proofs occurring in most (text)books are just a *minimal* or even sub-minimal 'code' for transmitting mathematical knowledge that mature mathematicians are able to decode which is, however, missing important information, e.g. the proof idea. He found in his classes that many students are simply unable to decode those proofs and, therefore, their actions are reduced to meaningless manipulation of the 'code'.

The additional information that proof planning offers consists of methods and their explanation at different levels of abstraction, mathematical search heuristics, and the systematic construction of mathematical objects. This information is rarely taught and exercised in college-level mathematics, currently. We believe that teaching mathematical methods and proof know-how and know-when has to be introduced into mathematics teaching as an augmentation of the traditional teaching of axioms, theorems, and procedures. Indeed, first experiments suggest that instruction materials based on the description of proof planning *methods* yield a better subsequent problem solving performance than traditional (textbook-like) instruction material [23].

The idea to use proof planning in an educational context was also a reason for developing the Barnacle system [20], an extension of the proof planner $CLAM$. It extended $CLAM$ mainly by an interface that is more accessible and tried to convey the rippling heuristic for proofs by mathematical induction.

4.1 Method Knowledge

In his foreword to *how to solve it* [28] Ian Stewart writes: n order to be able to select the relevant information, and make use of it, mathematicians spend a great deal of their time acquiring both, a broad background and a repertoire of more specific tricks.

In our research for proof planning, we tried to acquire those general as well as specific methods for a restricted area of mathematics [22]. Aside from such general 'background methods' as `Case-Split`, `UnwrapHyp`, and `TellCS` we designed the `ComplexEstimate` method whose idea is a modification and generalization of the limit heuristic in [5], as well as other estimation methods implicitly used in many proofs in calculus.

A simple variant of `ComplexEstimate` was used in the instruction material of the empirical research reported in [23] and mentioned above. An explanation of this method goes as follows:
`ComplexEstimate` proves the estimation of a complicated term $|b|$ by representing b as a linear combination $k * a\sigma + l$ [3] of a term a whose estimation is already known and by reducing the goal $|b| < \epsilon$ to simpler subgoals that contain $\mathbf{M}$ – a positive real number whose existence is postulated by `ComplexEstimate`. The subgoals are

1. $|k| \leq \mathbf{M}$,
2. $|a| < \epsilon/(2 * \mathbf{M})$,
3. $|l| < \epsilon/2$.

The rationale behind the reduction to the subgoals is contained in the proof schema of `ComplexEstimate` in which the subgoals, the Triangle Inequality, and monotony properties are employed to prove $|b| < \epsilon$.

For instance, in planning LIM+, at some point the goal
$|f(x) + g(x) - (l_1 + l_2)| < \epsilon$ is reduced by `ComplexEstimate` to

[3] where σ is a substitution

(1) $|1| \leq \mathbf{M}$,
(2) $|f(x) - l_1| < \epsilon/(2*\mathbf{M})$,
(3) $|g(x) - l_2| < \epsilon/2$.

`ComplexEstimate` is not necessarily to be used in its full generality. It would make sense to vary the explanation (and maybe even the instantiation of the method in the actual proof process), if the coefficients k or l have the trivial values $k = 1$ or $l = 0$ because this makes the explanation easier to follow.

By making this method explicit the student can be supported in understanding the rationale behind the otherwise correct but senseless manipulation of the term b and in generalizing simple proofs that are special cases with $k = 1$ or $l = 0$ which might be solvable without knowing a systematic procedure. If equipped with this 'trick' and its rationale, it is easier for students to find proofs themselves as we have shown empirically [23].

4.2 Heuristic Knowledge

An analysis of Polya's famous heuristics for mathematical problem solving [28] shows that these heuristics are guidelines and very general strategies rather than rules. Dependent on the concrete situation, they have to be expanded to a set of concrete operations if possible [31]. Moreover, they comprise at least two levels of heuristics: problem solving (search) heuristics and meta-cognitive heuristics. In §6.1 we discuss how meta-cognitive heuristics can be used to support students in the future.

For problem solving Polya suggests to try the following complex heuristics (strategies):

- reformulate the problem
- find a special case to work on first; introduce an auxiliary variable or an auxiliary problem
- decompose the problem into subproblems

Some general problem solving heuristics such as 'decompose into subgoals', 're-represent goal', 'look for a common proof structure' are inherent in proof planning and methods already. Other proof heuristics are (or can be) represented by control rules in ΩMEGAs proof planner. Control rules have been used for all kinds of search control in ΩMEGA [21]. More recent ideas for guiding the search by control rules are due to Andreas Meier. They include, e.g., the instantiations of variables, and backtracking control. The backtracking control employs heuristic knowledge on when search branches are likely to fail and can enforce backtracking even if there are still applicable methods. It also can suggest, where to backtraqck to.

In this following, some more concrete search heuristics from proof planning that can be beneficially employed for learning, are discussed. In particular, rippling [14, 8] and the introduction of a case split.

Rippling is a search heuristic for systematic difference reduction. In the proof planner CL^AM it is performed based on an annotated logic calculus that handles

annotated terms and uses annotated matching.[4] More specifically, a skeleton-annotation indicates the commonalities between the induction hypothesis and the induction conclusion (the 'skeleton') and a context-annotation indicates the difference between the induction hypothesis and the induction conclusion (the 'context'). Rippling is essentially a difference-reducing rewriting technique that preserves the skeleton, i.e., the commonalities. Such a difference reduction works in particular for equational proofs and proofs by mathematical induction. An attempt to teach rippling to students has been made by Helen Lowe [19].

Another typical mathematical heuristic that is captured in control rules in the proof planner ΩMEGA suggests the introduction of a case split. It analyzes the overall proof attempt rather than a local search heuristic. It heuristically suggests a repair the proof attempt by introducing a case split. For the introduction of a case split, lets have a look at the proof of the following theorem. [5]

Theorem 1. *A convergent sequence of real numbers is bounded.*

We zoom into the proof process, where – under the assumption $n > k_1$ – the inequality $|x_n - lim| < e_1$ has been employed to prove the boundedness, i.e., $|x_n| \leq B$ for $2 \cdot |lim| \leq B$.
In this situation, $n > k_1$ has still to be proved or, if this is difficult or impossible as here, a *heuristic* says that a case split $(n > k_1 \vee \neg n > k_1)$ has to be introduced and the subproof obtained so far under the condition $n > k_1$ provides the proof branch for the first case $n > k_1$. A student can learn this heuristic, if she is informed by the proof planner.

The second branch resulting from the case split requires to prove $|x_n| \leq B$ under the assumption $\neg n > k_1$. Since k_1 is a natural number, the new assumption can be split into the cases $n = 1 \ldots n = k_1$. The proof can then be completed with further restricting $|x_1| \leq B, \ldots |x_{k1}| \leq B$ in addition to $2 \cdot |lim| \leq B$, which finally yields $B = max(|x_1|, \ldots |x_{k1}|, 2 \cdot |lim|)$.

4.3 Support for Constructing Mathematical Objects

Many mathematical proofs require the construction of mathematical objects. For instance, in order to prove that a function $f(x)$ converges to l, if x converges to a, for each arbitrarily small positive real number ϵ a positive real number δ has to be constructed that meets certain requirements; or in order to show that there are infinitely many prime numbers, the proof by contradiction assumes that there exist only finitely many prime numbers whose biggest one is p, and for the contradiction a prime number is constructed which is bigger than p.

Typically, such a construction is a difficult part of the proof. Students may lack a proper technique or support to determine an object satisfying all the – maybe many and maybe somewhat hidden – requirements [17]. Traditional teaching does not provide support for this task. It just delivers correct instances

[4] Rippling can also be represented by control rules as shown in an unpublished BlueNote [25].
[5] Theorem 3.2.2 in [3]

out-of-the-blue rather than providing techniques for systematically constructing the objects. For a real understanding of a proof and meaningful learning it is essential to watch and understand which constraints for an object occur in the proof process and are *collected in a store*, and then to understand the *search* for an object that satisfies all the collected constraints.

ATPs use unification as a technique for constructing objects satisfying equality requirements, that is, domain-independent constraints. In addition, proof planning can integrate constraint solving for constructing objects with certain domain-specific constraints. By readably displaying the constraint store, the proof planner informs the student of the restrictions of an object and can support the creative act of constructing an object, as shown with the systematic restriction of B in the example of §4.2.

The following example[6] contains information about proof steps, meta-reasoning, and the constraint store (single entries of the constraint store are indicated by a box surrounding them).

Note that a computational presentation of the example would be more convincing since the process character of theorem proving can be presented and illustrated by introducing parts of the non-linear proof at different ends and hierarchical levels of the proof and can be accompanied by explanations of the methods, alternative search branches, and information about the constraint store. A computational presentation could offer useful features by a hierarchical and hypertext presentation, by a separate window that displays the constraint store, etc. The presented sequence of steps is not necessarily the sequence of their final appearance in a top-down proof. Only in the end, this can yield a final proof presentation (hiding some information again, if desired). The paper form of this article, however, prevents such a presentation of the proof process.

4.4 Example

Theorem 2. *If $f : I \mapsto \mathbf{R}$ has a derivative at $c \in I$, then f is continuous at c.*

1. By expanding the definitions, the proof assumption can be re-stated (forward application) as
   $\forall\epsilon_1(\epsilon_1 > 0 \rightarrow \exists\delta_1(\delta_1 > 0 \wedge \forall x_1(|x_1 - c| < \delta_1 \rightarrow x_1 \neq c \rightarrow |\frac{f(x_1)-f(c)}{x_1-c} - f'(c)| < \epsilon_1)))$,
2. the goal can be re-stated (backward application) as
   $\forall\epsilon(\epsilon > 0 \rightarrow \exists\delta(\delta > 0 \wedge \forall x(|x - c| < \delta \rightarrow |f(x) - f(c)| < \epsilon)))$.
3. The goal is reduced to $|f(x) - f(c)| < \epsilon$ yielding the assumptions $|x - c| < \delta$ and $\boxed{\epsilon > 0}$.
4. The subformula $|\frac{f(x_1)-f(c)}{x_1-c} - f'(c)| < \epsilon_1$ is extracted from the assumption which leaves $|x_1 - c| < \delta_1$ and the 'not so important' condition $x_1 \neq c$ as subgoals.
5. The subgoal $|x_1 - c| < \delta_1$ is proved by the assumption $|x - c| < \delta$ and this sends $\boxed{x = x_1}$ and $\boxed{\delta \leq \delta_1}$ to the constraint store.

[6] Theorem 6.1.2 in [3]

6. `ComplexEstimate` reduces the goal $|f(x)-f(c)| < \epsilon$ to the following subgoals for which the existence of a positive real number M is assumed
   (a) $|x - c| \leq M$
   (b) $|\frac{f(x)-f(c)}{x-c} - f'(c)| < \frac{\epsilon}{2 \cdot M}$
   (c) $|(x - c) \cdot f'(c)| < \frac{\epsilon}{2}$.
   The first subgoal can be proved by assuming $\boxed{\delta \leq M}$, the second can be proved by assuming $\boxed{\epsilon_1 \leq \frac{\epsilon}{2 \cdot M}}$, the third can be proved by a case split on $f'(c) = 0 \vee f'(c) \neq 0$ and assuming $\boxed{\delta \leq \frac{\epsilon}{2 \cdot |f'(c)|}}$ in the second case.
7. So far, the condition $x \neq c$ is not proved. Now it can be discovered by the user or by the proof planner that $x \neq c$ does not hold generally, hence cannot be proved. As in the example of §4.2, the meta-reasoning of the proof planner can suggest to introduce a `Case-Split` $(x \neq c \vee x = c)$ into the proof attempt and to take the subproof obtained so far – which needed $x \neq c$ – as the proof branch for the first case.
8. The second branch assuming $x = c$ has to be tackled separately. Its subproof is simple because in this case $0 = |f(x) - f(c)| < \epsilon$ holds.

5 Structured Presentation

For learning of mathematical proofs, Leron and others [17, 9] have shown that the traditional sequential presentation of proofs which has been used in most textbooks and courses is inappropriate for a real understanding and for learning proof skills. The more appropriate, structured (and multi-modal) presentation with more relevant information yields longer proofs because it makes explicit the complexity inherent in the proof anyway. For learning, however, it is useful to explicate the proof ideas, proof methods, and other information as empirical research and experience has shown.

In [24] we have shown how to construct a hierarchically structured proof presentation from a schematic verbalization of proof planning methods. An additional information dimension is delivered by the contraint solving in proof planning:

The construction of objects belongs to the information that is relevant and that can be made explicit in a hierarchical proof plan presentation. The constructions of objects is a frequent and often a tricky part of proofs. There are two ways of presenting such proofs: the 'classical' that simply defines the solution's object and shows that this object satisfies the constraints (typically introduced by "let.." or similar phrases) and a presentation that uses the collected constraints to search for an object (verbally this can be introduced by a phrase such as "suppose, we had already found a solution-object, what would it look like?"). The second presentation may not be as concise and 'elegant' as the first one, but it contains additional information that is important for meaningful learning that enables students to prove theorems on their own later on. In a multi-modal presentation, the display of the constraint store that contains the constraints on solution-objects can provide the relevant information.

6 Future Work: Polya and Friends

Although the user can already browse a partial proof plan during her proof attempt, the heuristic reasoning that can guide the proof attempt is not yet presented, even though it is essential for learning. This information will be communicated to the learner in a user-adaptive way.

In addition, we believe that a mixed-initiative system [7] is more appropriate what an average student can handle properly while concentrating on the mathematics rather than on handling the system. The following lists some of the features that will be targeted for a re-design of LOUI for a truly interactive and mixed-initiative use of proof planning.

- The input should be as similar to mathematical language as possible.
- The user has to be able to easily and in an intuitive way choose, backtrack, and tentatively try the application of a strategy (including backtracking etc.), a method, and of parameters etc.
- The user should easily be able to investigate the success or failure of a method and get feedback about repair opportunities.
- Different levels of detail of a proof plan should be inspectable.
- In the presentation of proof attempts, parts such as formulas, terms, and methods have to carry *semantics*. [8]
- A proof verbalization needs to show the structure of the proof and the goal structure
- To support a user, highlighting of proof steps in focus (or hiding those not in focus) is needed in order to avoid cognitive overload by too high a complexity.

Moreover, the language for communicating with the user should approximate the commonly used mathematical (not so much logical) language. This way, the additional effort that is necessary to learn the language is not in vain but produces one of the skills belonging to the repertoire of mathematical work anyway. Examples of systems that use such languages are Mizar, ISETL [10], and a preliminary attempt by Schmidt [30].

In addition to the discussed changes of the current proof planner and its GUI, we shall augment the actual proof planning with radically new facilities described next.

6.1 Polya: Support of Meta-Cognition

The actual theorem proving process is merely one part of the problem solving process in which exploration and producing conjectures plays a crucial role [7, 33]. This does not only apply for learning but also for the mathematician's work as, e.g., described by Buchberger's creativity spiral and in [6] that identifies the following activities in mathematical theorem proving

[7] See [12, 35] for an introduction to mixed-initiative (planning) systems.

[8] Carry semantics in the sense, that behind the presentation the semantics is represented and can be used for manipulations such as drag'n drop, for making it an assumption, etc.

- production of a conjecture (exploration of the problem situation)
- formulation of the statement according to conventions
- identification of appropriate arguments and link to the existing theory, exploration of (limits of) validity of the conjecture
- discovery or selection of proof methods and mathematical objects
- organization of the arguments into a coherent proof that is acceptable to current mathematical standards (for the communication of the proof).

Polya suggests a student to proceed with problem solving in a larger context that involves the phases of:

1. understanding the problem,
2. devising a plan for a solution,
3. executing (and debugging) the plan,
4. analyzing the success or failure of the plan.

In an explorative environment an additional start phase would comprise the invention of an hypothesis. For guidance, Polya suggests to ask the following questions

- what is known and what is unknown?
- do you remember a similar problem?
- find the connection between the known and the unknown
- carry out the plan and check each step
- is there an alternative way to solve the problem?
- examine the solution, can you use the result or the method for another problem?

Polya's heuristics for problem solving caught attention in the early years of AI and their implementation was targeted. In 1981, Allen Newell summarized why these attempts were not successful. He explained that Polya's heuristics are too general and each represents a whole class of heuristics [26]. Consequently, for an automated problem solving system it may be too difficult even today to realize such a heuristic guidance.

However, Polya's heuristics for meta-reasoning can be used for supporting interactive learning. Such a support for the student's meta-cognition is desirable because cognitive research [32] has indicated that a large part of what comprises competent problem solving behavior consists of the ability to guide, monitor, reflect, and assess the own problem solving process. It also indicates that students are poor at this, partly because these abilities and the introspection into the reasoning are rarely trained or discussed in traditional teaching.

A *meta-level cycle* as depicted in Figure 1 can help students who do not have those meta-cognitive skills yet. We plan to implement such a supportive cycle in a POLYA module. This module for supporting meta-cognitive activities will in some sense be similar to ThinkerTools [37], a tool for inquiry learning in experimental sciences. ThinkerTools provides the cycle: question, hypothesize, investigate, analyze, model, and evaluate. Of course, we expect quite a few

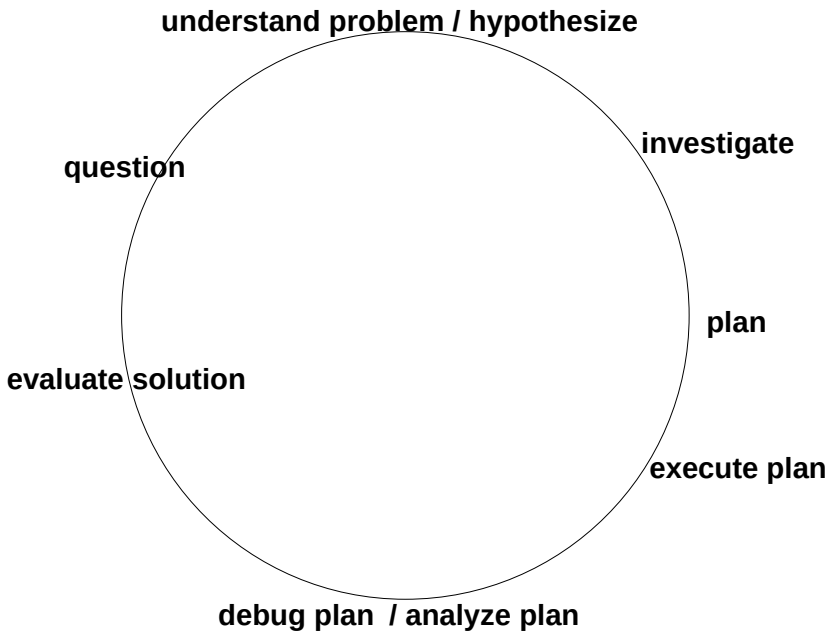

Fig. 1. The meta-level cycle

changes that are mostly relevant for the mathematical problem solving as opposed to physics or biology experiments. Moreover, our meta-cognitive support will, be user-adaptive in the sense, that the student will be supported depending on her own abilities and needs.

6.2 Proof Planning by Programming

A use of proof planning as a cognitive tool that is geared towards constructive learning could be based on 'programming' of proof planning methods and proofs by students. For experience in this direction we refer to the work of the mathematicians Dubinsky and Leron who replace pure lecture by constructive, interactive methods involving programming and cooperative learning. They report an radically increased amount of meaningful learning for average students [18].

Different from other mathematical textbooks, their algebra course [10] is based on the constructivist belief that, before students can make sense of any presentation of abstract mathematics, they need to engage in mental activities which construct the base for future verbal explanations. Notions such as Co-sets and quotient groups become more meaningful to the students when definitions, examples, proofs etc. are closely related to activities than just presented in a lecture. Their course requires to actively *do* things (mainly programming) and to

discuss them with the fellow students. It asks students to formulate a conjecture, test it, and try to give an explanation. This includes an exploration with a computer until the understanding of a topic is satisfying, as opposed to the traditionally practiced attitude to avoid mistakes and immediately correct faulty solutions.

In order to achieve this goal with proof planning, a relatively simple programming language for methods has to be designed that is close to mathematical language that students have to learn anyway.

7 Conclusion

We recognized proof planning as a methodology that is useful for a successful learning of mathematical proof. Therefore, we want to use a proof planner as a user-adaptive cognitive tool for learning mathematical problem solving, in particular mathematical theorem proving. The first step towards this goal has been the integration of the currently proof planner of the ΩMEGA system into the web-based learning environment ACTIVEMATH `http://www.activemath.org` for working through example proofs and for interactive proof exercises. In addition to this proof tool, the learner can benefit from other facilities of ACTIVEMATH such as an ontological overview of a mathematical area (concept map), a user-adaptive choice of examples and exercises, suggestion mechanisms, etc.

We described which support already exists and how the proof planner and its GUI will be modified and enhanced in order to ease its use and to support all phases of mathematical problem solving/proving as they occur in learning and in mathematicians' work. This 'future work' description deals with some of the essential objectives of the Mippa project and invites other researchers to collaborate for these objectives.

References

1. Gutachten zur Vorbereitung des Programms "Steigerung der Effizienz des mathematischen-naturwissenschaftlichen Unterrichts. Bund-Länder-Kommission, 1997. Materialien zur Bildungsplanung und zur Forschungsförderung.
2. V. Aleven, K.R. Koedinger, and K. Cross. Tutoring answer explanation fosters learning with understanding. In S.P. Lajoie and M. Vivet, editors, *Artificial Intelligence in Education*, pages 199–206. IOS Press, 1999.
3. R.G. Bartle and D.R. Sherbert. *Introduction to Real Analysis.* John Wiley& Sons, New York, 1982.
4. J. Baumert, R. Lehmann, M. Lehrke, B. Schmitz, M. Clausen, I. Hosenfeld, O. Köller, and J. Neubrand. *Mathematisch-naturwissenschaftlicher Unterricht im internationalen Vergleich.* Leske und Budrich, 1997.
5. W.W. Bledsoe, R.S. Boyer, and W.H. Henneman. Computer proofs of limit theorems. *Artificial Intelligence*, 3(1):27–60, 1972.
6. P. Boero. Argumentation and mathematical proof: A complex, productive, unavoidable relationship in mathematics and mathematics education. Proof Newsletter, 1999. .

7. P. Boero, R. Garutti, and M.A. Mariotti. Some dynamic mental processes underlying producing and proving conjectures. In *Proceedings of PME-XX*, volume 2, pages 121–128, 1996.
8. A. Bundy, F. van Harmelen, A. Ireland, and A. Smaill. Extensions to the rippling-out tactic for guiding inductive proofs. In M.E. Stickel, editor, *Proceedings of the 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*. Springer, 1990.
9. R. Catrambone and K.J. Holyoak. Learning subgoals and methods for solving probability problems. *Memory and Cognition*, 18(6):593–603, 1990.
10. E. Dubinsky and U. Leron. *Learning Abstract Algebra with ISETL*. Springer-Verlag, 1993.
11. M.T.H. Chi et al. Self-explanation: How students study and use examples in learning to solve problems. *Cognitive Science*, 15:145–182, 1989.
12. G. Ferguson, J. Allen, and B. Miller. Trains-95: Towards a mixed-initiative planning assistant. In B. Drabble, editor, *Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77, 1996.
13. G. Holland. *Geolog-Win*. Dümmler, 1996.
14. D. Hutter. Guiding inductive proofs. In M.E. Stickel, editor, *Proceedings of 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Artificial Intelligence*. Springer, 1990.
15. W:R: Joolingen and T. Jong. Design and implementation of simulation-based discovery environments: the SMISLE solution. *Journal of ARtificial Intelligence and Education*, 7:253–277, 1996.
16. S. Lajoie and S. Derry, editors. *Computers as Cognitive Tools*. Erlbaum, Hillsdale, NJ, 1993.
17. U. Leron. Heuristic presentations: the role of structuring. *For the Learning of Mathematics*, 5(3):7–13, 1985.
18. U. Leron and E. Dubinsky. An abstract algebra story. *American Mathematical Monthly*, 102(3):227–242, March 1995.
19. H. Lowe, A. Bundy, and D. McLean. The use of proof planning for co-operative theorem proving. Research Paper 745, Department of AI, 1995.
20. H. Lowe and D. Duncan. Xbarnacle: Making theorem provers more accessible. In W. McCune, editor, *Proceedings of the Fourteenth Conference on Automated Deduction (CADE-14)*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 404–408. Springer, 1997.
21. E. Melis. AI-techniques in proof planning. In *European Conference on Artificial Intelligence*, pages 494–498, Brighton, 1998. Kluwer.
22. E. Melis. The "limit" domain. In R. Simmons, M. Veloso, and S. Smith, editors, *Proceedings of the Fourth International Conference on Artificial Intelligence in Planning Systems*, pages 199–206, 1998.
23. E. Melis, Ch. Glasmacher, C. Ullrich, and P. Gerjets. Automated proof planning for instructional design. In *Annual Conference of the Cognitive Science Society*, pages 633–638, 2001.
24. E. Melis and U. Leron. A proof presentation suitable for teaching proofs. In S.P. Lajoie and M. Vivet, editors, *9th International Conference on Artificial Intelligence in Education*, pages 483–490, Le Mans, 1999. IOS Press.
25. Erica Melis and Julian Richardson. Separation of control and logic in rippling and unwraphyp, 1998.
26. 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.

27. J. Piaget. *Equilibration of Cognitive Structures.* Viking, New York, 1977.
28. G. Polya. *How to Solve it.* Princeton University Press, Princeton, 1945.
29. J. Richter-Gebert and U.H. Kortenkamp. *The Interacitive Geometry Software Cinderella.* Springer-Verlag, 1999.
30. P. Schmidt. Preparing oral examinations of mathematical domains with the help of a knowledge-based dialogue system. In *Proceedings of Ed-Media*, 2001.
31. A.H. Schoenfeld. *Mathematical Problem Solving.* Academic Press, New York, 1985.
32. A.H. Schoenfeld. *Learning to Think Mathematically: Problem Solving, Metacognition, and Sense Making in Mathematics*, chapter 15. McMillan Publ.Company, New York, 1992.
33. M. Simon. Beyond inductive and deductive reasoning: The search for a sense of knowing. *Educational Studies in Mathematics*, 30:197–210, 1996.
34. D. Suthers, A. Weiner, J. Connely, and M. Paolucci. Belvedere: Engaging students in critical discussion of science and public policy issues. In *7th World Conference on Artificial Intelligence in Education, AIED-95*, pages 266–273, 1995.
35. Manuela M. Veloso. Towards mixed-initiative rationale-supported planning. In A. Tate, editor, *Advanced Planning Technology*, pages 277–282. AAAI Press, May 1996.
36. L. Vygotsky. *Thought and Language.* MIT press, Cambridge, MA, 1986. originally published 1962.
37. B.Y. White and J.R. Frederiksen. *Inquiry, Modeling, and Metacognition: Making Science Accessible to all Students.* Lawrence Erlbaum, 1998.