Towards an Intelligent Online Textbook for Discrete Mathematics

William Billingsley
Peter Robinson
Rainbow Group, Computer Laboratory
University of Cambridge
15 JJ Thompson Avenue
Cambridge CB3 0FD, United Kingdom
Email: William.Billingsley@cl.cam.ac.uk

***Abstract*— We have developed a web-based homework tutor for discrete mathematics that is a step of progress towards building an intelligent adaptive textbook. The student works on mathematical problems in a notation that is closely mapped to the notation the student would be expected to write on an exam paper. The tutor gives advice and feedback as the student is working, in a co-operative manner rather than submission-and-response. This feedback is linked into the topic structure of the intelligent book, allowing the student to query for content material relating to a piece of advice. More than one content item is available on any topic, allowing server rules to choose items the that are likely to be useful to the individual student, while still allowing the student to reject the tutor's selection and choose a different content item.**

I. Introduction

The Intelligent Book project is a joint project between the University of Cambridge and the Massachusetts Institute of Technology to develop online textbooks that:

1) automatically improve their own content
2) adapt their content to the individual student who is using them
3) have example problems that are not static presentations, but are tutorial questions that actively help the student to work through them
4) allow students to add material into the book and annotate existing material with their comments

The motivation for this project is the cost and difficulty of providing individual human tutoring to students. The University of Cambridge aims to provide small group tutorials, called supervisions, to its undergraduate students in each of their lecture courses. Typically two or three undergraduates meet with a faculty member or graduate student. For a 16 lecture course with 120 students, approximately 160 hours of supervisions will take place in total. This does not take into account preparation time or the time taken to mark students' homework. This is very labour intensive. A further issue is that many supervisors are graduate students with litte formal training in tutoring, so there is some variation in their teaching skills.

Various studies have shown that automated tutoring is an effective technique, at least compared to classroom teaching alone [6], [7], [8], and can sometimes be favourably compared to tutoring with inexpert human tutors [4]. We believe that providing automated individual tutoring, as a supplement to human tutoring or as an eventual replacement, would help reduce the need for very large numbers of supervisors and, by being a common resource available to all students, would help even out the quality of tutoring that each student receives. However, it also appears sensible to combine the automated tutoring with online content. If the tutor tells the student that he or she didn't take a particular concept into account, there is a reasonable chance the student will want to ask for more details on that concept, the reason why it matters, and how it fits in with surrounding content material. This leads us to the concept of an intelligent textbook where additional content material can be added into the student's textbook as he or she works through the example problems, and existing content can be substituted for explanations the student found more helpful. Some of these aims are shared by the Living Book project [1], which also aims to build an adaptive textbook with interactive exercises, but takes a different approach to the both the exercises and the content.

In this paper we describe the intelligent homework system we have developed, which is a step of progress towards an intelligent book for discrete mathematics.

II. Maths Tiles

One feature of intelligent books is that they should be capable of asking many different questions on the full range of topics that you would expect to find in a textbook. For this reason, the intelligent books we are developing have a very general structure which allows different content clients and diagram domains to be included in different questions [2]. However, for the discrete mathematics book, a particular focus is a mechanism for the student to interact with mathematical expressions and proofs. Because it is a teaching interface, a number of characteristics of this mechanism become particularly important. These are described here using the Cognitive Dimensions of Notations framework [3]:

1) Closeness of mapping. The notation used by the students to enter and manipulate maths should look like the maths they are expected to write on the exam paper. If the maths entry notation is very different, for example a

0-7803-9035-0/05/$20.00 ©2005 IEEE

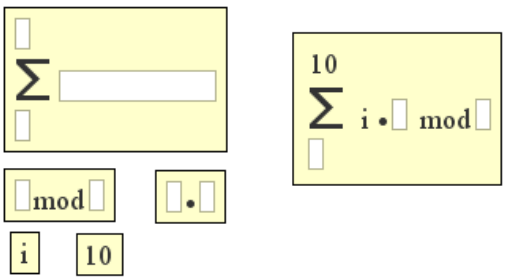

Fig. 1. Some maths tiles, loose and combined.

text-based formula language, then this adds a learning burden which is not directly related to the material being taught.

2) Low viscosity. We expect students to enter incorrect expressions and proofs a lot of the time, otherwise they would not need tutoring. So, it is important that students can make changes to their expressions very easily.
3) Low premature commitment. We do not want the interface to force the student to answer the question in a particular order. While there are occasions where we do want to teach students to use a particular methodology, this should be enforced in the explicit teaching feedback, rather than as an implicit by-product of the interface design.
4) Low provisionality. It needs to be possible for students to sketch out and play around with fragments of answers without being committed to them. So, for example, if a question expects a single expression as the answer, the interface should still allow students to construct as many answer fragments as they like in parallel just to see what they look like.
5) Progressive evaluation. Sometimes, students will know what part of the proof or expression needs to look like, but get stuck on how to complete the structure. They need to be able to ask for feedback from the tutor on an incomplete answer fragment.

We have developed an interface called MathsTiles to meet these criteria. It is based around graphical tiles containing arbitrary pieces of mathematics. Students can drag these tiles and drop them into sockets in other tiles to build up expressions and proofs. In this way, the notation is kept closely mapped to handwritten mathematics, but the students are made aware of the structure of the maths they are building. Some example tiles are shown in Figure 1.

The tiles are defined by XML which can be kept in tile library documents. This makes it easy for a content author to define different tiles for different questions. Tiles can contain text, diagrammatic shapes, and sockets, and they can be laid out either in a simple horizontal alignment along a text baseline or in a two-dimensional layout defined by a sequence of alignment steps in the tile's XML definition.

Tiles can be put anywhere on a canvas, allowing students to sketch out part-answers freely and to work equally well in a top-down or bottom-up manner. They can be pulled out of and dropped into sockets by holding control when pressing or releasing the mouse over the tile or socket, so the effort required to change a structure is low. When a student drops a tile into a socket in another tile, the border of the contained tile is removed so that the appearance of the constructed maths is not interrupted. However, the tile border reappears when the mouse is moved over the tile, highlighting the structure of the tile. This can show cases where different structures may appear similar but be mathematically quite different, as shown in Figure 2. In practice, it would be better to use brackets to distinguish these, and there is a mechanism for forcing students to do so by defining priority tags on the tiles and sockets.

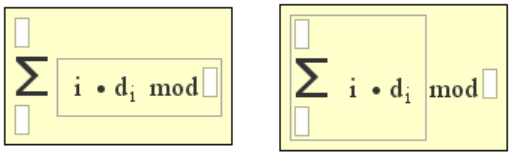

Fig. 2. The two structures looks similar in written notation, but their structure is different. This is apparent when the tile borders are highlighted.

One particular use for definable tiles is to teach students to think more carefully about what they need to do to answer an exam question fully. For example, what is needed to complete an induction proof, or how to show some property of a set relation. Tiles can be defined that include sockets for each step that the student is expected to include. If students omit a step, it is obvious because there is an empty socket in the tile; if they only partially complete a step, this is also obvious because there are empty sockets inside the tiles they have put into the main socket. Figure 3 shows a tile for an induction proof. Filling in the proofs for the base case and inductive step involves using tiles for different kinds of proof, each of which has its requirements laid out as a set of sockets. To support working with proofs, MathsTiles includes numbered statement tiles, variable tiles, and expanding socket lists for lists of assumptions and proofs that involve a sequence of steps.

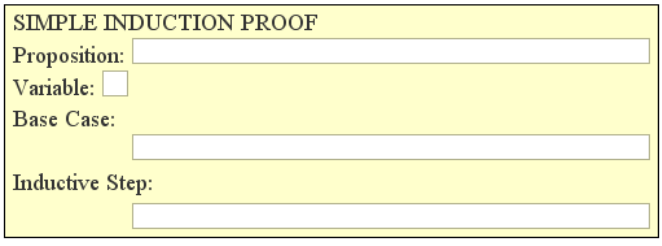

Fig. 3. Tiles can be defined as "rubber stamps" for proofs, relations, and other mathematical concepts, showing the student exactly what they are expected to complete.

Typically, a question includes some fixed tiles into which the student is expected to drop the answer. In a simple question,

this may be a single tile with the text “Answer” and a socket for the assembled answer. For a more complex question, it may be a partially built proof that students are expected to complete. For early questions introducing the interface, the tiles needed to build an answer start scattered on the canvas. For later questions, the student can add tiles to the canvas using a panel of buttons that are defined by XML in the question. An example canvas is shown in Figure 4.

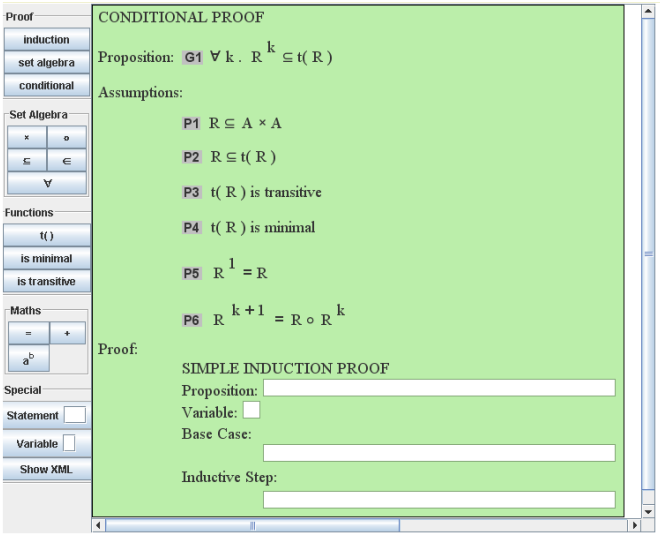

Fig. 4. A demonstration example showing the features used in a complex question. The student is given part of a proof to complete. The tile selection panel is also configured from XML in the question.

Some questions limit the number of tiles of a particular type that students are allowed to use in constructing the answer. This is a simple way of reducing the set of possible correct answers the student might come up with, and can make the server-side teaching script simpler to write.

III. Client Organisation

The client architecture of the intelligent book has been described before in [2], but it is worth recapping it here. A diagram of the architecture is shown in Figure 5 and an example screenshot in Figure 6. The contents of an intelligent

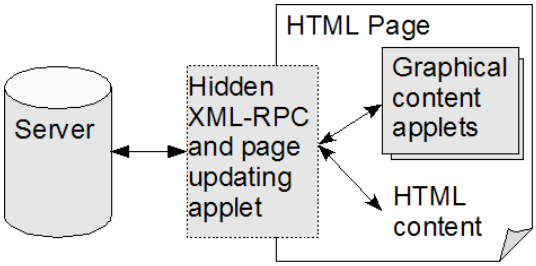

Fig. 5. The client architecture of a question in an intelligent textbook. Text, HTML and forms advice from the tutor is shown dynamically in the page. Graphical editable content is shown in Java content applets. A hidden applet communicates with the server over XML-RPC, interfaces to the change API of the content applet, and updates the page’s HTML.

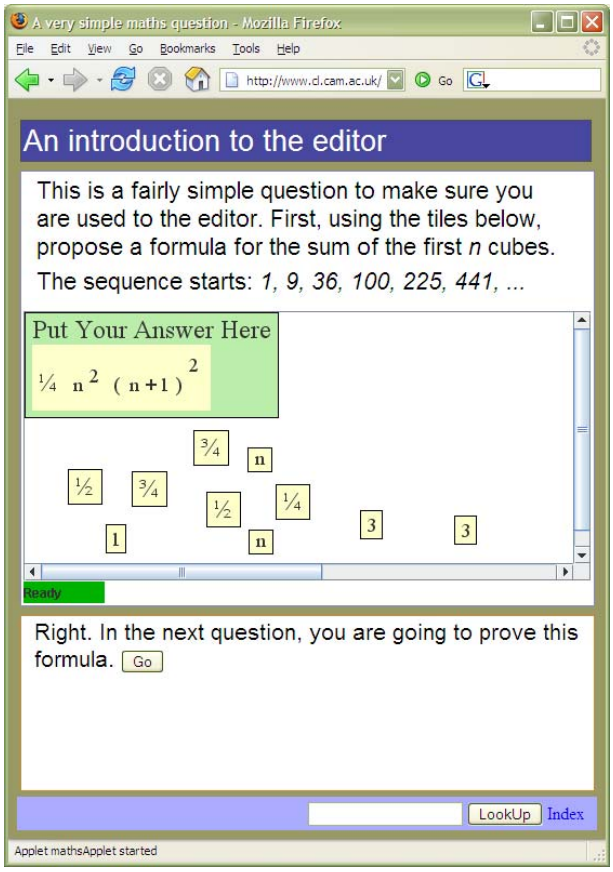

Fig. 6. A web page containing a question using MathsTiles.

book, including the questions, are shown in web pages. To support interactive graphical process questions, these pages can contain graphical content applets, such as MathsTiles. These content applets can be thought of as a way for the student to work diagrammatically on an XML document that contains the current state of the problem. Changes the student makes to the diagram are reported to the server as changes to the XML document. The content applet also publishes an XML change API, so that changes sent from the server or from JavaScript controls in the webpage, can be applied to the document in the content applet. The student’s progress through the tutorial is characterised, then, by a stateful conversation of document changes between the client and the server.

Comments and advice the server wishes to make to the student are displayed in HTML on the web page. This HTML can include forms controls and JavaScript. A small applet handles communication with the server over XML-RPC, and performing dynamic updates to the HTML of the web page.

The change messages are sent to the server in a separate thread from the interface. This prevents the interface from appearing to lock up while the tutor communicates with the server. The student is shown that communication is happening by a status indicator on the page, although this is kept unobtrusive. Only changes which are considered significant are sent to the server. So, for example, dragging a tile across

the canvas is not communicated to the server, but dropping the tile into a socket is.

The tutor generally does not make structural changes to the student's work directly. Doing so might be jarring to the student - a sudden unexpected and possibly unwanted change being made - and it could also cause concurrency problems. The student can continue working on the diagram while the tutor is preparing a response, and if there is a very long latency, it would be possible for the tutor's change to no longer be meaningful in the student's copy of the work, but to have already been applied to the tutor's copy. Instead, the tutor tends to highlight parts of the student's work, and to provide an explanation in the HTML on the page. This HTML can contain JavaScript-backed controls which the student can use to apply the tutor's intended change at the client. A trivial example of this is shown in Figure 7. It is still possible that the advice will no longer be meaningful, but this should be apparent to the student, and in any case the change has not yet been applied on the server, so the student's and tutor's documents will remain synchronised.

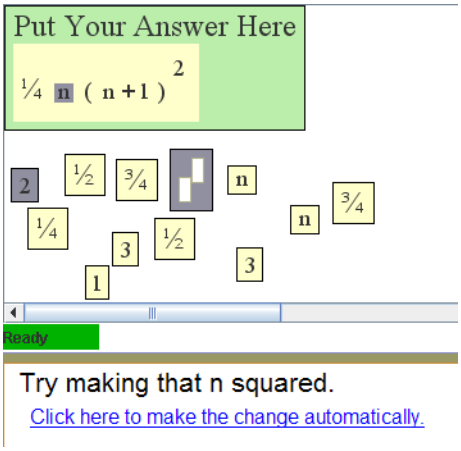

Fig. 7. The tutor generally does not make structural changes to the student's work directly. Instead, it tends to highlight parts of the student's work and provide a comment in HTML on the page. This HTML can contain JavaScript-backed controls which the student can use to apply the tutor's intended change at the client.

IV. Teaching Scripts

In automating tutoring, two successful methods for analysing students' work have been Model Tracing [5] and Constraint Based Tutoring [7]. Model Tracing focuses on strategic feedback, identifying common procedural mistakes that students make, and comparing the students' working to a set of production rules that represent an ideal strategy. Constraint Based Tutors examine the errors students introduce into the problem, and assess the students' understanding of the rules that determine whether the current problem state is valid. Traditionally, a textbook should teach both the rules of the subject matter and strategies for solving problems, so the intelligent book needs to support both these analysis techniques. Practically, it can also be useful to follow a mixed approach. For example, a question might not contain a full set of Model Tracing rules but only a few production rules to warn the student away from a the most common procedural mistakes.

In earlier work, we developed a tutorial client for electronic circuits [2] where the XML changes sent from the client were not actually applied to a document on the server, but translated into settings to make in a constraint propagation system in the Scheme language. For the discrete mathematics system, however, the XML changes are applied to a document kept in an XML database. This gives us three steps to processing a student action, as shown in Figure 8.

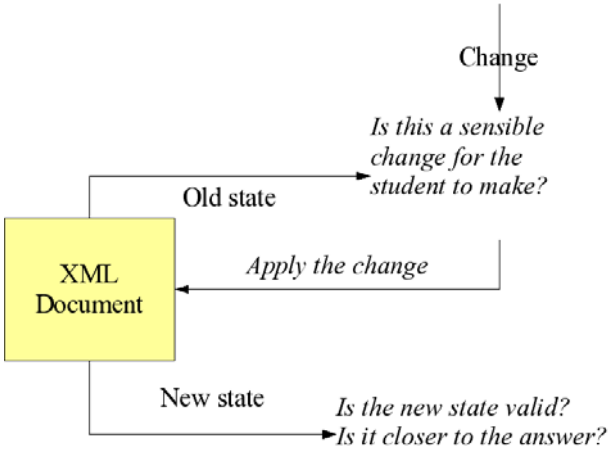

Fig. 8. Processing a student's action

In the first step, the teaching script looks at the change the student is making and the current state of the student's work, and interprets what the student is trying to do. This is the process analysing step that can be used to implement Model Tracing or just to identify common tactical errors. In the second step, the student's change is applied to the document in the database, giving a new current state of the problem. In the third step, the new state is examined to see if it is a correct state, or if the student has broken any rules or criteria from the question.

Because the teaching strategy can vary from question to question, the scripts take a loose approach to student modelling. Rules that are written into the scripts record annotations against the students in the database. These can be marking a student's understanding of a particular rule, but they can also be indicate the approach the students tends to use, or track floundering students by watching the number of changes they make before they make any progress towards a solution.

Helpers

The intelligent book clients are very open in what they permit the student to do, so if the teaching script had to cater for every possible student action, it could become unmanageably large. Instead, the system is designed so that scripts for complex questions can use external helpers to perform most of the checking. The teaching script itself does not fully examine the problem state, but looks at the results that come back

fom these helpers instead. We have tested this using simple XQueries as helpers, but the intention is to interface to third party AI systems such as theorem provers as well. Calls out to helpers can happen at both the process analysing step and the state checking step. The returned data is stored in the database so that when the teaching script is called for the next change made by the student, the annotations made by the helper from the previous change can be used in the process analysing step.

Writing a teaching script is a matter of filling in six slots, any of which can be left empty:

1) Any initialisation needed by this script.
2) Call any AI helpers based on the current state and the student's change.
3) Generate any advice or response based on the current state and the student's change.
4) Call any AI helpers based on the new state.
5) Generate any advice or response based on the new state.
6) Any clean-up needed by this script.

Additionally, teaching scripts can include other public functions to expose to the client as XML-RPC methods, and functions that are to be exposed to the adaptive advice function system described in Section V. The public functions can be used to provide question-specific help or to let the teaching script dynamically ask the student forms-based questions as hints or to make the students explain their working. Hyperlinks and forms are shown on the question's HTML page, and backed with a line of JavaScript that calls the XML-RPC in the teaching script via a forwarding method in the tutorial client applet.

Teaching scripts can be written in Java, or compatible languages such as Jython and Groovy, and are implemented as subclasses of an abstract teaching servlet superclass.

V. Advice Topics and the Content Model

As described in the introduction, if the tutor tells a student they have failed to take a concept into account, then the student is likely to want to know more about that concept. When the teaching script sends comments and advice to the client, it can also ask the client to show topic links. These link into the intelligent textbook's content model and request the most suitable entry matching a topic string and, optionally, an explanation type such as "summary" or "example". Where topic links appear depends on the layout and organisation of the HTML page, but an example is shown in Figure 9.

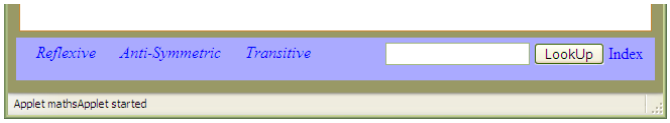

Fig. 9. Topic links appear that are relevant to the advice being given. The student can also type a topic into the box.

The tutor often has more than one piece of content covering a topic, so rules on the server choose which one to deliver to the student. This allows the textbook to adapt to the individual student. Each content entry is a URI. Too keep a consistency of look within an intelligent book, URIs should normally point to internal content, but external materials can be included. Additionally, there is a "last resort" entry which runs a search on a popular search engine. Content entries can also point to pages of a wiki, allowing the students to edit and improve the explanation. To guard against poor edits, the original version of the page should also be included as a topic entry. We are also investigating using systems that allow students to annotate contents page with pop-up comments and links, but not edit the text directly.

The server uses metadata on each content entry to choose which one to show to a given student. At the moment, the rules which do the choosing are delegated to a set of XQuery scripts. Some experimentation is still needed to determine good algorithms for choosing appropriate entries for individual students, but generally the scripts work by preferring content this student has seen and not rejected before, and then preferring content that has been recommended by students with similar metadata to this student.

The content entries always pop up in a new window that is divided into two frames. The main frame shows the content itself, and a smaller header frame contains a number of feedback buttons and links. A screenshot of this header frame is shown in Figure 10. In the top left corner, the frame shows

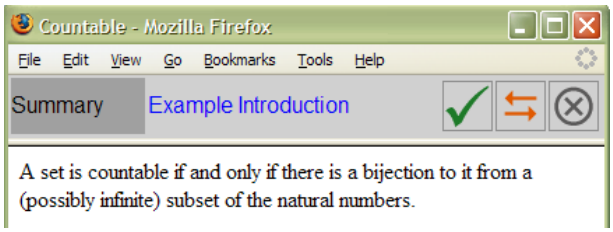

Fig. 10. A header frame is attached to content explanations when they are shown. This shows the type of explanation being shown, and links to request other types of explanation for this topic. The buttons on the right allow the student to recommend this explanation, ask to replace it with a better entry from the database, or just close it.

the type of explanation that is currently being shown. Next to this, it shows links for all the other types of explanation the database has for this topic. This makes it easy, for instance, for a student to read a summary and then ask for an example on the same topic. When clicked, these links call the content selection scripts on the server for the same topic string and the specified explanation type, and also pass in the URI of the currently displayed content entry so that entries that are marked as being related to it can be preferred. On the right hand side of the header frame are a number of feedback buttons, each of which can be used to close the window. From left to right, these are:

- "Recommend" - closes this entry, and records that this student recommended it. If the student has already recommended this content item, this is replaced by an "unrecommend" button.
- "Replace" - closes this entry and brings up a list of the entries for this topic so that the student can choose an alternative. This also records that this student declined

this entry.

- "Close" - just closes this entry.

Over time, the recommendations should cause better content to be offered more often, and explanations that students do not understand to remain idle. This could then be used by a content author to replace poor explanations, and improve the textbook from year to year. Recording the fact that a student declined or recommended a particular entry also allows the choice scripts to increase or reduce the likelihood that they will offer similarly marked content to this student in the future.

Advice functions in the teaching script can also be made adaptive. This works by keeping a structure in the script linking the function to a relevance function that determines when it should be considered, and an advice topic key. Metadata on each function is stored in a small XML document in the database. When the student requests advice matching the advice topic key, the selection script chooses from the relevant functions based on the metadata they have built up, and again gives the student the option of accepting or rejecting the advice. The purpose of this is for strategic advice that is not a key part of the question itself, but that may be useful to some students.

Potentially, we would also like to allow students who have completed the question to leave hints for students in subsequent years, although there is currently no mechanism for a student to enter these hints.

VI. Conclusions

The intelligent textbook for discrete mathematics is a work in progress, and at the time of writing the question scripts and content are still being written. The system is being prepared to be deployed to students in Cambridge in October 2005.

Future work for the book includes working with students and supervisors to see how the use of an intelligent book as a homework system affects student learning and also how it affects the human-tutored supervisions. On the one hand, the system may reduce the workload on the supervision system by reducing the time needed to explain homework corrections, but on the other hand it may remove a tool supervisors use for getting to know the student's abilities.

We are also looking to give the system a more book-like structure in parallel to the topic queries. While the student should be able to easily access content by topic and by advice given during example questions, ideally the student would also be able to "open the book to page one" and be shown the material in a pedagogically useful order.

Acknowledgements

The authors acknowledge the advice and assistance of their project partners Hal Abelson, Gerald Sussman and Chris Hanson from the Massachusetts Institute of Technology, and of Mark Ashdown and Kasim Rehman from the University of Cambridge. The project is funded by the Cambridge MIT Institute.

References

[1] P. Baumgartner, M. Gross-Hardt, and A. B. Simon. Living book - an interactive and personalized book. In V. Milutinovic, editor, *SSGRR 2002s - International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*, 2002.

[2] W. Billingsley, P. Robinson, M. Ashdown, and C. Hanson. Intelligent tutoring and supervised problem solving in the browser. In *Proceedings of the IADIS International Conference WWW/Internet 2004, Madrid, Spain*, pages 806 – 811, 2004.

[3] A. Blackwell and T. Green. Notational systems - the cognitive dimensions of notations framework. In J. M. Carroll, editor, *HCI Models, Theories and Frameworks*, pages 103 – 133, Amsterdam, 2003. Morgan Kaufmann.

[4] A. Corbett. Cognitive computer tutors: Solving the two- sigma problem. In *User Modeling 2001: 8th International Conference, UM 2001, Sonthofen, Germany*, pages 137 – 147, 2001.

[5] 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*, 8:30 – 43, 1997.

[6] C. C. Kulik and J. A. Kulik. Effectiveness of computer based instruction: An updated analysis. *Computers in Human Behaviour*, 7:75 – 94, 1991.

[7] A. Mitrovic, M. Mayo, P. Suraweera, and B. Martin. Constraint-Based Tutors: A success story. In *Engineering of Intelligent Systems : 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2001, Budapest, Hungary*, 2001.

[8] R. Shelby, K. Schulze, D. Treacy, M. Wintersgill, K. VanLehn, and A. Weinstein. An assessment of the Andes tutor. In *Proceedings of the Physics Education Research Conference, July 21-25, Rochester, NY*, 2001.