

Project Description

MathQuery: Improving Computer-Based Mathematics Assessment Using XML

Topic B2. Teaching and Learning: Tools for Curriculum Development and Learning Assessment

Part 1: Identification and Significance of the Innovation

This proposed SBIR Phase I project seeks to determine the feasibility of creating a radically new method for analyzing and evaluating free-form student responses within computer-based assessment and tutorial systems for mathematics. Dubbed MathQuery, this method will allow a computer-based system to compare the *structure* of a student's response to a problem against a rule set that characterizes how the problem can be solved, thereby identifying precisely how a student formulated the response and what errors may be present.

MathQuery will be an assessment-oriented query language implemented as an eXtensible Markup Language (XML) application. Specifically, MathQuery will combine elements of the following:

- the Mathematical Markup Language (MathML), the World Wide Web Consortium's (W3C) XML specification for mathematics markup [1];
- eXtensible Stylesheet Language transformations (XSLT), the W3C's specification for conditional formatting of XML documents; and
- the XML Path Language (XPath), the W3C's specification for identifying "nodes" or elements within XML documents and the relationships between those elements (typically processed by an XSLT engine) [2].

MathQuery will implement processing instructions and reporting functions in accordance with the IMS Global Learning Consortium's Question and Test Interoperability (QTI) specification [3]. A primary goal of the IMS is "to facilitate working relationships among LMS [learning management system] vendors," and the QTI specification is specifically designed to provide common application program interfaces (APIs) that simplify the integration of LMS extensions [4].

We do not anticipate that MathQuery will be a stand-alone application; rather, it will be an extension that can be incorporated into existing LMS systems—particularly assessment and tutorial systems—and that provides much more robust evaluation of free-form mathematical input than is currently available. Conformance with QTI will streamline integration of MathQuery into existing LMS systems. The design of MathQuery as an XSLT application means its functionality can be implemented using any of the readily available XPath-capable XSLT engines, such as Saxon and Xalan.

Current methods for computer-based mathematics assessment rely heavily on multiple choice or fill-in-the-blank questions, which provide little opportunity to analyze the *process* used by a student to work a problem. Most assessment systems that support free-form response require that a student use a specialized syntax rather than traditional mathematical notation, which makes those systems inappropriate for assessment across a broad range of mathematics disciplines and skill levels. MathQuery will support free-form responses through the use of highly customizable MathML-enabled equation editors, which support traditional mathematical notation while retaining the semantic or computational meaning of those responses. Because student responses will be highly structured XML documents, the structure of those responses can be analyzed in detail using XPath with little or no ambiguity in meaning.

MathQuery's ability to analyze the structure of free-form responses will provide deeper insight into a student's math skills, which in turn can be used to formulate a more precise assessment or to adapt instruction more effectively for a student's individual needs.

Part 2: Background and Phase I Technical Objectives

Computer-based learning and assessment is gaining increased acceptance in both K-12 and college education because of its promises of lower cost, easy access, flexibility, and instant feedback. In college-level mathematics, many introductory textbooks are now accompanied by online diagnostic and homework systems, giving students round-the-clock practice and feedback. At the K-12 level, the recent “No Child Left Behind” (NCLB) legislation is expected to boost the use of computer-based assessment because states must make rapid changes in curriculum in response to mandatory annual assessments [5, 6]. Traditional pencil and paper tests cannot be graded quickly enough to make those changes.

Despite the increased use of computer-based systems, there is still widespread dissatisfaction with their implementation and performance. Much criticism is aimed at their overreliance on multiple-choice questions, which can be implemented easily and reliably but yield only a shallow assessment of a student’s understanding of the tested material. Systems that allow richer forms of response are typically criticized for their poor performance and user interfaces [7, 8].

Criticism of multiple-choice testing is directed as much against standardized testing practices in general as against computer-based testing. Standardized tests are often designed to measure a school system’s accountability rather than individual student performance. Public schools are under tremendous pressure to prove their worth by producing high standardized test scores; as a result, there is widespread belief that educational systems shift their instructional emphasis toward test-taking skills rather than problem-solving skills. The 1996 Title I amendments to the Elementary and Secondary Education Act (ESEA) acknowledged that standardized testing had become “a primary tool for implementing program requirements” [9] rather than a means of improving education. The 1996 amendments stated explicitly:

What has been learned since 1988 [is that] use of low-level tests that are not aligned with schools’ curricula fails to provide adequate information about what children know and can do and encourages curricula and instruction that focus on the low-level skills measured by such tests. [10]

A follow-up report issued by the U.S. Department of Education emphasized that “performance-based assessment” and the use of “open-ended or constructed response items” gave students the opportunity to demonstrate their knowledge and problem-solving skills in a more meaningful manner than the traditional standardized tests [9]. This preference for performance-based assessment has been the impetus for using writing samples to evaluate reading comprehension and language skills [11, 12], and similar arguments have been made for performance-based assessment in mathematics [13, 14].

Computer-based systems that allow students to “show their work” are currently limited to computer-algebra systems and “intelligent” or “cognitive” tutoring systems. Computer-algebra systems are useful environments for giving students free reign in their exploration of mathematics; however, they provide inadequate mechanisms for defining the scope or goals of an exercise, which are necessary for both assessment and guided instruction (see, for example, [14]).

Several intelligent tutoring systems have demonstrated that the rules and processes needed to solve mathematics problems can be modeled using computers even when the systems have a pedagogical rather than computational focus [15, 14, 16]. These systems share several unique characteristics: they establish rules that describe a subject's knowledge base, they model the problem-solving process using those rules, and they require that a student demonstrate understanding of the solution process by applying the appropriate rules. These systems assess the student's choice of rules to determine appropriate feedback. In general, however, these systems have two drawbacks that hinder commercialization: content creation is extremely time consuming, and user interfaces do not capture student input in a natural way—requiring, for example, the use of computer-algebra syntax [16] or a series of palettes that are often overwhelming [14]. Substantial work has been done on using natural language in intelligent tutoring [17], but this work has not proven to be scalable nor has it been applied in any significant measure to mathematics.

The goal of this Phase I research is to determine whether existing XML tools can overcome many of the obstacles to implementing performance-based assessment. The technical issue is not whether performance-based assessment can be programmed; this has already been demonstrated. Rather, the issue is whether performance-based assessment can be programmed much more *efficiently* using XML techniques rather than more conventional programming languages. A MathML editing environment can bridge the gap between the semantic meaning of a mathematical expression and its corresponding graphical representation. In this respect MathML may offer something close to a natural-language interface for sustaining a dialog between students and computers. Given the potential of MathML, we hope to show that additional XML tools be used to derive information from MathML expressions that is sufficient to yield useful assessment of a student's problem-solving skills.

Further research will be required in Phase II to determine the commercial feasibility of MathQuery. Four issues will be critical to commercial acceptance:

1. A user syntax or graphical user interface must be developed that allows easy authoring of assessment questions, rules, and queries, including the logic and branching that are necessary to construct sophisticated analysis on multistep problems. XML is much too verbose to author by hand, so a usable authoring interface will be necessary to make MathQuery a scalable solution to assessment.
2. Significant improvements must be made in MathML equation editors. Several MathML equation editors are currently under development—including one by Integre—but at present none offer the combination of server delivery, simultaneous support for both Presentation and Content MathML, easy customization of palettes and notation, and ease of use. MathQuery will be subject to the same scrutiny as all other computer-based assessment applications; the student's interface to free-form response *must* be intuitive and work flawlessly.
3. MathQuery must be implemented in conformance with the IMS Global Learning Consortium's Question and Test Interoperability specification, and it must support (insofar as necessary) the emerging metadata standards produced by the Dublin Core Initiative and the IEEE's Learning Technology Standards Committee. These

qualities will allow MathQuery to “play nicely” in the realm of computer-based learning and assessment.

4. Much more substantial testing will be necessary to demonstrate that MathQuery can analyze free-form response accurately and reliably. Classroom testing will be necessary to validate the student’s use of the system and the utility of MathQuery’s reporting and feedback mechanisms. This can be achieved by partnering with a school or university that is already using online assessment or intelligent tutoring.

These issues are far beyond the scope of Phase I research, but require that the technical feasibility first be demonstrated. The success of this proposed Phase I research will create a solid foundation on which to pursue Phase II development and subsequent commercialization.

Part 3: Phase I Research Plan

The proposed research seeks to determine the feasibility of a query language that can be used to assess a student’s understanding of mathematics more thoroughly and at a deeper level than is possible with current methodologies. The premise behind MathQuery is that computer-based systems can discern key characteristics of free-form mathematical response by using several powerful XML tools that are approaching maturity: MathML, XSLT, and XPath.

MathML includes both Presentation and Content representations. Presentation MathML describes the visual layout of mathematical expressions, and can be customized to match the notation used in a particular subdiscipline or skill level. Content MathML describes the semantic meaning of an expression, is independent of the visual layout, and can be evaluated unambiguously by a MathML-capable computation engine, such as Maple or Mathematica. Because MathML applications need to support both representations, MathML equation editors can provide an interface in which students can enter mathematical expressions using the notation they have learned in class, while simultaneously providing the system with a semantic expression that can be analyzed or computed with high reliability (see Figure 1).

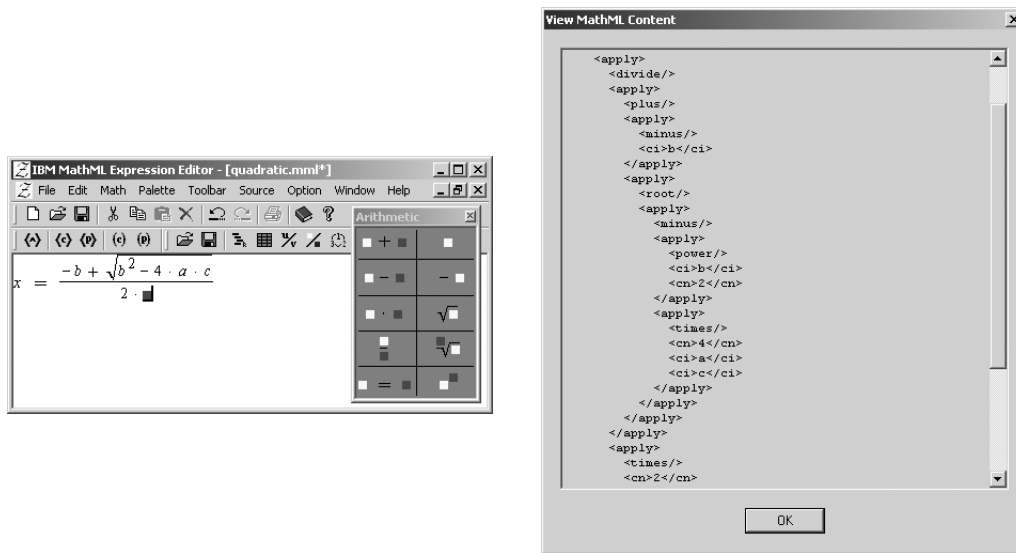


Figure 1. *Left:* The quadratic equation being entered into the MathML Expression Editor developed by IBM Research and being acquired by Integre. The visual layout and editing palettes can be customized to match the notation of a specific subdiscipline independently of the semantic meaning. *Right:* The Content MathML produced by the input.

XPath is an XML technology that is used to locate elements within an XML document. Typically an XPath instruction is used to search a document for content that meets specific criteria described in the instruction. These criteria could involve *node types*, the string content of particular nodes, or the structural or *axis* relationship between particular nodes or node types. The requested content is returned by XPath for further processing, or XPath can simply report whether a document contains content that meets the specified criteria.

In the case of MathQuery, the XML documents would consist of Content MathML expressions provided via an equation editor. As a simple example, consider the expression $\frac{1}{2}(a + b)$. The Content MathML for this expression is

```
<math>
  <apply>
    <times/>
    <apply>
      <divide/>
      <cn>1</cn>
      <cn>2</cn>
    </apply>
    <plus/>
    <ci>a</ci>
    <ci>b</ci>
  </apply>
</math>
```

MathML, like all of XML, is both highly structured and extremely verbose; yet these characteristics are what make XML so powerful. Notice that the purpose of and relationship between each element of the expression is clearly defined by a node type. Numeric values are identified by `cn` tags, whereas variables are identified using `ci` tags. All mathematical operations—including the *implicit* multiplication by $\frac{1}{2}$ —are identified *explicitly* using `apply` tags. Nothing in the Content MathML describes how the expression should be *formatted*; this is handled automatically using an XSL stylesheet or specified explicitly in the Presentation form of the expression.

XPath instructions could be constructed to answer questions about the structure of this expression. For example:

- Does the expression use variables? If so, how many are there and what are they? In this case XPath would search the expression for all `ci` nodes, and report that *a* and *b* are used.
- Are *a* and *b* being added together? Here XPath would search for an `apply` node that has three *child* nodes consisting of a `plus` node followed by two nodes (or their *descendant* nodes) that include `ci` nodes containing the strings ‘a’ and ‘b’.

The simple XPath instruction

```
//apply[plus]/child::ci
```

asks the XSLT or XPath engine to return all variables that are being added together. As shown in Figure 2, XPath identifies in this expression the two `ci` nodes containing *a* and *b*.

This simple example demonstrates how XPath could in theory serve as the mechanism by which a learning or assessment system accesses free-form mathematical responses. In practice, the analysis we would want to perform on an expression would be much more complex, as would the XPath instruction that queried the expression.

by conforming XSLT or XPath engines—and for establishing MathQuery as a properly defined XML application or QTI extension. We expect that MathQuery will be a “thin” specification, meaning that it will incorporate elements of MathML, XPath, and QTI by reference only. Very little information in the MathQuery specification will be unique to MathQuery, which makes writing the specification and schema plausible within the scope of the Phase I research. One potential obstacle to writing a successful MathQuery schema is that the current XPath specification is not very compatible with the current XML Schema specification (see, for example, [18, p. 115]). Our hope, however, is that simple reference to XPath in the MathQuery schema will avoid any inherent incompatibilities, and that the next scheduled release of the XPath specification will resolve the current discrepancies.

Once the specification and schema have been written, we will construct a reference implementation of the MathQuery specification. This implementation will consist of two elements:

- a preliminary mock-up of a user-interface syntax or graphical user interface for constructing rules and queries; and
- a Java servlet that coordinates problem delivery, student input, and feedback reporting, including communication with XSLT, computation, and computational equivalence engines.

To handle basic file management and reporting functions, this reference implementation will be built inside of the EDU assessment platform from Brownstone Research (www.brownstone.net), configured to include the XSLT and computation processors provided via MapleNet from Waterloo Maple (www.maplesoft.com). Integre has long-standing business relationships with these companies, and EDU’s programming interfaces are sufficiently exposed to allow integration of specialized assessment functions in the form of applets or servlets.

Upon completion of the prototype, we will simulate a real-world assessment situation by working each problem of the test suite within the system to evaluate the success of MathQuery. Specifically, we will determine whether the prototype accurately distinguishes proper application of rules from improper application. We do not expect that the MathQuery prototype will successfully implement all rules and queries that we would like to ask of a particular problem response, but we will target a success rate on the order of 50%. Unsuccessful queries will be evaluated to determine whether failure was a result of insufficient information in the implemented model, improper use of XPath and other mechanisms, weaknesses in the current XPath specification, or the inherent inability of this methodology to address mathematics assessment.

Part 4: Company Information

Integre Technical Publishing was founded in 1991 as a publishing services company specializing in the production of mathematically-intensive textbooks and professional references. As web technologies and access evolved during the mid-1990s, Integre added web and media development to its list of services.

Although we were excited by the possibilities presented by web publishing, we were constantly frustrated by the lack of web support for mathematical notation. In 1999 Integre began a collaboration with IBM Research, which was developing the techexplorer browser plug-in as a means of delivering web-based mathematics. In 2000 Integre was named an IBM techexplorer “Ambassador” company, and formally began a joint marketing effort with IBM to promote techexplorer. Because of poor economic conditions and the small size of the mathematics market, IBM discontinued techexplorer development in 2002. Integre is now acquiring the techexplorer technology from IBM and plans to develop a suite of MathML applications to support print and web publishing, computer-based mathematics, intelligent tutoring, distance learning, and web-based collaboration.

MathQuery will be a technology that complements Integre’s anticipated software offerings for web-based mathematics. We foresee the day when “instructional learning materials” offer continuous assessment, feedback, and instruction that adapts to the strengths and weaknesses of individual students. To make this vision a reality, we need more than rich content; we need mechanisms for interacting with that content and for creating a dialog between humans and computers.

Integre currently has nine full-time employees in composition and editorial (4), programming and developmental editing (3), and administrative (2) functions. In addition, Integre presently uses the services of nine additional part-time employees and contractors for production and developmental editing. Integre’s gross revenue has grown incrementally over the last several years, reaching \$440,000 in 2002. Integre’s anticipated revenue for 2003 is approximately \$700,000. We expect this revenue growth rate will continue in 2004 and beyond as we refine and begin delivery of our MathML technologies.

Part 5: Commercial Potential

The idea behind MathQuery evolved out of numerous requests from Integre clients—mostly educational publishers targeting the college market—for technologies that accept free-form student response and return context-sensitive hints, directed feedback, or partial credit within online homework and testing systems. Further motivation came from Integre’s efforts to implement web-based environments that allowed students to work mathematics problems entirely online using any valid solution path (see the sample “Show Me the Steps” applet in Figure 3). In both situations Integre has been moderately successful either in delivering the technology or in demonstrating its feasibility. Unfortunately, these products have required extensive programming efforts to implement the necessary features. Making these features available in a wide range of educational materials requires a different approach that streamlines the content creation.

As a technology component for mathematics assessment alone, MathQuery probably will not attract equity investors. However, assessment technology that provides directed feedback is in high demand among both assessment companies and educational publishers, and investment from these organizations as a “mission-critical vertical application” is a strong possibility, especially when combined with Integre’s other technology assets. We have already had discussions with three assessment-related

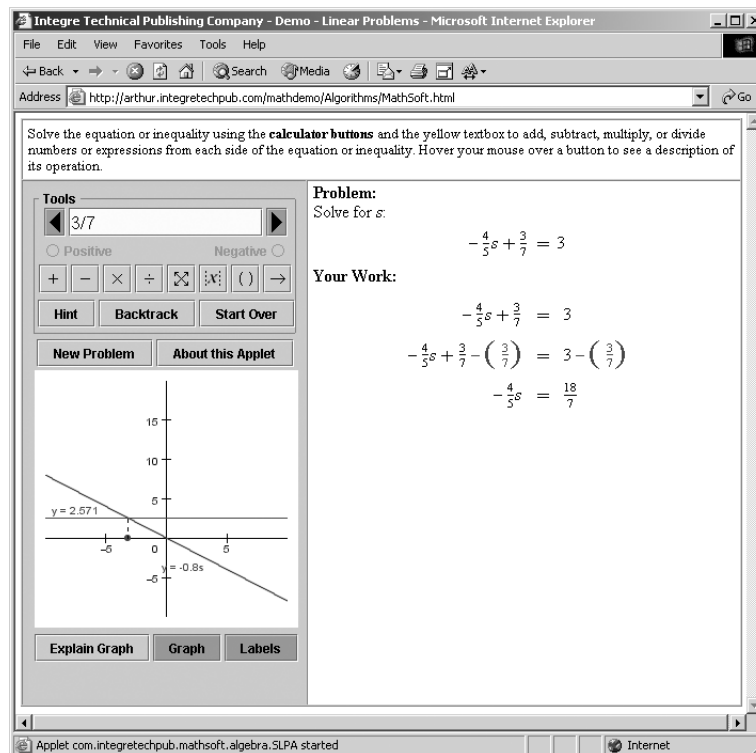


Figure 3. An example of the “Show Me the Steps” Java applets developed by Integre for MathSoft’s *StudyWorks!* self-tutoring learning package. These applets gave students a calculator-like interface for working problems online using any valid solution path.

companies that have expressed interest in the MathQuery technology (via license or acquisition), and we have identified at least a half-dozen other major players in the assessment market—each with over \$50 million in annual sales—that are likely adopters.

In the highly competitive college textbook market, innovative technology solutions are perceived as a significant competitive edge that yields higher textbook sales. This is especially true if the technology is closely aligned with textbook content and integrates effortlessly into the popular learning management systems, such as Blackboard and WebCT. Because the primary motivation for MathQuery derived from this competitive market, we expect the technology will gain rapid acceptance in the college market if we can demonstrate its feasibility.

And because directed feedback We have already begun discussions with several publishers and assessment publishers

The methods used by MathQuery are by no means applicable only to college-level mathematics, and we will leverage our presence in the college market and our marketing experience with IBM to segue into the K-12 market. If we can develop simple and reliable student interfaces, we can expect gradual acceptance of MathQuery in the K-12 market. Acceptance of technology solutions in the K-12 market is increasing, but at a much slower rate than in the college market. Unlike the college market, the K-12 market is much less experimental and is thus much more deliberate in their use of technology. Instructors typically have little or no say in textbook or technology adoptions, students are much less sophisticated, and the schools are held much more accountable for student performance. Under these conditions there is little market tolerance for unproven solutions.

Two factors will give MathQuery an advantage in the market. First, we are positioning MathQuery as an *extension* to existing systems rather than as a stand-alone application. The primary reason for this decision is that stand-alone assessment systems need substantially more infrastructure—file management, course management, and gradebook functionality, not to mention sales, marketing, and technical support requirements. Development of this infrastructure would impede our progress to market. A secondary reason is that we are positioning our other software products as technology components. Our primary competition with respect to mathematics rendering and editing software is Design Science (www.dessci.com), makers of the MathType equation editor for Microsoft Word and the WebEQ MathML equation editor. Design Science has positioned their products as off-the-shelf software, which follows naturally from their strategy of targeting end users who use desktop applications; but it also means they cannot readily customize their product offerings. High-end technology adopters want customized solutions. OEM licensing arrangements are much more suited to the small size of our company, and our ability to customize our technologies for individual clients will allow for premium pricing of those technologies. There is substantially more *immediate* market demand for MathML editing and rendering technologies than for the functionality provided for by MathQuery. Nevertheless, the two technologies are complementary, and our ability to demonstrate how they can work together will improve the salability of each.

Second, we expect that substantial benefits can be gained from MathQuery without the need to implement a complete intelligent tutoring system. To implement hints inside

of an online homework system, for example, a few simple MathQuery expressions would suffice to determine an appropriate hint. Integration of MathQuery into existing systems could be done incrementally, which would give licensees an opportunity to evaluate MathQuery in their products without investing substantial resources in content creation.

As suggested earlier, our hope is that the MathQuery specification will evolve into a widely accepted standard, or will be integrated into the QTI standard. A MathQuery instruction would not use a proprietary file format; rather, it would be little more than an XML document that is freely open to inspection. Under those conditions, what intellectual property can be protected? As with many XML applications, the protected intellectual property is not in the end content, but in the user interface that is used to create that content. Integre's user interface to MathQuery authoring, which will be especially unique because of our MathML editing technology, will be sold only as compiled object code that is protected by copyright and appropriate software licensing.

Part 6: Consultants and Subawards

No consultants or subawards will be used in Phase I of this project.

Part 7: Equivalent or Overlapping Proposals to Other Federal Agencies

No equivalent or overlapping proposals have been submitted to the National Science Foundation or other federal agencies.

Part 8: Letters of Support or Commitment

No commitment letters are required.

References

- [1] Mathematical Markup Language (MathML). <http://www.w3c.org/Math/>
- [2] XML Path Language (XPath). <http://www.w3c.org/TR/xpath>
- [3] “IMS Question & Test Interoperability Specification.” <http://www.imsproject.org/question/index.cfm>
- [4] “IMS Question & Test Interoperability Specification: A Review.” <http://www.imsproject.org/question/whitepaper.pdf>
- [5] “Online or on Paper, States Must Mesh Testing with Federal Mandates,” *Educational Marketer*, Feb 18, 2002.
- [6] “State Interest in Putting Tests Online Grows,” *Educational Marketer*, Sept 2, 2002.
- [7] “Computer-Aided Assessment in Mathematics: Panacea or Propaganda?” *CAL-laborate*, **9**, October 2002. <http://science.uniserve.edu.au/pubs/callab/vol9/lawson.html>
- [8] Stanley Rabinowitz and Tamara Brandt, “Computer-Based Assessment: Can It Deliver on Its Promise?” WestEd research report produced under contract to the U.S. Department of Education, 2001. http://www.wested.org/online_pubs/kn-01-05.pdf
- [9] “Improving America’s Schools: A Newsletter on Issues in School Reform,” U.S. Department of Education, Spring 1996. <http://www.ed.gov/pubs/IASA/newsletters/assess/index.html>
- [10] “Title I: Amendments to the Elementary and Secondary Education Act of 1965,” U.S. Department of Education, 1996. <http://www.ed.gov/legislation/ESEA/sec1001.html>
- [11] Carmen Chapman, “Authentic Writing Assessment,” *Practical Assessment, Research & Evaluation*, **2**(7), 1990. <http://edresearch.org/pare/>
- [12] Lorraine Valdez Pierce, “Performance-Based Assessment: Promototing Achievement for English Language Learners,” Center for Applied Linguistics report. <http://www.cal.org/ericcl1/News/2002fall/performance.html>
- [13] Tej Pandey, “Authentic Mathematics Assessment,” *Practical Assessment, Research & Evaluation*, **2**(1), 1990. <http://edresearch.org/pare/>
- [14] Raymond Ravaglia, Theodore Alper, Marianna Rozenfeld, and Patrick Suppes, “Successful Pedagogical Applications of Symbolic Computation,” in N. Kajler (ed.), *Computer-Human Interaction in Symbolic Computation* (New York: Springer, 1999).

- [15] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray Pelletier, “Cognitive Tutors: Lessons Learned,” *Journal of Learning Sciences*, **4** (1995) 167–207.
- [16] Erica Melis, Eric Andrès, Jochen Büdenbender, et al., “ActiveMath: A Generic and Adaptive Web-Based Learning Environment,” *Artificial Intelligence in Education*, **12**(4), 2001. <http://www.activemath.org/#Publications>
- [17] Rachel E. DiPaolo, Arthur C. Graesser, Douglas J. Hacker, and Holly A. White, “Hints in Human and Computer Tutoring,” in Mitchell Rabinowitz (ed.), *The Impact of Media on the Technology of Instruction* (Mahwah NJ: Erlbaum, forthcoming). <http://mnemosyne.csl.psyc.memphis.edu/trg/papers/Hints2.html>.
- [18] John E. Simpson, *XPath and XPointer* (Sebastopol, CA: O’Reilly, 2002).