# Accessible Integrated Formal Reasoning Environments in Classroom Instruction of Mathematics

Andrei Lapets
Boston University
lapets@bu.edu

## 1 Introduction and Motivation

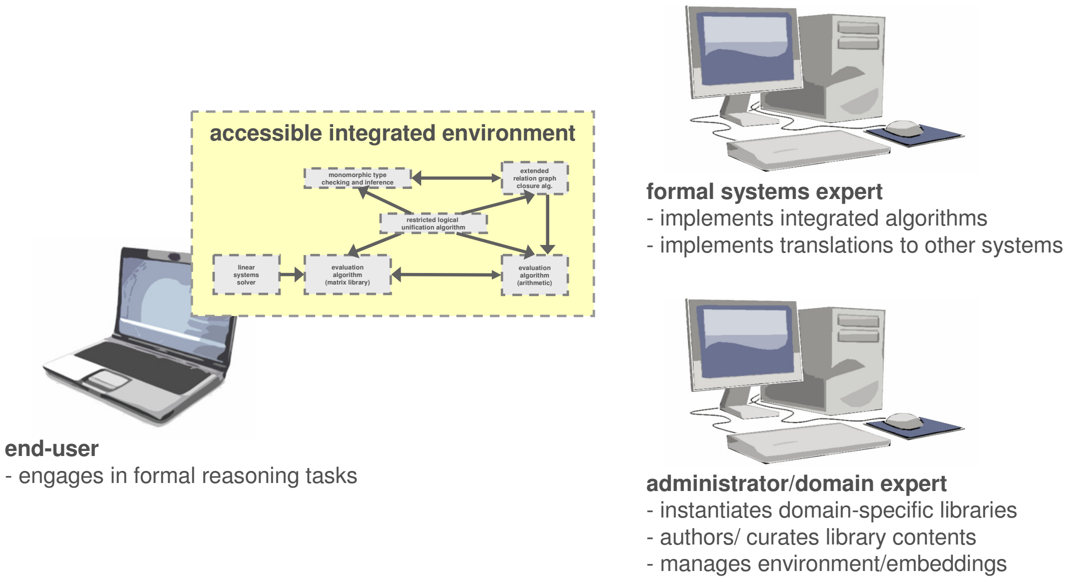
Computer science researchers in the programming languages and formal verification communities, among others, have produced a variety of automated assistance and verification tools and techniques for formal reasoning: parsers, evaluators, proof-authoring systems, software verification systems, interactive theorem provers, model-checkers, static analysis methods, and so on. While there have been notable successes in utilizing these tools on the development of safe and secure software and hardware, these leading-edge advances remain largely underutilized by large populations of potential users that may benefit from them; among these are instructors and students engaged in the undergraduate-level instruction and study of mathematics.

Tools such as MATLAB and Mathematica are utilized by instructors and students in mathematics courses at the undergraduate level. Existing tools such as Isabelle/Isar [25], Mizar [24], Coq, or even basic techniques such as monomorphic type checking, computation of congruence closures, and automated logical inference, provide an unambiguous way to represent and verify formal arguments, from chains of basic algebraic manipulations in arithmetic all the way up to graduate-level mathematics topics. What impediments cause these tools to be so seldom used in presenting and assembling even basic algebraic arguments and proofs in undergraduate mathematics courses? Impediments faced by instructors and students who may wish to adopt existing automated formal reasoning assistance and verification tools and techniques include (we cite examples of related work that shares our motivation in addressing each impediment listed): syntaxes that may be unfamiliar and entirely distinct from the syntax used in textbooks and lecture materials [13], a lack of extensive high-level domain-specific libraries [7] (especially libraries that would make formal arguments as manageable as those found in a textbook or as those typically presented in the classroom [1, 5, 23]), a lack of indexing, search, and exploration capabilities that would allow end-users to explore and employ the contents of such libraries [3, 9, 14, 20, 21], a potentially idiosyncratic semantics focused on low-level logical correctness rather than practical high-level relationships typically explored in a mathematics course, a steep learning curve in learning to use existing implementations of a tool or technique [3, 11], and logistical difficulties in setting up a working environment so that both the instructor and students can use the same tool without much supporting IT infrastructure [11, 12].

Building on earlier work in assembling and evaluating user-friendly and accessible formal verification tools for research and classroom instruction [10, 15, 16, 17, 18, 19], we have recently attempted to address many of these issues while making more manageable the task of utilizing existing formal reasoning assistance and verification techniques within the classroom. We have assembled a web-based accessible integrated formal reasoning environment for classroom instruction that incorporates several standard techniques drawn from work done by the programming languages and formal verification communities over the last several decades. These techniques include basic evaluation of expressions, monomorphic type inference, basic verification of a subset of logical formulas in propositional and first-order logic, and an algorithm for computing congruence closures. This interactive web-based environment can run entirely within a standard web browser, and can be seamlessly integrated with a web page of lecture materials and homework assignments. The lectures and assignments contain within them formal arguments (including both complete examples and problems to be completed) that can be evaluated and/or automatically verified interactively. The environment provides an explicit library of logical definitions and formulas that students can utilize to complete assignments; these formulas can be browsed and filtered by students within the environment. The environment provides interactive and instant verification feedback about logical validity as well as an interactive, exhaustive list of all inferred properties for a given formal argument input.

## 2 Overview of Components, Deployment, and Interfaces

The design of the accessible integrated environment is built on an infrastructure with components that support the activities of three (not necessarily distinct) user roles (illustrated in Figure 1): *end-users* that employ the integrated environment to perform formal reasoning tasks, *administrators* that populate and manage the environment for end-users, and *formal systems experts* that instantiate an environment by integrating the underlying components.



accessible integrated environment
monomorphic type checking and inference
extended relation graph closure alg.
restricted logical unification algorithm
linear systems solver
evaluation algorithm (matrix library)
evaluation algorithm (arithmetic)
formal systems expert
- implements integrated algorithms
- implements translations to other systems
end-user
- engages in formal reasoning tasks
administrator/domain expert
- instantiates domain-specific libraries
- authors/ curates library contents
- manages environment/embeddings

Figure 1: User roles supported by the accessible integrated environment infrastructure.

Consider the following roles within the context of the application domain being considered: students and instructors of an undergraduate linear algebra course in a mathematics department; the role of end-user corresponds to a student, and the role of the administrator corresponds to a course instructor. First, a formal systems expert instantiates an accessible integrated environment by defining several algorithms relevant to the application domain (linear algebra). The administrator can then present in lectures examples of formal arguments (e.g., proofs of theorems and solutions to problems) in a standard syntax that can be processed automatically, and can have end-users (students) write their homework solutions while enjoying the benefits of an interactive, automatic verification tool that requires no special environment or setup procedure.

**Integrated Components.** For this application domain, the formal systems expert uses the infrastructure's syntax metalanguage to define a common input language to be used by end-users (its syntax is a subset of first-order logic inspired by common conventions used in LaTeXand MATLAB). Using the tools made available by the infrastructure, the expert also implements several common algorithms over the input language: a monomorphic type checking algorithm, an extended relational graph closure algorithm, a restricted logical unification algorithm, a linear systems solver, and some evaluation algorithms. Figure 2 illustrates these algorithms and how they interact when they are collectively applied to an input in the language.

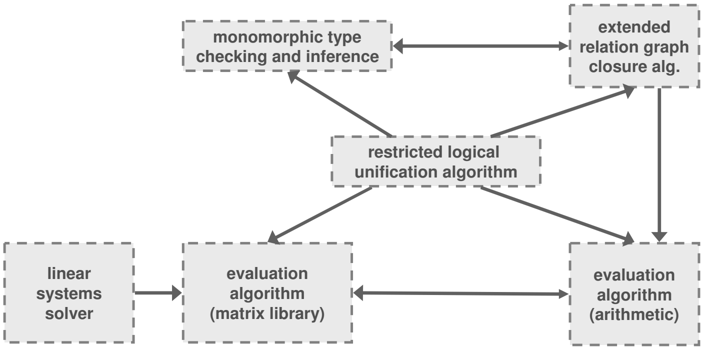

monomorphic type checking and inference
extended relation graph closure alg.
restricted logical unification algorithm
linear systems solver
evaluation algorithm (matrix library)
evaluation algorithm (arithmetic)

Figure 2: Interacting components underlying the accessible integrated environment.

**Deployment of the Accessible Integrated Environment.** Once instantiated, the accessible integrated environment is deployed to the cloud as a web application (illustrated in Figure 3). The integrated components implemented by the formal systems expert are compiled into (1) JavaScript and PHP parsers and pretty-printers for the common input language (to be used by the end-user interface, course notes webpage, database of facts to be curated by the administrator, and the content management system), and (2) an interactive web application that can invoke JavaScript implementations of the integrated algorithms implemented by the formal systems expert.

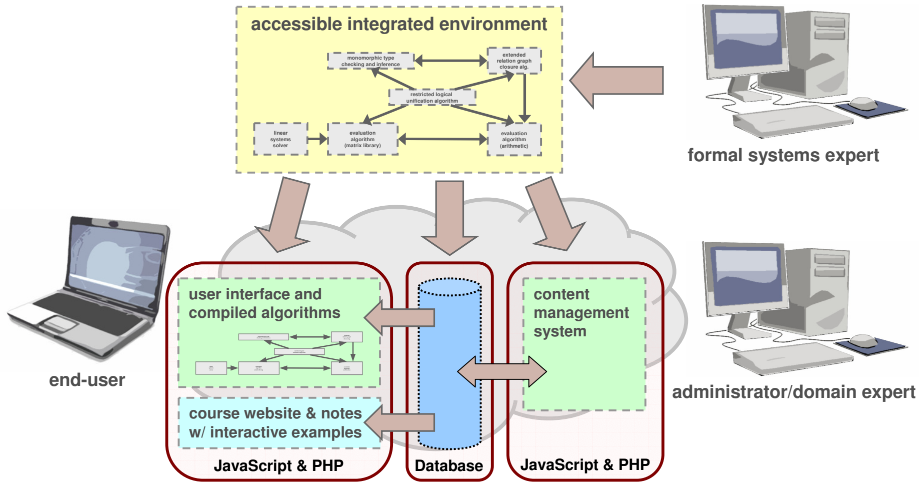
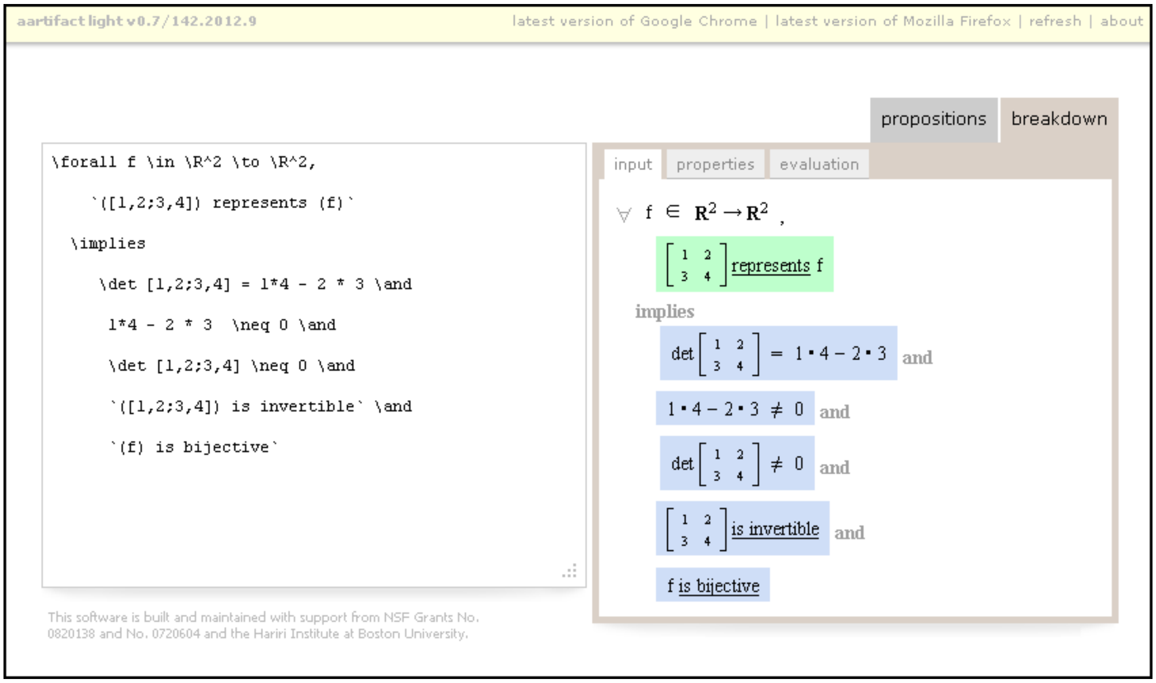

accessible integrated environment
monomorphic type checking and inference
extended relation graph closure alg.
restricted logical unification algorithm
linear systems solver
evaluation algorithm (matrix library)
evaluation algorithm (arithmetic)
formal systems expert
end-user
user interface and compiled algorithms
course website & notes w/ interactive examples
JavaScript & PHP
Database
content management system
JavaScript & PHP
administrator/domain expert

Figure 3: Deployment of the integrated environment as a web application on the cloud.

**Web Interfaces for End-users and Administrators.** The web-based integrated environment provides the following interfaces for end-users (students) and administrators (course instructors).

- An interactive web-based verification environment that allows a user (e.g., a student) to interactively construct a formal argument while receiving instant verification feedback based on the results of one or more integrated verification algorithms (e.g., an evaluation algorithm for expressions containing matrices, a simple logical inference algorithm based on computation of congruence closures, and a basic logical verification algorithm based on unification). Figure 4 illustrates how our own prototype environment appears within a standard web browser when it is used to author a basic linear algebra argument.


aartifact light v0.7/142.2012.9
latest version of Google Chrome | latest version of Mozilla Firefox | refresh | about
propositions
breakdown
input
properties
evaluation

```
\forall f \in \R^2 \to \R^2,
    `([1,2;3,4]) represents (f)`
  \implies
     \det [1,2;3,4] = 1*4 - 2 * 3 \and
      1*4 - 2 * 3  \neq 0 \and
      \det [1,2;3,4] \neq 0 \and
      `([1,2;3,4]) is invertible` \and
      `(f) is bijective`
```

$\forall\ f \in \mathbf{R}^2 \to \mathbf{R}^2$ ,

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ represents f

implies

$\det \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = 1 \cdot 4 - 2 \cdot 3$ and

$1 \cdot 4 - 2 \cdot 3 \neq 0$ and

$\det \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \neq 0$ and

$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ is invertible and

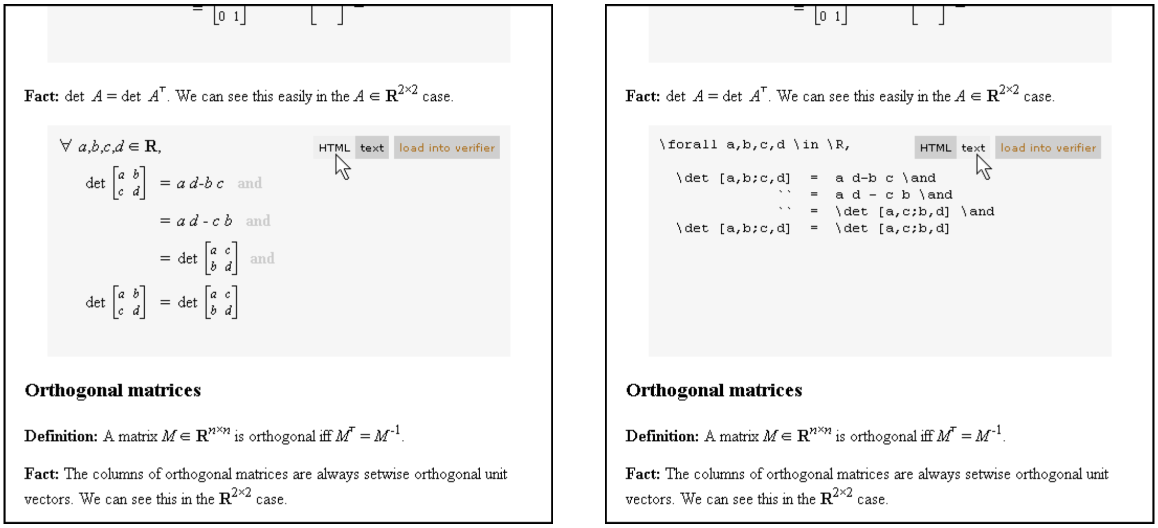
f is bijective

This software is built and maintained with support from NSF Grants No. 0820138 and No. 0720604 and the Hariri Institute at Boston University.

Figure 4: Screen capture of end-user interface.

- A content management system (CMS) for authoring and posting lecture notes online. This CMS should be extended with features that allow inclusion of formal arguments that are automatically verifiable. Students who view the lecture notes would be able to see both a "friendly" version of a formal argument, or its raw source (as illustrated in Figure 5). Students should be able to load any formal argument instantly into the environment (as illustrated in Figure 6), whether it is complete or, in the case of an exercise or assignment to be completed by students, incomplete.

- A tool for assembling and including in the lecture notes problems that students must solve within the interactive environment (including, for each problem, a subset of formal facts that students may employ). An instructor should be able to use the CMS to assemble a collection of facts that may be used in a given assignment. When students load a particular assignment into the environment as in (2) above, they should be able to see, browse, and search using keywords the formal facts the instructor has made available for them to use in their solutions (as illustrated in Figure 7).



**Fact:** det $A$ = det $A^T$. We can see this easily in the $A \in \mathbf{R}^{2\times 2}$ case.

$\forall\ a,b,c,d \in \mathbf{R}$,

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a\,d\text{-}b\,c \quad \text{and}$$
$$= a\,d - c\,b \quad \text{and}$$
$$= \det \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \text{and}$$
$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \det \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

**Orthogonal matrices**

**Definition:** A matrix $M \in \mathbf{R}^{n\times n}$ is orthogonal iff $M^T = M^{-1}$.

**Fact:** The columns of orthogonal matrices are always setwise orthogonal unit vectors. We can see this in the $\mathbf{R}^{2\times 2}$ case.

**Fact:** det $A$ = det $A^T$. We can see this easily in the $A \in \mathbf{R}^{2\times 2}$ case.

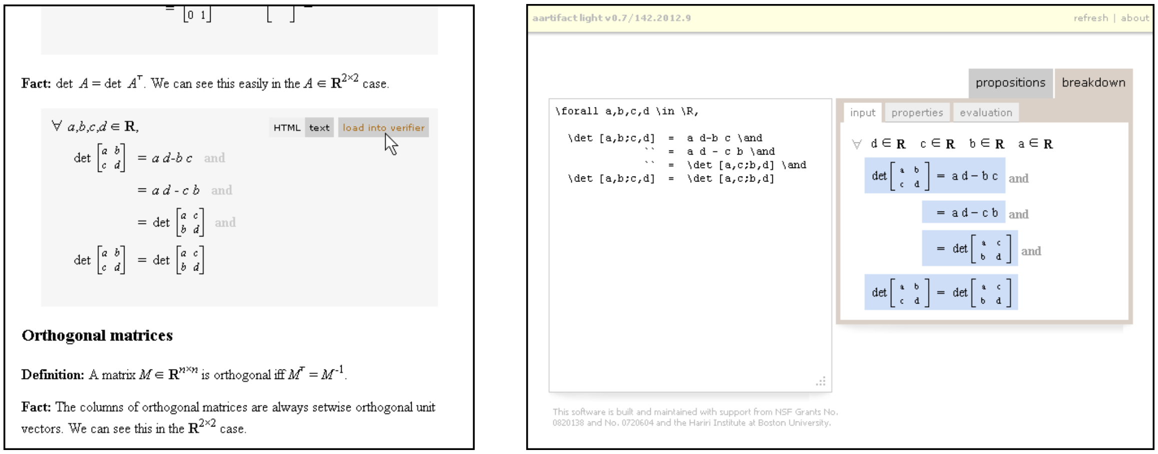
```
\forall a,b,c,d \in \R,

  \det [a,b;c,d]   =   a d-b c \and
               ``   =   a d - c b \and
               ``   =   \det [a,c;b,d] \and
  \det [a,b;c,d]   =   \det [a,c;b,d]
```

**Orthogonal matrices**

**Definition:** A matrix $M \in \mathbf{R}^{n\times n}$ is orthogonal iff $M^T = M^{-1}$.

**Fact:** The columns of orthogonal matrices are always setwise orthogonal unit vectors. We can see this in the $\mathbf{R}^{2\times 2}$ case.

Figure 5: Screen captures of formal arguments integrated into online lecture notes.



**Fact:** det $A$ = det $A^T$. We can see this easily in the $A \in \mathbf{R}^{2\times 2}$ case.

$\forall\ a,b,c,d \in \mathbf{R}$,

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a\,d\text{-}b\,c \quad \text{and}$$
$$= a\,d - c\,b \quad \text{and}$$
$$= \det \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \text{and}$$
$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \det \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

**Orthogonal matrices**

**Definition:** A matrix $M \in \mathbf{R}^{n\times n}$ is orthogonal iff $M^T = M^{-1}$.

**Fact:** The columns of orthogonal matrices are always setwise orthogonal unit vectors. We can see this in the $\mathbf{R}^{2\times 2}$ case.

```
\forall a,b,c,d \in \R,

  \det [a,b;c,d]   =   a d-b c \and
               ``   =   a d - c b \and
               ``   =   \det [a,c;b,d] \and
  \det [a,b;c,d]   =   \det [a,c;b,d]
```

$\forall\ d \in \mathbf{R} \quad c \in \mathbf{R} \quad b \in \mathbf{R} \quad a \in \mathbf{R}$

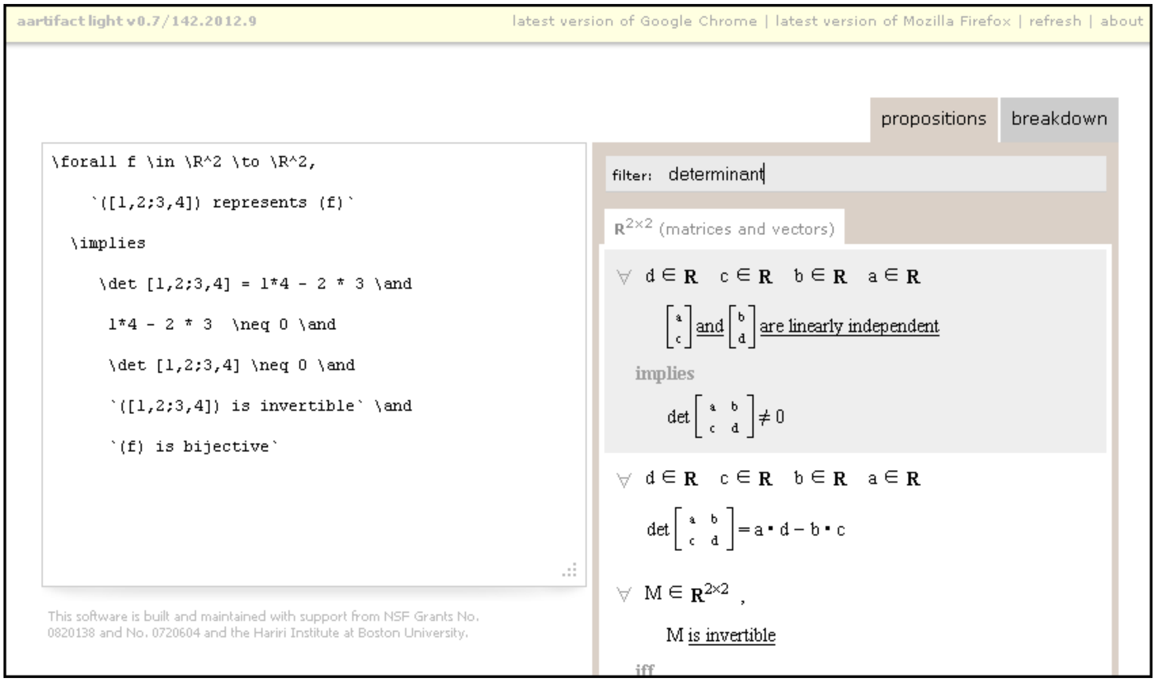
$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a\,d - b\,c \quad \text{and}$$
$$= a\,d - c\,b \quad \text{and}$$
$$= \det \begin{bmatrix} a & c \\ b & d \end{bmatrix} \quad \text{and}$$
$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \det \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$

This software is built and maintained with support from NSF Grants No. 0820138 and No. 0720604 and the Hariri Institute at Boston University.

Figure 6: Loading a formal argument in the lecture notes into the environment.

# 3 Concluding Remarks

A prototype web-based integrated formal reasoning environment has been utilized within the classroom for an undergraduate-level course in linear algebra, both by the instructor in presenting examples during lectures, and by students when completing homework assignments. The focus of this work is not on the *expressiveness* or *performance* of the underlying tools and techniques within the environment (most examples used in such a course are simple); rather, it is on the design and *practical utilization* of an *integrated* environment of basic, existing tools and techniques that is *accessible* (logistically and semantically) to non-expert end-users. We found that even basic capabilities are helpful in teaching students what is a formal argument and how to assemble a formal argument by examining possible formulas and applying those formulas while working towards a goal. Students were also able to transfer to an exam setting the techniques they used in assembling formal arguments within the environment.


aartifact light v0.7/142.2012.9
latest version of Google Chrome | latest version of Mozilla Firefox | refresh | about
propositions
breakdown
filter: determinant
$\mathbf{R}^{2\times 2}$ (matrices and vectors)

```
\forall f \in \R^2 \to \R^2,
    `([1,2;3,4]) represents (f)`
  \implies
     \det [1,2;3,4] = 1*4 - 2 * 3 \and
      1*4 - 2 * 3  \neq 0 \and
      \det [1,2;3,4] \neq 0 \and
      `([1,2;3,4]) is invertible` \and
      `(f) is bijective`
```

$\forall$ $d \in \mathbf{R}$ $c \in \mathbf{R}$ $b \in \mathbf{R}$ $a \in \mathbf{R}$

$\begin{bmatrix} a \\ c \end{bmatrix}$ and $\begin{bmatrix} b \\ d \end{bmatrix}$ are linearly independent

implies

$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} \neq 0$

$\forall$ $d \in \mathbf{R}$ $c \in \mathbf{R}$ $b \in \mathbf{R}$ $a \in \mathbf{R}$

$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = a \cdot d - b \cdot c$

$\forall$ $M \in \mathbf{R}^{2\times 2}$ ,

M is invertible

iff

This software is built and maintained with support from NSF Grants No. 0820138 and No. 0720604 and the Hariri Institute at Boston University.

Figure 7: Screen capture of end-user interface.

# 4 Related Work

This work aims to address the disincentives to utilizing automated formal reasoning assistance systems by integrating multiple systems within an accessible environment. We share motivation with, are inspired by, and incorporate ideas from related efforts to address practical usability in the formal systems communities. Some aim to provide interfaces that have a familiar syntax [2, 13, 22, 26]; some aim to make optional the need to provide explicit references to the formal facts being used within the individual steps of a formal argument [1, 5, 23]; some aim to eliminate steep learning curves [3, 6, 11]; some aim to reduce the logistical difficulties of utilizing automated formal reasoning assistance systems [11, 12]. We are inspired by search mechanisms for libraries of formal facts [4, 7, 8] and programming language constructs [21], as well as keyword-based lookup mechanisms for programming environments [9, 20]. Providing the functionality of a formal reasoning environment within a browser is a goal that has been adopted by some projects [12], though that work focuses on delivering the look and functionality of an existing proof assistant.

# References

[1] A. Abel, B. Chang, and F. Pfenning. Human-readable machine-verifiable proofs for teaching constructive logic. In U. Egly, A. Fiedler, H. Horacek, and S. Schmitt, editors, *PTP '01: IJCAR Workshop on Proof Transformations, Proof Presentations and Complexity of Proofs*, Siena, Italy, 2001.

[2] E. Allen, D. Chase, J. Hallett, V. Luchangco, J.-W. Maessen, S. Ryu, G. L. S. Jr., and S. Tobin-Hochstadt. The Fortress Language Specification Version 1.0. March 2008.

[3] A. Asperti, C. S. Coen, E. Tassi, and S. Zacchiroli. User interaction with the matita proof assistant. *Journal of Automated Reasoning*, 39(2):109–139, 2007.

[4] G. Bancerek and J. Urban. Integrated semantic browsing of the Mizar Mathematical Library for authoring Mizar articles. In *MKM*, pages 44–57, 2004.

[5] C. E. Brown. Verifying and Invalidating Textbook Proofs using Scunak. In *MKM '06: Mathematical Knowledge Management*, pages 110–123, Wokingham, England, 2006.

[6] R. M. Burstall. Proveeasy: helping people learn to do proofs. *Electr. Notes Theor. Comput. Sci.*, 31:16–32, 2000.

[7] P. Cairns and J. Gow. Integrating Searching and Authoring in Mizar. *Journal of Automated Reasoning*, 39(2):141–160, 2007.

[8] T. Hallgren and A. Ranta. An extensible proof text editor. In *LPAR '00: Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning*, pages 70–84, Berlin, Heidelberg, 2000. Springer-Verlag.

[9] S. Han, D. R. Wallace, and R. C. Miller. Code completion from abbreviated input. In *ASE '09: Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*, pages 332–343, Washington, DC, USA, 2009. IEEE Computer Society.

[10] V. Ishakian, A. Lapets, A. Bestavros, and A. Kfoury. Formal Verification of SLA Transformations. In *Proceedings of CloudPerf 2011: IEEE International Workshop on Performance Aspects of Cloud and Service Virtualization*, Washington, D.C., USA, July 2011.

[11] D. Jackson. Alloy: a lightweight object modelling notation. *Software Engineering and Methodology*, 11(2):256–290, 2002.

[12] C. Kaliszyk. Web interfaces for proof assistants. *Electronic Notes in Theoretical Computer Science*, 174(2):49–61, 2007.

[13] F. Kamareddine and J. B. Wells. Computerizing Mathematical Text with MathLang. *Electronic Notes in Theoretical Computer Science*, 205:5–30, 2008.

[14] S. Katayama. Library for systematic search for expressions. In *AIC '06: Proceedings of the 6th WSEAS International Conference on Applied Informatics and Communications*, pages 381–387, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS).

[15] A. Lapets. Lightweight Formal Verification in Classroom Instruction of Reasoning about Functional Code. Technical Report BUCS-TR-2009-032, CS Dept., Boston University, November 2009.

[16] A. Lapets. User-friendly Support for Common Concepts in a Lightweight Verifier. In *Proceedings of VERIFY-2010: The 6th International Verification Workshop*, Edinburgh, UK, July 2010.

[17] A. Lapets and D. House. Efficient Support for Common Relations in Lightweight Formal Reasoning Systems. Technical Report BUCS-TR-2009-033, CS Dept., Boston University, November 2009.

[18] A. Lapets and A. Kfoury. Verification with Natural Contexts: Soundness of Safe Compositional Network Sketches. Technical Report BUCS-TR-2009-030, CS Dept., Boston University, October 1 2009.

[19] A. Lapets and A. Kfoury. A user-friendly interface for a lightweight verification system. *Electronic Notes in Theoretical Computer Science*, 285(0):29 – 41, 2012. Proceedings of the 9th International Workshop On User Interfaces for Theorem Provers (UITP10).

[20] G. Little and R. C. Miller. Keyword programming in java. In *ASE '07: Proceedings of the 22nd IEEE/ACM International Conference on Automated Software Engineering*, pages 84–93, New York, NY, USA, 2007. ACM.

[21] N. Mitchell. Hoogle overview. *The Monad.Reader*, 12:27–35, November 2008.

[22] P. Rudnicki. An overview of the Mizar project. In *Proceedings of the 1992 Workshop on Types and Proofs for Programs*, pages 311–332, 1992.

[23] J. H. Siekmann, C. Benzmüller, A. Fiedler, A. Meier, and M. Pollet. Proof Development with OMEGA: sqrt(2) Is Irrational. In *LPAR*, pages 367–387, 2002.

[24] A. Trybulec and H. Blair. Computer Assisted Reasoning with MIZAR. In *Proceedings of the 9th IJCAI*, pages 26–28, Los Angeles, CA, USA, 1985.

[25] M. Wenzel. *The Isabelle/Isar Reference Manual*. January 2011.

[26] M. M. Wenzel. *Isabelle/Isar - A versatile environment for human-readable formal proof documents*. PhD thesis, Institut für Informatik, Technische Universität München, 2002.