e-Learning Logic and Mathematics: What We Have and What We Still Need

Erica Melis and Jörg Siekmann

1 Introduction

Intelligent tutoring systems provide a promising application area for techniques from many subfields of Artificial Intelligence (AI), including knowledge representation, user modelling, rule-based systems, automated diagnosis, automated reasoning, adaptive hypermedia, natural language processing and automated reasoning. But as it turned out this is not just a one way road, but a give and take in both ways as these educational applications led to new research problems as well, among others, open student modelling, tutorial dialogues, and adaptive hypermedia, see, e.g., [Aleven and Koedinger, 2000; Brusilovski, 1996]. This also applies to automated reasoning as we shall see in the sequel.

With the widespread availability of the Web, there is the great opportunity that educational tools developed at one place can be used anywhere as long as they are encapsulated in Web-applications that are interoperable and compliant with standard input languages.

While e-learning tools are now widely used in life-long learning applications such as industrial training courses for specialist knowhow and skills, these systems slowly but surely enter into school teaching [Koedinger et al, 1997; Matsuda and vanLehn, 2005], academic teaching, and professional training as well. Some countries even embrace these new opportunities on a grand scale. China has currently about 90 million Internet users with a yearly growth rate of more than 10%. Provided this trend is not disturbed by external events, China will have more Internet users than the US by 2007 and it has been predicted that within the next ten years, China will have more officially registered Internet users than the rest of the world [China, 2005]. This is the fastest growing market in the world right now and the Chinese government intends to use this as a backbone of its next five year development plan on a grand scale — in particular to develop and educate

the western parts of the country.

In this article, we shall focus on e-learning for mathematics (and logic) with special emphasis on its relationship to automated theorem proving and deduction. As first publications show (e.g. [Melis, 2000]) the field of automated theorem proving begins to recognise the potential of educational systems and some research groups of the deduction community have started to work on deductive components inside of educational systems [Melis *et al.*, 2001b; Baumgartner and Furbach, 2003]. Others developed theorem proving systems as interactive tools for education. Historically first was Patrick Suppes' use of a deduction system at Stanford [Suppes, 1981] and Peter Andrews' TPS system to teach higher order logic at Carnegie Mellon University [TPS, 2004]. Other recent work includes [Lowe and Duncan, 1997; Buchberger *et al.*, 1997; Bornat, ; Sieg and Byrnes, 1998; Sommer *et al.*, 2000; Melis *et al.*, 2001b]. Most of these systems, however are based primarily on their developers' experience and insight rather than on objective cognitive psychological or pedagogical results and — even more importantly — have generally not been empirically validated for their educational impact and cognitive adequacy.

As experienced in other areas before, it is not the development of the technology per se that has an impact on education but only a technology responding to the actual needs of the learner. In order to build *useful and usable* educational applications, empirical investigation and observations have to set the stage for technologies effectively supporting the learner. As we will see in Section 2, there are a number of cognitive and pedagogical results that we can rely upon and we should pay attention to.

Most of this article focuses on mathematics (including logic) education.[1] It distinguishes the usage of theorem proving/deduction within modules the student is not using directly and the usage of systems as interactive problem solving tools.

The paper is organised as follows. In chapter 2 we shall give a brief account of some relevant (empirical) psychological and pedagogical results. Then we analyse in chapter 3 how current techniques can or cannot realise these needs and provide some examples. Finally, we pose some challenges and propose future work in chapter 4.

2 What e-Learning Systems Need

Learning resembles research and discovery in many ways [White and Shimoda, 1999]. However, a student is not like an expert mathematician or a developer of a theorem proving system. A system that is useful for learning

[1] However, most needs and some techniques apply more generally to e-learning systems per se.

drastically differs from systems that are valuable for a mathematician or other expert users.

This statement may sound like a platitude, but it points to a potentially rich and new research programme. A student's goals in learning (mathematics) may be just to understand existing problem solutions and proofs or else to learn how to find a solution or to prove a theorem on her own (with or without system support). The goal might be to train solution procedures or to learn how to gather information and search for solutions in the literature.

Cognitive psychology sheds light on human learning: current paradigms and instructional theories assume learning requires that a student *constructs* knowledge, dependencies and procedural skills in her mind [Piaget, 1977; Vygotsky, 1978]. As many empirical results show (see e.g. [Mandl *et al.*, 1997]), this should be promoted inter alia by:

- contextual real life learning experience
- a personalised learning context
- active, explorative learning opportunities
- feedback on the learner's activities
- coaching and stimulating meta-reasoning
- an appropriate level of abstraction and detailed presentation of the solution
- an appropriate user interface.

Let us elaborate these aspects in turn:

Context. If a learner can match her learning experience with a real-life context it will be more likely that she learns more rapidly and that she can transfer this knowledge better to problem solving in reality [Andriessen and Sandberg, 1999]. A 'real-life context' could be introduced by realistic visual impressions from video clips, pictures or diagrams or simply by a verbal description of a concrete situation representing a problem from the experience of the targeted learner group.

Another relevant feature concerns the complexity of real world problems and the various phases their solution requires. Such a problem solving cycle may require very different activities and skills from a pure textbook or academic problem: recognising the problem in the real life setting, developing the mathematical model, then mathematical problem solving and possibly revision and, finally, translating the solution back from the mathematical result into the real life setting.

Personalisation. Personalised learning experience is crucial not just for the motivation but also for efficiency and effectiveness of learning [Andriessen and Sandberg, 1999]. There are several ways of adapting an e-learning tool to the student. The most obvious one is to tailor the learning material (in particular the examples and exercises) to the capabilities of the student and to her learning goals. Furthermore, a course on statistics for example should be tailored to the special interest of the student: a student of physics expects different statistics examples than, say, a chemist or an electrical engineering student.

Active Learning. One of the most important ingredients of effective learning is *active problem solving* and exploration of alternatives and the discovery of faulty steps and assumptions [VanLehn *et al.*, 2001]. In particular learning from errors and failures is an important ingredient of active learning and should be maximally exploited.

Feedback. Empirical investigations corroborate that an appropriate feedback during problem solving improves subsequent performance [Jacobs, 2001]. An intelligent learning assistant should therefore diagnose mistakes, adaptively scaffold the individual process of problem solving and generate appropriate feedback and hints, see e.g., [Narciss, 2001; Tsovaltzi, 2005].

Meta-reasoning. Meta-reasoning plays a crucial role in successful problem solving [Polya, 1945; Schoenfeld, 1985; Melis and Ullrich, 2003] which includes planning, monitoring, self-regulation and self-explanation [Chi *et al.*, 1989]. Take, for example, Polya's 'How to Solve It' [1945]: it has the form of a 'how-to manual', i.e. a formulation of a set of heuristics cast in the form of brief commands within a frame of the four following problem solving stages

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

These stages should be followed and actively monitored by the student, and exactly how this can be done is the subject of this very influential book. So far, however, this kind of reasoning is rarely taught, let alone implemented in current e-learning systems.

Cognitively Adequate Presentation. The presentation of a solution and in particular, the presentation of proofs have to be appropriately structured [Catrambone and Holyoak, 1990] and to hide irrelevant details in order to be comprehensive and transferable.The presentation should exploit

multiple modalities such as text, formulas, diagrams, but also speech and animations if appropriate.

User Interface. The user interface of an e-learning tool has to be designed such that the student can focus on the essentials of the particular learning task, undisturbed much as possible by system peculiarities. Anderson [Anderson and Pelletier, 1991] claims that the design of the user interface substantially influences what a student will learn using the LISP tutor [Anderson *et al.*, 1995] as a case in point. Depending on the user interface, in particular, depending on the input language, a learner will either just learn a particular programming syntax or else be able to focus on general programming strategies. Moreover, just clicking buttons is unlikely to stimulate serious learning. This applies to learning mathematics just as well. Since e-learning tools have to be usable rather than just useful, a user-centred design of the interface built according to current cognitive requirements is key.

3 What we Have: ActiveMath

Learning environments have to meet realistic and complex needs, both technically as well as psychologically. So let us look at the the pedagogical and technical goals of our research for the ActiveMath system.

Pedagogical Goals

ActiveMath aims at an interactive and exploratory learning process and assumes the student to be responsible for the actual learning session. Therefore, the system supports relative freedom for navigating through a course and the user defines choices. The system supports a dynamic student model and by default, the student model is scrutable, i.e., inspectable and modifiable. Moreover, dependencies between the mathematical concepts can be inspected in a dictionary in order to help the student to learn the overall structure of a domain (e.g., analysis, algebra or number theory).

ActiveMath can adapt a course to the learner's goals, prerequisites and learning scenarios. In colleges and universities, the same subject is taught differently for different groups of users in different contexts, e.g., statistics has to be taught differently for students of mathematics, for students of economics, or in medicine. Therefore, the adaptive choice of content to be presented as well as examples and exercises is pivotal. In addition, an adaptation of examples and exercises to the student's *actual* capabilities is highly desirable in order to keep the learner in the zone of proximal development [Vygotsky, 1978] rather than overtax or undertax her.

Moreover, web-based systems can be used in several learning contexts, e.g., long-distance learning, homework, or teacher-assisted learning. Person-

alization is required in all of them because even for teacher-assisted learning in a classroom with, say, 30 students and one teacher, the teacher cannot really respond to all the individual needs. ActiveMath's current version provides adaptive content and adaptive presentation features.

Technical Goals

Building hyper-media content with assured quality is a time-consuming and costly process, hence the content should be *reusable* in different contexts. As most of today's interactive textbooks consist of a collection of predefined documents, typically canned `HTML` pages and multimedia animations, it is difficult to reuse them in another context. A re-combination of the encoded learning objects or a new adaptation of the course presentation and content of the course to other users is impossible most of the time.
ActiveMath's generic and semantically annotated knowledge representation supports re-usability and interoperability. In particular, it is compliant with the emerging mathematical knowledge representation and communication standards such as Dublin Core, `OpenMath`, `MathML`, and LOM.[2] Some of the buzzwords here are metadata, ontological XML, and standardized content packaging. Such features of the knowledge representation ensure a longer life cycle even with new and changing technologies in browsers and other devices.

In order to use the potential power of existing web-based technology e-learning systems need an open architecture to integrate and connect to new components including student management systems such as Ilias, Moodle, Sakay, WebCT as well as assessment tools, collaboration tools, and problem solving tools.

3.1 Architecture

The architecture of ActiveMath, as sketched in Figure 1, strictly realizes the principle of separation of (declarative) knowledge from functionalities as well as the separation of different kinds of knowledge. For instance, pedagogical knowledge is stored in a pedagogical rule base, the educational content is stored in MBase, and the knowledge about the user is stored in the student model. This principle has proved valuable in many AI-applications and eases modifications as well as configurability and reuse of the system.

ActiveMath has a client-server architecture whose client is a browser (or several browsers in case of multi-user mode). This architecture serves not only its openness but also its *platform independence*, and a browser such as Netscape, Mozilla, or IE with MathPlayer is sufficient to work with ActiveMath. The components of ActiveMath have been designed in a

[2] `http://ltsc.ieee.org/wg12/`

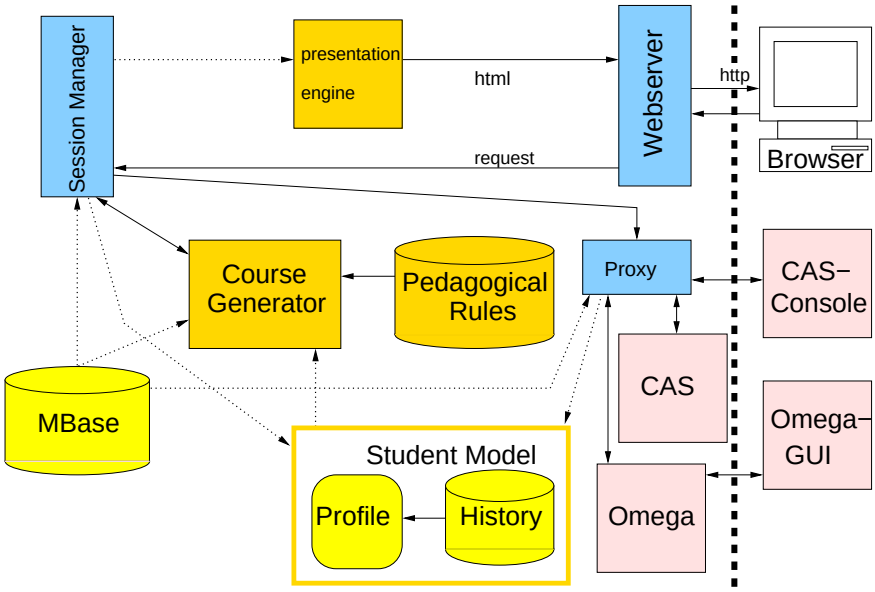

Figure 1. Architecture of ActiveMath

modular way in order to guarantee exchangeability and robustness.

The actual flow of control (and data) in the above diagram is as follows: when the student has chosen her goal concepts and learning scenario, the session manager sends a request to the *course generator*. The course generator is responsible for choosing and arranging the content to be learned. The course generator contacts the *mathematical knowledge base* in order to fetch the identifiers (IDs) of the mathematical concepts that are required for learning the goal concepts, queries the student model in order to find out about the student's prior knowledge and preferences, and uses *pedagogical rules* to select, annotate, and arrange the content — including examples and exercises — in a way that is suitable for this particular learner in this particular session. The resulting instructional graph of IDs is sent to the *presentation engine* which retrieves the actual mathematical content corresponding to the IDs and transforms the `XML`-data to output-pages which are then finally presented via the student's browser.

The *course* generator and the suggestion mechanism [Melis and Andres, 2004] previously worked with the rule-based system Jess [Friedman-Hill, 1997] that evaluates the (pedagogical) rules in order to decide which particular adaptation and content to select and which actions to suggest. Jess

uses the Rete algorithm [Forgy, 1982] for optimization.

External systems such as the computer algebra systems Maple [Maple, 1986] and MuPad [Sorgatz and Hillebrand, 1995] and the proof planner MULTI [Melis and Meier, 2000] communicate with the ACTIVEMATH system. They serve as cognitive tools [Lajoie and Derry, 1993] and support the learner in complex interactive exercises. They also assist in generating feedback by evaluating the learner's input. Finally, a diagnosis is passed to the student model in order to update the student model.

In exercises ACTIVEMATH does not necessarily guide the user strictly along a predefined expert solution. It only evaluates whether the student's input is mathematically equivalent to an admissible subgoal, i.e., maybe it is irrelevant, but not outside the solution space (see [Buedenbender *et al.*, 2002]). Moreover, the external systems can support the user with automated problem solving, i.e., they may take over some parts in the human problem solving process and thereby help the user to focus on important learning tasks and to delegate routine tasks.

Actually, the diagnoses of a student's performance is well known to be an 'AI-complete' problem and hence several context-dependent equivalence checkings have been implemented so far. Moreover, most tutor systems encode the possible problem solving steps and the most typical misconceptions into their solution space or into systems that execute them. From this encoding, the system diagnoses the misconception of a student. This is, however, infeasible in realistic applications with large solution spaces as it is in general impossible to represent all potential misconceptions of a student [VanLehn *et al.*, 2002].

The *presentation engine* generates personalized web pages based on two frameworks: Maverick and Velocity. Maverick[3] is a minimalist model view controller (MVC) framework for web publishing using Java and J2EE, focusing solely on MVC logic. It provides a wiring between URLs, Java controller classes and view templates.

The presentation engine is a reusable component that takes a structure of `OMDoc`s and transforms them into a presentation output that can be PDF (print format) or `HTML` with different maths-presentations such as Unicode or `MathML` (screen format) [Ullrich *et al.*, 2004]. Basically, the presentation pipeline comprises two stages: stage 1 encompasses Fetching, Pre-Processing and Transformation, while stage 2 consists of Assembly, Personalization and optional Compilation. Stage 1 deals with individual content fragments or items, which are written in `OMDoc` and stored in a knowledge base. Content items in the knowledge base do not depend on the user who is to view them, they have unique identifiers and can be handled separately. It is only

[3]Maverick: `http://mav.sourceforge.net/`

in stage 2 that items are composed to user-specific pages.

3.2 Adaptivity

ActiveMath adapts the course generation and presentation to the student's

- technical equipment (*customization*)
- environment variables, e.g., curriculum, native language, and the field of study (*contextualization*) and
- her cognitive and educational needs and preferences such as learning goals, and prerequisite knowledge (*personalization*).

As for personalization, individual preferences (such as the style of presentation), goal-competencies, and mastery-level are taken into account by the course generator. The goal-competencies are characterized by concepts that are to be learned and by the competency-level to be achieved: knowledge (k), comprehension (c), or application (a).

The learner can initialize her student model by self-assessment of her mastery-level of concepts and choose her learning goals and learning scenario, for instance, the preparation for an exam or learning from scratch for k-competency level. The course generator processes this information and updates the student model and generates pages/sessions as depicted in the sreenshots of Figures 2 and 3. These two screenshots differ in the underlying scenarios as the captions indicate.

The adaptation to the capabilities of the learner is carried out by the course generator and later, during the actual session, by the suggestion mechanism. The course generator checks whether the mastery-level of prerequisite concepts is sufficient for the goal competency. If not, it presents the missing concepts and/or explanations, examples and exercises for these concepts to the learner when a new session is requested. The suggestion mechanism acts dynamically in response to the student's activities. Essentially, this mechanism works with two blackboards, a diagnosis blackboard and a suggestion blackboard on which particular knowledge sources operate.

We also investigated special scenarios that support a student's metacognitive activities, such as those proposed in Polya's book [1945]. A Polya-scenario structures the solution space with headlines such as "understand the problem", "make a plan", "execute the plan", and "look back at the solution". It augments and structures exercises with additional prompts similar to the above headlines [Melis and Ullrich, 2003].

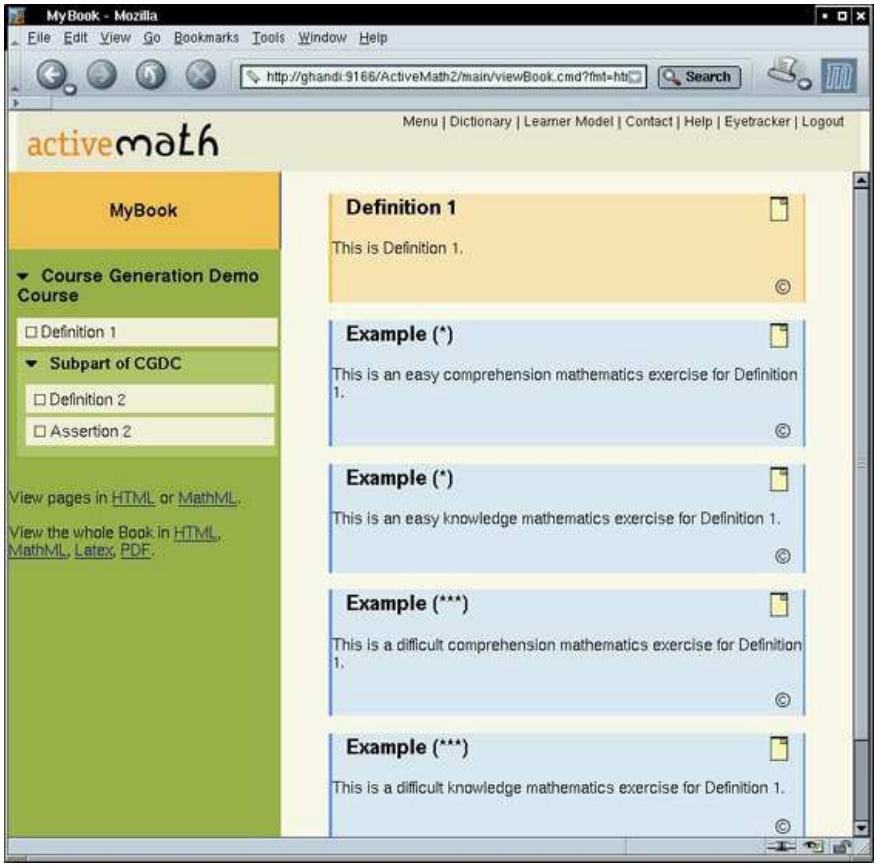

Figure 2. A screen shot of an ActiveMath session for exam preparation

3.3 The Student Model

User modeling has been a research area in AI for some time. It actually started by developing techniques for student modeling and still continues with the investigation of representational issues as well as diagnostic and updating techniques.

As ActiveMath's presentation is user-adaptive, we need to incorporate persistent information about the student as well as a representation of the student's learning progress. Therefore, *static* (wrt. the current session) properties such as field, scenario, goal concepts, and preferences as well as *dynamic* properties such as the mastery values for concepts and the student's

Figure 3. k-level session of ActiveMath

actual behaviour, are stored in the current student model.

The profile is initialized with the learner's entries entered into ActiveMath's registration page which holds the preferences (static), scenario, goals (static for the current session), and self-assessment values for knowledge, comprehension, and application of concepts (dynamic).

The history component stores the information about the learner's actions. Its elements contain information such as the IDs of the content of a read page or the ID of an exercise, the reading time, and the success rate of the exercise. We also developed a "poor man's eye-tracker" which allows to trace the student's attention and reading time [Ullrich and Melis, 2002].

To represent the concept of mastery, the current (dynamic) profile con-

tains values for a subset of the competences of Bloom's mastery taxonomy [Bloom, 1956]:

- Knowledge (K)
- Comprehension (C)
- Application (A)

Finishing an exercise or going to another page triggers an update of the student model. Different types of learner actions may exhibit different competencies, hence reading a concept mainly updates 'knowledge' values, reading examples mainly updates 'comprehension', and solving exercises mainly updates 'application'. When the student model receives the notification that a student has finished reading a page, an evaluator fetches the list of its items and their types (concept, example, ...) and delivers an update of the values of those items. When the learner finishes an exercise, an evaluator delivers an update of the values of the involved concepts that depend on the difficulty and on the rating of how successful the solution was.

The student model is inspectable and modifiable by the student as shown in Figure 4. Our experience is that students like to inspect their student model in order to plan what to learn next.

3.4 Knowledge Representation

As opposed to the purely syntactic representation formats for mathematical knowledge such as LaTex or `HTML`, the knowledge representation used by ActiveMath is the *semantic* `XML`-language `OMDoc` [Kohlhase, 2000] which is an extension of `OpenMath` [Caprotti and Cohen, 1998]. `OpenMath` provides a collection of `OpenMath` objects together with a grammar for the representation of mathematical objects and sets of standardized symbols (the content-dictionaries). That is, `OpenMath` talks about objects rather than syntax.

`OpenMath` does not have the means to represent the content of a mathematical *document* nor its structure, whereas `OMDoc` defines logical units such as "definition", "theorem", and "proof" with semantical annotations. In addition, the purely mathematical `OMDoc` representation is augmented by educational metadata such as the difficulty of a learning object or the type of an exercise.

This representation has several advantages, among them

- it is human and machine understandable
- the presentation can automatically and dynamically be linked to concepts and learning objects and thus,

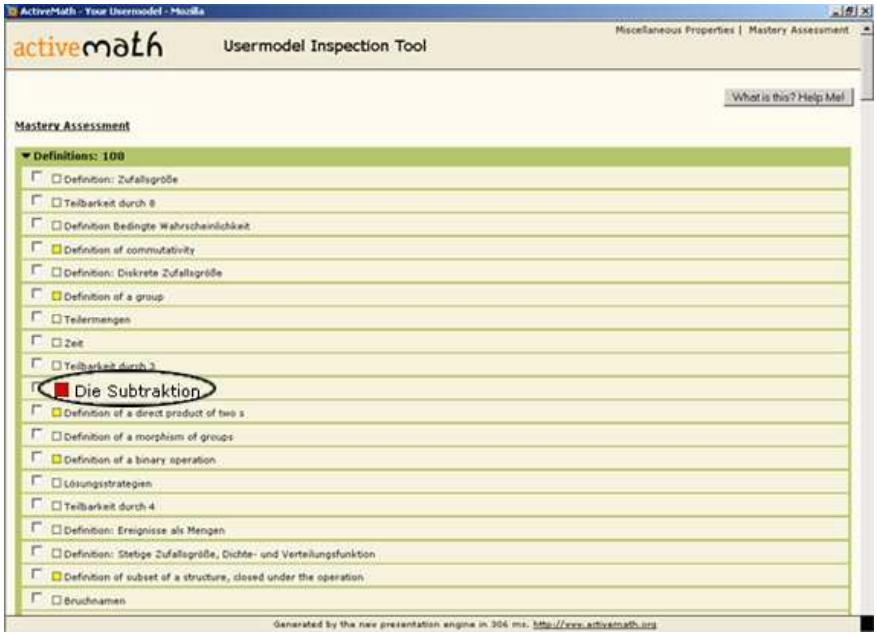

Figure 4. Inspection of the student model (mastery-level)

– concepts can easily be fetched from ActiveMath's dictionary when clicking on a concept or formula during the course
– and mathematical objects can in principle be copied and pasted.

For more details on the representation of mathematical knowledge in ActiveMath, see [Melis *et al.*, 2003].

The web-based ActiveMath system has been under development now for several years at the CCeL (Competence Centre for e-Learning) of the DFKI and at the University of Saarland. Its research and development is supported by several projects of the BMBF (the German Ministry for Research and Development) and the European Union. A demo (and demo guide) is available at `http://www.activemath.org`.

3.5 Automated Reasoning Tools for System Functionalities

Deduction systems have been used as internal modules in learning systems for user modeling [Kobsa and Pohl, 1995], diagnosis [Hoppe, 1994], and course generation [Baumgartner and Furbach, 2003; Melis, 2005]. We discuss the last usage here.

Personalization In order to present the learning material according to the user's needs and abilities, the characteristics of the individual user have to be stored and updated and personalization actions can then be inferred.

So far, ActiveMath uses relatively simple deductive techniques to infer the personalization of content and its presentation from the information in the user model and from pedagogical knowledge in a rule base [Ullrich, 2003]. The course generator of ActiveMath dynamically generates mathematical courses by

1. retrieving the appropriate content from a knowledge base and
2. by applying pedagogical knowledge that is formalized in rules.

In the second stage, when the content is already assembled from the collection of concepts the user has to learn in order to meet the learning goals, the pedagogical rules are applied to select instructional items that are related to these concepts. They also determine the order of items in the learning material. The rules have a *condition* and an *action* part. The condition part of a rule specifies the conditions that have to be fulfilled for the rule to be applicable, the action part specifies the actions to be taken when the rule is applied. The course generator uses the pedagogical rules in order to decide: (i) which information should be presented on a page; (ii) in which order this information should appear on a single page; (iii) how many exercises and examples should be presented and how difficult they should be; (iv) whether or not to include exercises and examples that make use of a particular service system. Since the work with service systems requires a certain minimal familiarity with these systems, ActiveMath presents these exercises only, if the capability to use them is confirmed in the user model. The following are examples of pedagogical rules for two different types of decisions. The rule

```
(defrule PatternForExamPrep
(scenario ExamPrep) =>
          (assert (definitions assertions methods exercises)))
```

determines the kind of items and the order in which they will appear on the course pages. In this example, the learner selected *preparation* for an exam (indicated by the fact `(scenario ExamPrep)`). When this rule fires, the facts `(definition, assertions, methods, exercises)` are asserted, i.e., added to the course. This implies that these items will appear on a page in the specified order.

In turn, these facts may cause other rules to fire, e.g., those choosing exercises with an appropriate levels of difficulty:

```
(defrule RequireAppropriateExercise
  (exercises)
  (definition (name ?definition)
              (userKnowledge ?user-knowledge))
  (test (< ?user-knowledge 0.3))
  =>(assert  (choose-exercise-for  ?definition (0.3 0.5 0.7))))
```

This rules determines that if exercises should be presented at all (indicated by `(exercises)`) and if there exists a definition d, then d's name is bound to the variable `?definition` and the learner's knowledge of d is bound to `?user-knowledge`. Then the `test` is evaluated and its value determines whether the rule fires or not. The above rule fires, if the learner's knowledge is less than 0.3. Then the fact `(choose-exercise-for ?definition (0.3 0.5 0.7))` is inserted into the assembled set of items and this triggers the selection of examples for d with difficulty levels 0.3, 0.5, and 0.7 respectively.

3.6 Automated Deduction Systems as Tools

Several systems have been developed for teaching mathematics, e.g., the interactive CMU proof tutor [Sieg and Byrnes, 1996], the EPGY theorem proving environment [Sommer *et al.*, 2000] where Otter [McCune, 1990] checks the correctness of the student's input, living Book [Baumgartner and Furbach, 2003] in which a theorem prover checks the correctness of truth values and normal forms, and Jape [Aczel *et al.*, 1999]. These systems have a deductive component, however, since the (deductive) service systems are fixed and do not adapt to the student, the students have to adapt to the system.

Adaptation to the Student. It may not always be the best idea to make only correct suggestions, as the students might then just click on these suggestions rather than learn anything. So sometimes it is better to include faulty suggestions. The decision of when it is most appropriate to present a faulty suggestion, depends on the learning goal, the learning context, the learning history, and the competency of the student, as well as on the pedagogical strategy.

The interaction console of the proof planner MULTI which is used as a mathematical service in ActiveMath, offers adapted suggestions by configurable suggestion agents [Pollet *et al.*, 2003]. A configuration is a set of agents and an agent encodes pedagogical knowledge. For instance, in order to understand why a method is applicable it may be useful for a student to encounter a situation in which a method is not. Sometimes alternative proof strategies should be learned, where these strategies represent different ways to prove a theorem. For instance, proofs of properties of residue classes in group theory can be tackled by three different proof planning strategies.

The first strategy tries to apply some related theorems, the second strategy reduces the problem to an equation for which a general solution must be found, and the third strategy introduces a case split over the (finitely many) elements of a residue class. The decision for the suggestion of a strategy depends on the knowledge of a student (whether she knows the theorems that are the prerequisites for the first strategy) but it could also be the result of her performance in previous exercises in which, e.g., the other strategies have been trained already.

Active Explorative Learning. An interesting application of proof planning in a system that teaches how to prove theorems is as a 'domain reasoner'. As there can be more ways to prove a given theorem than a tutoring system anticipates and stores, a student may come up with a proof idea which does not match any expert solution the system stored. Now, an interactive proof planner is a more (albeit not always) helpful assistant under these circumstances as it will just generate a new proof using the student's input as islands in this particular proof planning mode. If the system can successfully complete the proof it is accepted – and otherwise not.

Active problem solving and exploration play an essential role in apprenticeship learning and interactive service systems such as a proof planner or a computer algebra system can be used as cognitive tools here. [4]

As its name suggests, ActiveMath emphasizes the active role of the student and this feature is supported inter alia by the integration of some computer algebra systems, MuPAD and the proof planner Multi. They provide the backbone for interactive problem solving and for dynamically producing feedback to the user's actions. For university students as users, the philosophy of ActiveMath suggests that the user controls her exercise activities herself and no single pre-determined solution needs to be followed as long as any correct solution results eventually [Buedenbender *et al.*, 2002].

Standard theorem proving systems could, in principle, be employed for the exploration of mathematical proofs as well, however essential obstacles that prevent most of the systems from being used for mathematics (except for logic) learning are their low-level logical input language, the small-grained logic-level inference steps, and the poor user interface. Some user interfaces such as PCoq [PCoq, ongoing] are useful for experts who want to prove a theorem, but not for students who want to learn and understand how theorems from an undergraduate textbook, say, should be proved. Similarly, the size and nature of the inference steps (resolution or natural deduction) are too fine grained to be used in a classroom exercise.

[4] The term *cognitive tool* was coined in [Lajoie and Derry, 1993] and generally denotes instruments supporting cognitive processes by extending the limits of the human cognitive capacities, e.g., the working memory.

In learning mathematical proof we need a higher level of abstraction including the particular mathematical vernacular of the area to be taught. This is where proof planning comes into play. For proof planning, there is empirical evidence that instruction with proof planning examples, which work at a higher level of abstraction, can be effective for learning [Melis *et al.*, 2001c].

Moreover, in order to support the student's discovery of failed proof attempts and their repair we made explicit some information that was implicitly available in the proof planning process and we designed messages about failures that can be useful for a learner. Failures include failing meta-level application conditions of methods (e.g., inconsistency of collected constraints), failing object-level conditions of proof planning methods (e.g., missing precondition), as well as erroneous input for the construction of mathematical objects (instantiations of meta-variables). Moreover, it includes meta-reasoning about ways to backtrack, subsequent introduction of a case split into the proof plan, proof by analogy, or not yet sufficiently determined meta-variables.

Feedback. Theorem proving systems and computer algebra systems can be used to automatically check the student's input and return a 'correct' ('incorrect') as, e.g., in the systems MathDox [Cohen *et al.*, 1999] ActiveMath [Melis *et al.*, 2001b] and EPGY [Sommer *et al.*, 2000]. These automated problem solvers can also serve as back-engines for generating example proofs or example computations.

Although classical theorem provers based on a machine oriented logic are not very useful for teaching mathematical proofs. They are used to advantage however in courses on logic as for example eTPS [TPS, 2004].

Meta-Reasoning. Although still a far cry from effective meta-reasoning, first attempts to present or to employ Polya's problem solving heuristics have been made. Cairns [Cairns and Gow, 2001] gives an interpretation of Polya's stages in his presentation of a proof that includes pointers to prerequisites and to applications of some theorem. Similarly, ActiveMath uses Polya's stages in a presentation-scenario that provides structure and links to related topics. In the Polya-presentation scenario, the stages *Understand the Problem*, *Devise a Plan*, *Carry out the Plan*, and *Look Back at the Solution* are realised by assembling certain types of learning objects by their metadata. This way, proof examples and exercises can be presented at the appropriate stage of searching for a solution to the given problem.

Cognitively Adequate Presentations. An adequate presentation of a proof for education would have to address at least the following issues:

- hierarchically structuring a (complex) proof to make it more compre-

hensible

- emphasize the proof *process* in order to support the student's proof activities and derivational transfer.

- cognitive overload

As for hierarchical structuring, the presentation of proof trees in a box style is a good way to make the structure visible. However, the box structure is logic-oriented and does not help much for mathematical proofs for which a hierarchy helps to separate and express (and remember) proof ideas from details.

Most current approaches to natural language presentation of proofs [Dahn, 1998; Fiedler, 1999; PCoq, ongoing] target a presentation similar to proofs in a book. However, these presentations are result-oriented rather than process-oriented. More abstraction is introduced by presentations based on tactics using proof plans such as in [Holland-Minkley *et al.*, 1999] and [Melis and Leron, 1999].

As for emphasizing the proof process, presenting the search for a *partial proof plan* is often useful, because it makes the proof situation explicit. For instance, the collection of constraints of a meta-variable could be presented in order to provide a clue on how to (interactively) construct a mathematical object [Zimmer and Melis, 2004]. This can help the student to *construct an object* which is the most difficult task in many proofs.

4 Challenges and Future Work

Tools for learning mathematics (and logic) are leaving the lab to be used in practice,[5] however there is still plenty of room for improvement. The following open or only partially solved problems represent a spectrum from the user-centered design of user interfaces and presentations up to the formalization of pedagogical strategies. Most of these challenges require interdisciplinary research.

Proof Presentation and User Interfaces. Learning mathematics involves people, situations, and goals that are very different from those presupposed for an automated theorem proving system. Apart from other things, learning may have the goal to understand a given proof or find a proof. In order to help the student to *understand* solutions and proofs, we need a comprehensible proof presentation at various levels of abstraction, detail and explanation upon request. In order to support *learning by doing*,

[5] For example the geometry tutor from Carnegie Mellon is used now in more than 2000 schools in the US.

however, a better proof presentation alone is not sufficient as it requires more advanced user-adaptive user interfaces specially designed for learning.

Currently the design of a user interface for a theorem proving system typically starts with the special technical features and capabilities of the underlying system rather than with a user-centered approach. This is not just our own unfortunate experience with the user interface of the ΩMEGA system [Siekmann *et al.*, 1999], which is inadequate for an average student. A student concentrates on learning and problem solving and any focus on the tool produces a cognitive overload that gets in the way of learning and mathematical skills. The structured (and foldable) hypermedia presentation of worked-out problem solutions as well as the presentation of information relevant to the problem solving *process* is in fact much better than a traditional textbook proof, which is in the tradition of minimalistic proof presentation developed over the centuries in mathematics.

As part of the user interface, a simple to use input editor is needed. Some work in this direction was done at RISC [Nakagawa and Buchberger, 2001] with the Mathematica functionality, in Nice [Dirat *et al.*, 2000], in Grenoble [Nicaud *et al.*, 2002], and for ACTIVEMATH which by now features a full-fledged palette-based input editor that generates OpenMath.

Feedback in Problem Solving. A system should reason about the student's input and not just use pre-computed solutions and proofs in order to guide the student's problem solving. Moreover, it is certainly not sufficient to just respond 'correct' or 'incorrect'. Interesting feedback has to include the provision of counter-examples, similar proofs, explanations and hints.

Meta-Reasoning. The integration of heuristics and meta-reasoning into a learning tool for learning mathematics is still a challenge. An advanced tool support might offer means far beyond Polya's ideas. For instance, the student could use an online search tool for the Internet to find similar problems, analogous solutions, or the concepts which are prerequisites for her proof. Semantic search techniques, managing (little) theories, browsing theories, and maintaining and managing mathematical ontologies are currently research topics of the MKM Conference series and as partially implemented in ACTIVEMATH search/dictionary tool.

Proof Planning In order to use proof planning in an educational setting, several research directions are promising, among them

- more advanced support for reasoning about failed proof attempts and appropriate support for revising a proof plan,
- support for checking whether a path is heuristically promising or dead-ended altogether,

- support for the construction of mathematical objects.
- Island planning as suggested in [Melis, 1996] is a good starting point in order to develop a proof idea first and to leave the rest of the details to the proof system.

5 Conclusion

The student is not like me.[6] It is insufficient to rely on our own intuition on what may be useful or not. It is also not enough to use standard psychological results about human learning when designing a learning tool. The actual proof of usefulness and usability comes from observations on how students actually use a system and from empirical evidence in controlled experiments that measure the effect of a system.

BIBLIOGRAPHY

[PCoq, ongoing] http://www-sop.inria.fr/lemme/pcoq/.

[Aczel *et al.*, 1999] J.C. Aczel, P. Fung, R. Bornat, M. Oliver, T. OShea, and B. Sufrin. Using computers to learn logic: Undergraduates experiences. In G. Cumming, T. Okamote, and L. Gomez, editors, *Advanced Research in Computers and Communications in Education. Proccedings of the 7th International Conference on Computers in Education*, Amsterdam, 1999. IOS Press.

[Koedinger et al, 1997] K.R. Koedinger, J.R. Anderson, W.H. Hadley and M.A. Mark. Intelligent Tutoring Goes to School in the big City. *International Journal of Artificial Intelligence in Education*, vol 8: 30–43, 1997.

[Aleven and Koedinger, 2000] V. Aleven and K.R. Koedinger. The need for tutorial dialog to support self-explanation. In C. P. Rose and R. Freedman, editors, *Building Dialogue Systems for Tutorial Applications*, AAAI Fall Symposium, pages 65–73. AAAI Press, 2000.

[Anderson *et al.*, 1995] J. Anderson, A. Corbett, K. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[Anderson and Pelletier, 1991] J.R. Anderson and R. Pelletier. A development system for model-tracing tutors. In L. Birnbaum, editor, *The International Conference on the Learning Sciences*, 1991.

[Andriessen and Sandberg, 1999] J. Andriessen and J. Sandberg. Where is education heading and how about AI. *International Journal of Artificial Intelligence in Education*, 10:130–150, 1999.

[Baumert *et al.*, 1997] 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.

[Baumgartner, 2003] P. Baumgartner, editor. *IJCAI workshop on Knowledge Representation and Deduction for Education.* AAAI, 2003.

[Baumgartner and Furbach, 2003] P. Baumgartner and U. Furbach. Automated deduction techniques for the management of personalized documents. *Annals of Mathematics and Artificial Intelligence, Special Issue*, 38(1-3):211–228, 2003.

[Bloom, 1956] B.S. Bloom, editor. *Taxonomy of educational objectives: The classification of educational goals: Handbook I, cognitive domain.* Longmans, Green, New York, Toronto, 1956.

[6] Mantra from a personal communication with Ken Koedinger.

[Bornat,] R. Bornat. The JAPE web site. http://www.jape.org.uk.

[Brusilovski, 1996] P. Brusilovski. Methods and techniques of adaptive hypermedia. *User MOdeling and User Adapted Interaction*, 6(2-3):87–129, 1996.

[Buchberger *et al.*, 1997] B. Buchberger, T. Jebelean, F. Kriftner, M. Marin, E. Tomuta and D. Vasaru. An Overview of the Theorema Project. Proceedings if ISSAC, 1997.

[Buedenbender *et al.*, 2002] J. Buedenbender, E. Andres, A. Frischauf, G. Goguadze, P. Libbrecht, E. Melis, and C. Ullrich. Using computer algebra systems as cognitive tools. In S.A. Cerri, G. Gouarderes, and F. Paraguacu, editors, *6th International Conference on Intelligent Tutor Systems (ITS-2002)*, number 2363 in Lecture Notes in Computer Science, pages 802–810. Springer-Verlag, 2002.

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

[Caprotti and Cohen, 1998] O. Caprotti and A. M. Cohen. Draft of the OpenMath standard. OpenMath Consortium, http://www.nag.co.uk/projects/OpenMath/omstd/, 1998.

[Catrambone and Holyoak, 1990] R. Catrambone and K.J. Holyoak. Learning subgoals and methods for solving probability problems. *Memory and Cognition*, 18(6):593–603, 1990.

[Chi *et al.*, 1989] M.T. Chi, M. Bassok, M.W. Lewis, P. Reimann, and R. Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182, 1989.

[China, 2005] China Network Information Centre. Statistic Report on the Development and Status of Networks in China, 2005.

[Cohen *et al.*, 1999] A. Cohen, H. Cuypers and H. Sterk. *Algebra Interactive*, Springer-Verlag, 1999.

[Conati *et al.*, 1997] C. Conati, A.S. Gertner, K. VanLehn, and M. Druzdzel. On-line student modeling for coached problem solving using Baysian networks. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference, UM97*, pages 231–242, 1997.

[Dahn, 1998] B.I. Dahn. Using ILF as a user interface for many theorem provers. In R.C. Backhouse, editor, *Proceedings of the Workshop on User Interfaces for Theorem Provers*, volume 98-08 of *Eindhoven University of Technology*, pages 75–86. Department of Mathematics and Computing Science, 1998.

[Dirat *et al.*, 2000] L. Dirat, M. Buffa, J.-M. Fedou, and P. Sander. Jome, un composant logiciel pour le tele-enseignement des mathematiques via le web. In *Actes du Colloque Internationalal sur les Technologies de l'Information et de la Communication dans les Enseignements d'Ingenieurs et dans l'Industrie (TICE*, pages 7–16, 2000.

[Fiedler, 1999] A. Fiedler. Using a cognitive architecture to plan dialogs for the adaptive explanation of proofs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*. Morgan Kaufman, 1999.

[Forgy, 1982] C.L. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, pages 17–37, 1982.

[Franke and Kohlhase, 2000] A. Franke and M. Kohlhase. MBase: Representing mathematical knowledge in a relational data base. In F. Pfenning, editor, *Proc. 17th International Conference on Automated Deduction (CADE)*, Lecture Notes on Artificial Intelligence. Springer-Verlag, 2000.

[Friedman-Hill, 1997] E. Friedman-Hill. Jess, the java expert system shell. Technical Report SAND98-8206, Sandia National Laboratories, 1997.

[Holland-Minkley *et al.*, 1999] A.M. Holland-Minkley, R. Barzilay, and R.L. Constable. Verbalization of high-level formal proofs. In *National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[Hoppe, 1994] U. Hoppe. Deductive error diagnosis and inductive error generalization for intelligent tutoring systems. *Journal of Artificial Intelligence in Education*, 5(1):27–49, 1994.

[Jacobs, 2001] B. Jacobs. Aufgaben stellen und Feedback geben. Technical report, Medienzentrum der Philosophischen Fakultät der Universität des Saarlandes, 2001.

[Kobsa and Pohl, 1995] A. Kobsa and W. Pohl. The user modeling shell system bgp-ms. *User Modeling and User Adapted Interaction*, 4:59–106, 1995. http://www.ics.uci.edu/ kobsa/papers/1995-UMUAI-kobsa.ps.gz.

[Kohlhase, 2000] M. Kohlhase. `OMDoc`: Towards an internet standard for the administration, distribution and teaching of mathematical knowledge. In *Proceedings Artificial Intelligence and Symbolic Computation AISC'2000*, 2000.

[Lajoie and Derry, 1993] S. Lajoie and S. Derry, editors. *Computers as Cognitive Tools*. Erlbaum, Hillsdale, NJ, 1993.

[Leron, 1983] U. Leron. Structuring mathematical proofs. *The American Mathematical Monthly*, 90:174–185, 1983.

[Lowe and Duncan, 1997] H. Lowe and D. Duncan. XBarnacle: Making theorem provers more accessible. In W. McCune, editor, *Proceedings of the Fourteenth Conference on Automated Deduction (CADE)*, volume 1249 of *Lecture Notes in Artificial Intelligence*, pages 404–408. Springer, 1997.

[Maple, 1986] B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet and M.B. Monagan. A Tutorial Introduction to MAPLE. *Journal of Symbolic Computation*, 2(2):179–200, 1986.

[Matsuda and vanLehn, 2005] N. Matsuda and K. vanLehn Advanced Geometry Tutor: An intelligent Tutor that Teaches Proof-Writing with Construction. *Artificial Intelligence in Education*, Ch-K. Looi, G. McCalla, B. Bredewig and J.Breuker (editors) pages 443–450, IOS Press, 2005.

[McCune, 1990] W. W. McCune. Otter 2.0 Users' Guide. Argonne Natioanl Laboratory 1990. ANL-90/9, Maths and CS Division, Argonne, Illinois.

[Mandl *et al.*, 1997] H. Mandl, H. Gruber, and A. Renkl. *Enzyklopädie der Psychologie*, volume 4, chapter Lernen und Lehren mit dem Computer, pages 436–467. Hogrefe, 1997.

[Meier and Melis, 2005] A. Meier and E. Melis. Failure reasoning in multiple-strategy proof planning. In *Electronic Notes in Theoretical Computer Science*, M. Bonacina and T. Boy de la Tour, eds. pp. 67–90, Elsevier, 2005.

[Melis, 1996] E. Melis. Island planning and refinement. Seki Report SR-96-10, Universität des Saarlandes, FB Informatik, 1996.

[Melis, 2000] E. Melis, editor. *Proceedings of CADE-17 Workshop on Deduction in Education*, 2000.

[Melis, 2005] E. Melis. Why Proof Planning for Maths Education and How? In D. Hutter and W. Stephan, editors, *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg Siekmann*, Lecture Notes in Artficial Intelligence, no 2605, pp. 364–376. Springer-Verlag, 2005.

[Melis and Andres, 2004] E. Melis and E. Andres. Global Feedback in ActiveMath. *Internatioal Journal of Computers in Mathematics and Science Teaching*, 24:197–220, 2005.

[Melis and Leron, 1999] 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.

[Melis and Meier, 2000] E. Melis and A. Meier. Proof planning with multiple strategies. In *First Internatioanl Conference on Computational Logic*. J. Lloyd, V. Dahl and U. Furbach, eds. LNAI vol 1861. pp. 644–659, 2000.

[Melis *et al.*, 2001a] E. Melis, E. Andres, A. Franke, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath system description. In *Artificial Intelligence and Education*, pages 580–582, 2001.

[Melis *et al.*, 2001b] E. Melis, J. Buedenbender, E. Andres, A. Frischauf, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. ActiveMath: A generic and adaptive web-based learning environment. *Artificial Intelligence and Education*, 12(4):385–407, winter 2001.

[Melis *et al.*, 2001c] 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.

[Melis *et al.*, 2003] E. Melis, J. Buedenbender, E. Andres, A. Frischauf, G. Goguadse, P. Libbrecht, M. Pollet, and C. Ullrich. Knowledge representation and management in ActiveMath. *International Journal on Artificial Intelligence and Mathematics, Special Issue on Management of Mathematical Knowledge*, 38(1-3):47–64, 2003.

[Melis and Ullrich, 2003] E. Melis and C. Ullrich. Polya-scenarios in ActiveMath. In *AI in Education, AIED-2003*. U. Hoppe, F. Verdejo, and J. Kay, editors, pp. 141–147. IOS Press, 2003.

[Murray *et al.*, 1999] T. Murray, C. Condit, T. Shen, J. Piemonte, and S. Khan. Metalinks - a framework and authoring tool for adaptive hypermedia. In S.P. Lajoie and M. Vivet, editors, *Proceedings of the International Conference on Artificial Intelligence and Education*, pages 744–746. IOS Press, 1999.

[Nakagawa and Buchberger, 2001] K. Nakagawa and B. Buchberger. Presenting proofs using logicographic symbols. In A. Fiedler and H. Horacek, editors, *IJAR-2001 Workshop on Proof Transformation and Presentation*, 2001.

[Narciss, 2001] S. Narciss. Informatives Tutorielles Feedback. Habilschrift,Technische Universität Dresden, Fak. Mathematik und Naturwissenschaften, 2004.

[Nicaud *et al.*, 2002] J-F. Nicaud, D. Bouhineau, and T. Huguet. Aplusix-editor: A new kind of software for the learning of algebra. In S.A. Cerri, G. Gouarderes, and F. Paraguacu, editors, *Intelligent Tutoring Systems, 6th International Conference, ITS2002*, volume 2363 of *LNCS*, pages 178–187, 2002.

[Piaget, 1977] J. Piaget. *Equilibration of Cognitive Structures*. Viking, New York, 1977.

[Pollet *et al.*, 2003] M. Pollet, E. Melis, and A. Meier. User interface for adaptive suggestions for interactive proof. In *Proceedings of the International Workshop on User Interfaces for Theorem Provers (UITP)*, pp. 133–142, 2003.

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

[Schoenfeld, 1985] A.H. Schoenfeld. *Mathematical Problem Solving*. Academic Press, New York, 1985.

[Schoenfeld, 1990] A.H. Schoenfeld, editor. *A Source Book for College Mathematics Teaching*. Mathematical Association of America, Washington, DC, 1990.

[Sieg and Byrnes, 1996] W. Sieg and J. Byrnes. Normal natural deduction proofs. Technical Report CMU-PHIL-74, Department of Philosophy, Carnegie Mellon University, Pittsburgh, 1996.

[Sieg and Byrnes, 1998] W. Sieg and J. Byrnes. Normal natural deduction proofs (in classical logic). *Studia Logica*, 60:67–106, 1998.

[Siekmann *et al.*, 1999] J. Siekmann, S. Hess, Ch. Benzmüller, L. Cheikhrouhou, A. Fiedler, M. Kohlhase H. Horacek, K. Konrad, A. Meier, E. Melis, and V. Sorge. LΩUI: Lovely ΩMEGA User Interface. *Formal Aspects of Computing*, 11(3):326–342, 1999.

[Sommer *et al.*, 2000] R. Sommer, M. Rozenfeld, and R. Ravaglia. A proof environment for teaching mathematics. In E. Melis, editor, *Deduction in Education, CADE-17 workshop*, pages 35–43, 2000.

[Sorgatz and Hillebrand, 1995] A. Sorgatz and R. Hillebrand. MuPAD – Ein Computeralgebra System I, *Linux Magazin*, 12/95, 1995. http://www.geo.uni-bonne.de/software/wwpad/BIB

[Suppes, 1981] P. Suppes, ed. University-level Computer-assisted Instruction at Stanford: 1968–1980. Stanford. Institute for Mathematical Studies in teh Social Sciences. 1981.

[Thery *et al.*, 1992] L. Thery, Y. Bertot, and G. Kahn. Real theorem provers deserve real user-interfaces. Rapports de Recherche 1684, Institute National de Recherche en Informatique et en Automatique, Sophia Antipolis, 1992.

[TPS, 2004] P.B. Andrews, C.E. Brown, F. Pfenning, M. Bishop, S. Issar and H. Xi. ETPS: A System to Help Students Write Formal Proofs. *Journal of Automated Reasoning*, 32: 75–92, 2004

[Tsovaltzi, 2005] A. Fiedler and D. Tsovaltzi. Domaion-Knowledge Manipulation for Dialogue-Adaptive Hinting. *Artificial Intelligence in Education*, C.-K. Looi et al. (editors), IOS Press, pages: 801–803, 2005.

[Ullrich, 2003] C. Ullrich. Pedagogical rules in ActiveMath and their pedagogical foundations. Seki Report SR-03-03, Universität des Saarlandes, FB Informatik, 2003.

[Ullrich and Melis, 2002] C. Ullrich and E. Melis. The Poor Man's Eyetracker in ActiveMath. *Proceedings of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (eLearn-2002)*, vol.4, pages:2313–2316, 2002.

[Ullrich *et al.*, 2004] C. Ullrich, P. Libbrecht, S. winterstein and M. Muehlenbrock. A flexible and efficient presentation architecture for adaptive hypermedia: description and technical evaluation. *Proceedigns fo the 4th IEEE Internatioanl Conference on Advanced Learning Technologies (ICALT 2004)*. pp. 21–25, Kinshuk, C. Looi, E. Sutinen, D. Sampson, eds., 2004

[VanLehn *et al.*, 2001] K. VanLehn, S. Siler, C. Murray, T. Yamauchi, and W.B. Baggett. Human tutoring: Why do only some events cause learning? *Cognition and Instruction*, 2001.

[VanLehn *et al.*, 2002] K. VanLehn, C. Lynch, L. Taylor, A. Weinstein, R. Shelby, K. Schulze, D. Treacy, and M. Wintersgill. Minimally invasive tutoring of complex physics problem solving. In S.A. Cerri, G. Gouarderes, and F. Paraguacu, editors, *Intelligent Tutoring Systems, 6th International Conference, ITS 2002*, number 2363 in LNCS, pages 367–376. Springer-Verlag, 2002.

[Vygotsky, 1978] L. Vygotsky. *The Development of Higher Psychological Processes*. Harvard University Press, Cambridge, 1978.

[Weber and Brusilovsky, 2001] G. Weber and P. Brusilovsky. ELM-ART an adaptive versatile system for web-based instruction. *Artificial Intelligence and Education*, 2001.

[White and Shimoda, 1999] B.Y. White and T.A. Shimoda. Enabling students to construct theories of collaborative inquiry and reflective learning: Computer support for metacognitive development. *International Journal of Artificial Intelligence in Education*, 10:151–182, 1999.

[Zimmer and Melis, 2004] J. Zimmer and E. Melis. Constraint solving for proof planning. *Journal of Automated Reasoning*, 33:51–88, 2004.