Adaptive Course Generation and Presentation

The ΩMEGA group:
Jörg Siekmann, Christoph Benzmüller, Armin Fiedler,
Andreas Franke, George Goguadze, Helmut Horacek, Michael Kohlhase,
Paul Libbrecht, Andreas Meier, Erica Melis, Martin Pollet,
Volker Sorge, Carsten Ullrich, Jürgen Zimmer
Universität des Saarlandes/DFKI Saarbrücken
Germany

Abstract

Today's interactive mathematics textbooks use a collection of predefined documents, typically organized as a network of HTML pages. This makes a reuse and a sound re-combination of the encoded knowledge impossible and inhibits a radical adaption of course presentation and content to the user's needs. In order to avoid these drawbacks we have designed a web-based framework for dynamically producing interactive documents for learning mathematics called ID. The system design relies on the separation of knowledge representation from system functionalities. Salient features of our system are the individual generation of interactive documents based on general domain knowledge, user-specific preferences and the user's knowledge as well as the integration of external problem solving systems. The paper describes the distributed web-based architecture of our system and the principles of its components.

1 Introduction

Today's interactive mathematics textbooks [4, 5] use a collection of predefined documents including a fixed set of examples and exercises. Neither can the content (e.g., examples and exercises) be reused and tailored to the needs of a particular user nor can the presentation of these documents be adapted flexibly enough.

In order to allow reuse and adaptation of the learning material we have designed a web-based framework for dynamically producing interactive documents for learning mathematics [14]. The system design is characterized by the separation of knowledge representation from system functionalities. For the interactive mathematics document ID the knowledge comprises mathematical facts and methods as well as knowledge about the expertise of a particular user and pedagogical knowledge.

From these knowledge sources the system can compose individuated versions of interactive documents dynamically. The generation of an interactive textbook or course may include selectively composed parts in varying presentation form that differ, for example, in language, in notational preferences, in the multi-modal display, in the availability of expert services for solving subproblems, or in their depth of explanation. The generation may also be adaptive according to the context in which it is used, such as a classroom setting or stand-alone use.

Our system allows for exploratory problem solving rather than on mere content learning by looking up solutions, multiple choice for solutions. That is, the user can solve problems or discover proofs interactively within the system. The system supports the user by checking the correctness of proof steps and of the overall solution as well as by heuristically guiding the problem solving process. A more detailed discussion of exploratory problem solving supported by expert problem solving systems is given in [11].

In this paper we concentrate on the description of the architecture of ID, the representation format, and we present the session manager and the realizer that dynamically assemble interactive mathematics documents (not only single pages, but also whole textbooks and courses) by using information retrieved from a knowledge base, a user model, and a pedagogical module.

The paper is organized as follows: first we give an overview on the architecture of our system ID. Then, we detail the knowledge representation format of the underlying mathematical knowledge base. In section 3.1 we describe what kinds of information are used in the document creation. The document creation and the functionalities of the session manager and the realizer are explained in section 3.2. Finally, we give some examples to illustrate the benefits of our approach.

2 Content Representation

Current interactive mathematics textbooks and courses are at best based on a fix collection of predefined HTML-pages [4] or LaTeX-units [5] which are augmented by navigation features. Only few of them allow for a generation of documents from a particular subset of pages, but these few exceptions still use ready-to-present pages.

As opposed to these interactive mathematics textbooks our approach requires a representation of standard formalized mathematics in a (possibly distributed) knowledge base. Though this standardization requires some extra work it provides an ontology for the content of the course which is indispensable for a reuse of teaching or learning material and for a combination of material from different courses. Moreover, the presentation can be far more flexible when based on a unified representation in XML-format. In the following, we describe our content representation called OMDoc.

OMDoc is an extension of the emerging OpenMath standard [3] (see `http://www.openmath.org`), an XML-based general framework for encoding mathematical objects. OMDoc extends the OpenMath representation for mathematical objects by the representations of mathematical facts such as theorems, definitions, proof methods, and proofs. It includes natural language formulations as well as logical forms, the latter as OpenMath objects. Figure 1 shows an OpenMath representation of the law of commutativity for add-operation in the real numbers (the logical formula $\forall a, b.a + b = b + a$). The XML element `OMBIND` is used for the OpenMath binding construct (in this case used for universal quantification) and the `OMS`, `OMV` and `OMA` are the representations for logical constants,[1] variables and (function) applications. As an example, Figure 2 shows a definition of a monoid. The `CMP` (commented mathematical property) element is used for the natural language representation of the definition (including OpenMath representations for the objects involved) and the `FMP` (formal mathematical property) contains an OpenMath object (shown as ... in Figure 2) that encodes a logical formula similar to in Figure 1 that formalizes the content of the `CMP` (we have not included it for space reasons). For details on OMDoc, see [9].

3 The Open Architecture

The distributed setup of the system employs the MathWeb[2] system [8, 7] for distributed automated theorem proving that connects a wide range of *mathematical services* by a common *mathematical software bus*. The MathWeb system provides the functionality to turn existing theorem proving systems and mathematical tools into services that are homogeneously integrated into a networked proof development environment. The environment thus gains the services from these particular modules, but each module in turn gains from using the features of other plugged-in components. Currently, MathWeb integrates several mathematical services, a proof planner [12], several computer algebra systems, the $\mathcal{L}\Omega\mathcal{UI}$ graphical user interface for interactive theorem provers [15], and MBase [10], a knowledge base that stores and offers a universal repository of formalized mathematics that consists of theorems, definitions, proof methods, and proofs. The formal representation of MBase allows a semantics-based retrieval of mathematical knowledge. For

[1] OpenMath symbols, including the `cd` reference to the home theory of the mathematical concepts represented by these symbols.

[2] The system is available from `http://www.mathweb.org/mathweb`

```
<OMOBJ>
  <OMBIND>
    <OMS cd="quant1" name="forall"/>
    <OMBVAR>
      <OMV name="a"/>
      <OMV name="b"/>
    </OMBVAR>
    <OMA><OMS cd="logic1" name="implies"/>
      <OMA><OMS cd="logic1" name="and"/>
        <OMA><OMS cd="set1" name="in"/><OMV name="a"/><OMS cd="barshe" name="real"/></OMA>
        <OMA><OMS cd="set1" name="in"/><OMV name="b"/><OMS cd="barshe" name="real"/></OMA>
      </OMA>
      <OMA><OMS cd="relation" name="eq"/>
        <OMA><OMS cd="barshe" name="plus-real"/><OMV name="a"/><OMV name="b"/></OMA>
        <OMA><OMS cd="barshe" name="plus-real"/><OMV name="b"/><OMV name="a"/></OMA>
      </OMA>
    </OMA>
  </OMBIND>
</OMOBJ>
```

Figure 1: An OpenMath representation of $\forall a, b.a + b = b + a$.

```
<definition id="ida.c6s1p4.d1"
            item="monoid" theory="ida.monoid">
  <CMP>
    A structure
    <OMOBJ><OMA>
      <OMS cd="cartesian-products" name="triple"/>
      <OMV name="M"/><OMV name="*"/><OMV name="e"/>
    </OMA></OMOBJ>,
    in which
    <OMOBJ><OMA>
     <OMS cd="cartesian-products" name="pair"/>
     <OMV name="M"/>
     <OMV name="*"/>
    </OMA></OMOBJ>
   is a semi-group with unit <OMOBJ><OMV name="e"/></OMOBJ>, is called a
   <defemph>monoid</defemph>.
  </CMP>
  <FMP><OMOBJ> \dots </OMOBJ></FMP>
</definition>
```

Figure 2: A definition of a monoid in OMDoc representation

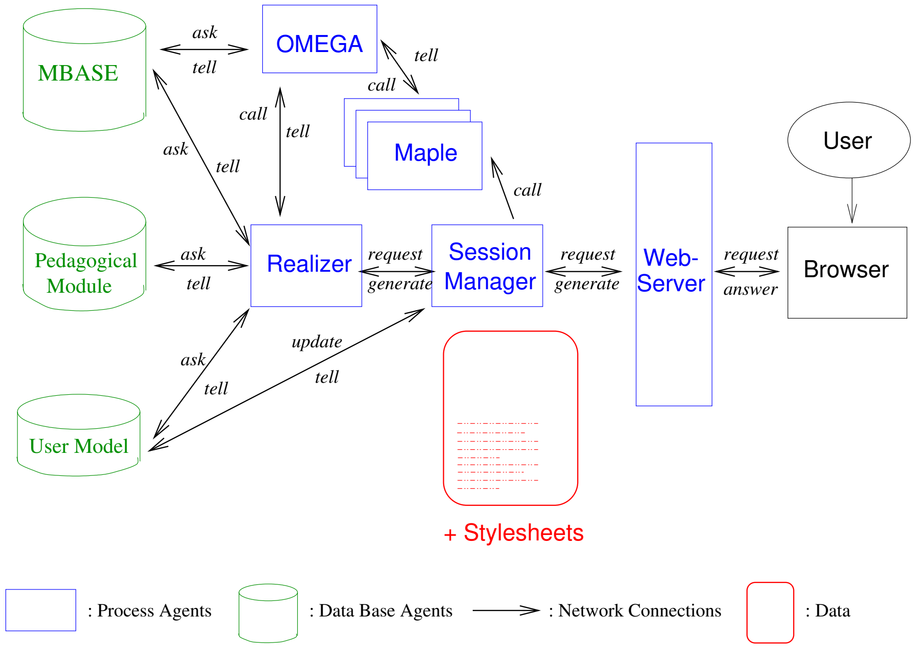

Figure 3: Architecture of ID

realizing an interactive mathematics course, the MathWeb environment additionally comprises a user model, a pedagogical module, a session manager, and a realizer.

Figure 3 depicts the architecture of ID. It gives an overview of the services used in the context of web-based user-adaptive interactive textbooks and courses and of the communications between these services. The right hand of the figure shows a standard client-server web-architecture which handles requests of the user via a browser, a web-server. All MathWeb services can communicate over the Internet by a standardized protocol based on Xml [2] and Kqml [6].

The session manager administers the current session. Session tracking and simple request such as login or changing display preferences are handled by the session manager. More complex tasks concerning document creation and alteration are passed to the realizer. The realizer dynamically creates the preliminary OMDoc-documents to be presented to the user by the browser.

3.1 Communicated Information

The realization of several functionalities requires various kind of information from several knowledge sources.

Content Information *Content information* comprises the mathematical information to be presented to the user (in ID: mathematical concepts, theorems, examples, exercises, proof methods, and proofs). It can be received from several information sources, namely (1) from the database MBase containing the knowledge items and relations specifying their structure, and (2) from service systems, such as the proof planner ΩMEGA or the computer algebra system Maple [13] who can deliver a proof or the result of a computation.

User Information The *user information* consists of the relevant parts of the user model and *session specific information*. It comprises the following knowledge about the user: her preferences, learning capabilitie, and goals provided by the user upfront, her domain knowledge, and the knowledge about the session history.

Some knowledge kept in the user model is rather stable or it is at least slowly changing, in particular knowledge about the user's general capabilities and preferences, e.g., the suitable level of abstraction for a presentation, how much explanation is necessary as well as general preferences, e.g., whether the user prefers natural language over formulas and which modalities to choose in multi-modal presentation.

The session-specific information (e.g., what is the current/previous/next page) is needed for the on-line generation of ID. From this information, conclusions for the navigation support can be inferred (e.g., indicating the mathematical concepts accessible by the user with her current domain knowledge).

First we shall implement the user model as a combination of an annotated session list and a simple table-based overlay-model since the user modeling component is not our focus currently. As the user modeling component will be realized as a MathWeb database object with a clearly defined interface, it is easily replaceable by a more sophisticated component.

Pedagogical Information The *pedagogical information* contains the pedagogical know-how about which content to retrieve for a specific type of user. It covers the know-how of how to present the content in a way adapted to the user's capabilities. It also determines under which conditions previously visited lessons should be (partially) repeated or remembered, when to ask questions, and under which conditions which service systems should be available for the user. The pedagogical information may also contain some knowledge on which presentation is to be chosen. How the pedagogical information will be represented is not yet decided. General information such as "For a novice, choose easy examples" will probably be represented by a set of rules.

3.2 Dynamic Document Generation

For the document creation, the realizer processes content information, user information, and pedagogical information.

In order to answer a request, the realizer requests content information sources from MBase and transforms the raw data provided by MBase into a document in OMDoc format representing the sequential content. The realizer then computes XSL[18]- and CSS[1]-style-sheets. This content creation is performed according to the intended presentation functionality and the user model. Then the realizer sends the OMDoc-document and style-sheets to the session manager. The session manager sends the start page via the web-server to the browser who displays the content using the style-sheets. Simple requests, such as next or previous page, can be handled directly by the session manager. More complex requests requiring new content generation, such as explaining a concept in more depth or generation of more examples, are passed to the realizer. During a session some interaction information, such as learned concepts or problem solving results, is passed to the user model where it is used for updating.

When the session manager receives a request requiring content creation, the request is passed to the realizer. A document in generated in five steps:

1. Dependent on the user and pedagogical information, mathematical content is retrieved from information sources like MBase. For example, all the MBase items on which a requested concept depends and that are not marked as known in the user model are collected.

2. The collection of content items is then processed according to the information in the user model and the pedagogical module. For example, the suitable level of abstraction is chosen. If more content (such as exercises, remarks, and examples) is needed, the first step can be repeated. In this step interaction facilities are chosen too, e.g., the availability of a service system for solving a task.

3. Then this structure is linearized. The user model and the pedagogical module influence the linearization, as the amount to be presented on one page will vary according to the users preferences and capabilities and different teaching strategies will require different linearizations. After this step, the course is represented as a sequence of OMDoc-documents.

4. style-sheets that determine the presentation features are generated according to the information in the user model and the pedagogical module.

5. The OMDoc-documents and style-sheets are passed to the session manager. The session manager then adds further components to the single OMDoc-pages, for example, navigation buttons. This step mainly uses information about the session.

As the result of the document creation one or more documents in OMDoc-format are generated as well as style-sheets. The documents can be presented to the user by the browser using the style-sheets.

3.3 Adaptive Functionalities

Some possible dimensions of the adaptation performed by the realizer are

- the choice of the content to be presented (e.g., guided tours explained below, adaptive testing, prepared kinds of a textbook),
- the availability of service systems (e.g., a calculator, a computer algebra system, or a proof planner). For example, for routine tasks which are not in the focus of a session, a service can be used.
- the selection of the service functionalities (e.g., which methods can be used by the user when using a proof planner).

These dimensions are discussed again in section 3.4. The adaptations depend on the features of the user and of the overall session, for instance on

- the goal of a session,
- the setting (e.g., standalone use or classroom setting where the teacher can set the session goal),
- the user's capabilities (e.g., how abstract a presentation must be in order to be understood or how much explanation is necessary),
- the user's understanding of concepts (e.g., well understood concepts do not need to be (re)explained)
- the user's preferences (e.g., language, natural language vs. formulas, modalities in multimodal presentation),
- the user's need for motivation.

3.4 Examples

The following examples illustrate how adaptations will look like. The first example shows how different document contents and presentations can be generated from the same mathematical content (see Figures 4 and 5). If the concept *monoid* is presented to an expert, then the presentation contains only the desired concept as shown in the screenshot of Figure 4. If it is presented to a student, then the document whose presentation is shown in Figure 5 contains additional exercises, remarks, and examples.

The following two example sessions illustrate some impacts of the user model and of the pedagogical information.

Jane Doe is a beginner mathematician, she wants to learn about integer modulo n. Her background, as entered into the system is fairly small, mainly set and group theory aside of classic calculus and polynomial algebra.

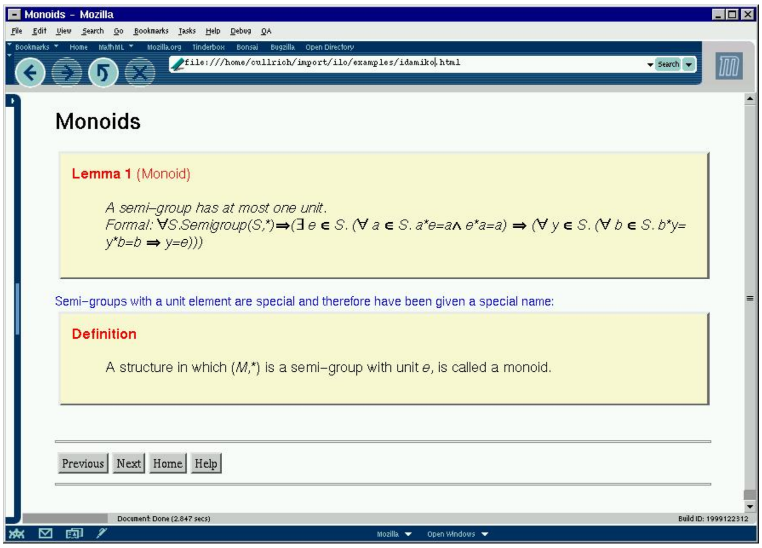

Figure 4: A presentation of Monoids

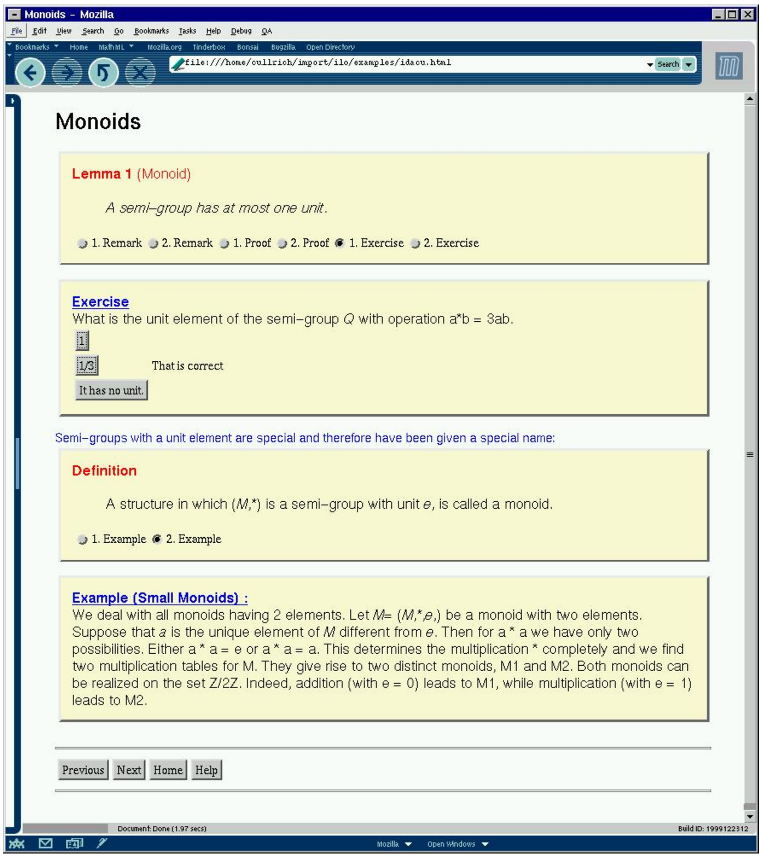

Figure 5: An extended presentation of Monoids

Allan Blees is an engineer. He wants to investigate integers modulo n. His background contains only classical calculus and algebra.

They both select the *guided tour* feature in the menu, asking for $\mathbb{Z}_n$.

The session manager receives the request and associates it with the user information and pedagogical information. It passes these to the realizer as a request to prepare a guided tour. Then the realizer asks MBase to provide several presentable definitions of integers-modulo-n. One of them uses plain equivalence classes, the other uses quotients of a ring by an ideal. Both possibilities are explored till the graph of dependencies for knowledge is rooted in the user's knowledge.

Pedagogical information, especially the fact the Jane Doe is a mathematician, make it clear that the definition of $\mathbb{Z}_n$ would be more interesting for her when presented with ideals and rings. The graph of dependencies shows that it is possible to present the concepts in a quick survey about rings, ideals., and quotient rings. The goal notion, integers modulo n, is then presented to her as an example to quotient ideals. This makes it possible for the realizer to include some theorems, such as the Chinese remainder theorem, as well as other examples and exercises.

Similar to proofs of theorems, the solutions of exercises and examples can be presented in a static way or, if available, in an interactive way that employs a proof-assistant, such as Ωmega. Jane Doe wants to try out this exercise by herself. An action is triggered which launches the user interface on the client side. The proof-assistant is then provided with the conjecture of the exercise and the relevant methods that are supposedly known to the user.

She can now try out the proof planner. Her results and failures are reported to her user modeling component

For Allan Blees, the situation is fairly different. The pedagogical module infers that, as an engineer, he might prefer to solve some more concrete examples instead of learning abstract theories. The realizer inserts more concrete examples. The measure of the concreteness implies that a presentation using $\mathbb{Z}_n$'s definition as simple equivalence classes of numbers is best suited. His guided tour contains several exercises from physics as well as an introduction to the complex-exponential and roots of unity.

Here, the exercises and examples are presented as a static textual solution, an interaction with a computer algebra system such as Maple [13] is possible in places though. The results are reported to the user modeling component.

4 Related Work

The Dynamic Courseware Generator (DCG)[16] is similar to ID in that it generates individual courses according to the user's goals and knowledge. In DCG a planner searches for sub-graphs connecting the goal concept with concepts known by the user. A linearized version of this plan is offered to the user to follow. If a user fails to perform successfully on tests related to a certain concept, new course-plans can be generated. The main difference to ID is the separation of the domain concept structure from the learning material. That is, in DCG concepts have links to fix HtML-pages that present the actual content to be learned, whereas in ID the content is generated from MBase. Moreover, DCG is a general authoring tool and therefore an integration with service systems is not intended.

ELM-ART II[17] is a web-based tutoring system for learning programming in LISP. It provides adaptive navigation support by annotating links in a traffic lights metaphor and the possibility of selecting the next best step in the course. The course itself is fixed, therefore it is not possible to construct courses according to the user's goals as in ID.

The already mentioned interactive mathematics textbooks [4, 5] use a collection of predefined LaTeX documents or HtML pages and include a fixed set of examples and exercises. In [5] courses are split in several pages (slices) that can be combined in order to construct textbooks. The adaptivity consists in selecting those pages leading to a certain goal.

5 Conclusion

This paper describes the design principles of the system ID that dynamically generates interactive documents according to the user's content needs and presentation preferences, ID individually generates interactive documents based on user-specific preferences and the user's knowledge as well as on general domain knowledge. It also provides a tight interaction with service systems. The architecture, the representation of knowledge, and the approach of dynamic generation are well-suited not only for an interactive mathematics textbook but more generally for knowledge-intensive learning and tutor systems whose knowledge acquisition and representation is tedious and should therefore be reused, for systems including (expert) services for exploratory learning, and for systems flexibly generating individual versions of documents to be browsed.

Our system is still under development. MATHWEB and OMDOC are already implemented, while the session manager and the realizer are currently specified and implemented. We are semi-automatically re-coding the content of an existing interactive textbook in OMDOC format.

References

[1] B. Bos, H. Lie, C. Lilley, and I. Jacobs. Cascading Style Sheets, level 2 CSS2 Specification. W3C Recommendation. Technical report, May 12, 1998.

[2] Extensible Markup Language (XML). W3C Recommendation TR-XML, World Wide Web Consortium, December 1997. Available at `http://www.w3.org/TR/PR-xml.html`.

[3] Draft of the Open Math standard. The Open Math Society, `http://www.nag.co.uk/projects/OpenMath/omstd/`, 1998.

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

[5] B.I. Dahn and H. Wolters. *Analysis Individuell.* Springer, Berlin, Heidelberg, 2000.

[6] T. Finin and R. Fritzson. KQML — a language and protocol for knowledge and information exchange. In *Proceedings of the 13th Intl. Distributed Artificial Intelligence Workshop*, pages 127–136, Seattle, WA, USA, 1994.

[7] A. Franke, S. M. Hess, C. G. Jung, M. Kohlhase, and V. Sorge. Agent-oriented integration of distributed mathematical services. *Journal of Universal Computer Science*, 5:156–187, 1999.

[8] A. Franke and M. Kohlhase. System description: MATHWEB, an agent-based communication layer for distributed automated theorem proving. In H. Ganzinger, editor, *16th International Conference on Automated Deduction (CADE-17)*, volume 397, pages 217–221. Springer, 1999.

[9] M. Kohlhase. OMDOC: Towards an OPENMATH representation of mathematical documents. Seki Report SR-00-02, Fachbereich Informatik, Universität des Saarlandes, 2000. `http://www.mathweb.org/ilo/omdoc`.

[10] M. Kohlhase and A. Franke. Mbase: Representing knowledge and context for the integration of mathematical software systems. *Journal of Symbolic Computation, special Issue on the integration of computer algebra and deduction systems*, 11:1–37, 2000.

[11] E. Melis and A. Fiedler. On the benefit of expert services in mathematics education systems. Submitted to the ITS-2000 Workshop on Modeling Human Teaching Tactics and Strategies, 2000.

[12] E. Melis and J.H. Siekmann. Knowledge-based proof planning. *Artificial Intelligence*, 115(1):65–105, November 1999.

[13] Darren Redfern. *The Maple Handbook: Maple V Release 5.* Springer Verlag, Berlin, Germany, 1998.

[14] J. Siekmann. Interaktives Lehrbuch. Project Proposal, July 1999.

[15] J. Siekmann, S.M. Hess, C. Benzmüller, L. Cheikhrouhou, A. Fiedler, H. Horacek, M. Kohlhase, K. Konrad, A. Meier, E. Melis, M. Pollet, and V. Sorge. $\mathcal{LOUI}$: $\mathcal{L}$ovely ΩMEGA $\mathcal{U}$ser $\mathcal{I}$nterface. *Formal Aspects of Computing*, 11:326–342, 1999.

[16] J. Vassileva. Dynamic course generation on the www. In B. d. Boulay and R. Mizoguchi, editors, *Proceedings of AI-ED'97, 8th World Conference on Artificial Intelligence in Education*, pages 498–505, Amsterdam, August 1997.

[17] G. Weber and M. Specht. User modeling and adaptive navigation support in WWW-based tutoring systems. In Anthony Jameson, Cécile Paris, and Carlo Tasso, editors, *Proceedings of the 6th International Conference on User Modeling (UM-97)*, volume 383 of *CISM*, pages 289–300, Wien, June 02–05 1997. Springer.

[18] Extensible stylesheet language (xsl) specification. W3c working draft, W3C, 1999. Available at `http://www.w3.org/TR/WD-xsl`.