

# Creating and Evaluating Interactive Formal Courseware for Mathematics and Computing

Robert L. Constable  
Department of Computer Science  
Cornell University  
Ithaca, NY 14853

## Abstract

*The application of the Nuprl proof development system to the teaching of college-level mathematics is explored. The advantages of this method include the flexibility to accommodate the needs of students with very diverse backgrounds and the creation of an environment that encourages collaborative learning.*

## Introduction

The educational heart of our research is a plan to explore the value of formal explanation in course settings where this is possible and even natural. By a *formal explanation* we mean one that can be easily reduced to a highly readable proof in a formal logic. The entire enterprise rests on the results of over 25 years of research that produced a new generation of reasoning tools such as Nuprl.

Explanation itself is central to education. Whether we are talking about an English essay, a chemistry experiment, a legal argument or a mathematical demonstration, college students are taught to answer the question, "How do you know?" They are taught to give evidence and to say what statements follow from others. This ability to provide evidence and evaluate arguments is critical to a liberal arts education or an engineering one.

In many fields there are *systematic methods* to find errors in arguments. The added value of formal explanation is that computers can help us create them and check them, making a systematic search for errors quickly and thoroughly. The value of computers in logic education was definitively established in the 60's, but these early systems were limited to small domains of logic. The advantage of adapting a mature research tool such as Nuprl to education is that the system will not "run out" inconveniently when a certain degree of sophistication is reached.

We believe that these tools have created considerable educational opportunity. The main opportunity is in creating new courseware, but we can also

study the use of the proof development system to help students solve exercises and experiment on the courseware.

Since 1975 we have been building these *proof development systems* -- computer systems that help people create formal proofs. Our latest version is Nuprl 4, the first major *theorem prover* to treat proofs as mathematical objects. Nuprl has been used for hardware verification, for programming language semantics and as a programming logic, for supporting symbolic algebra, and for building components of theory provers.

We have taught formal mathematics using Nuprl and discovered an exciting way to present material. The method requires careful explanation because it is based on some features of *computer supported formal mathematics*. The idea is that an extensive and completely formalized body of mathematics is used as the *underlying reference material* for a course; the lectures, lessons, and exercises are then built on top of it using both ordinary text and Nuprl's *formal hypertext*. These lessons offer great scope for teaching innovation, and are especially suited to the notion of the professor and class as collaborators in exploring "a network of knowledge" supported by knowledge processing computer systems.

The educational advantages of this approach are numerous, including:

- The level of detail at which students interact with material is *flexible*, from high-level summaries and guided readings down to an examination of *every detail* and every lemma required for a proof.
- There are *diverse entry points* to match the diversity of the student body. Students can also contribute their own commentary or annotate what is already provided. This personalizes the course material and encourages *collaborative learning*.
- Students know both the goals (e.g. asking for proofs) and what methods are allowed to reach them. This setting is highly conducive to learning *technique* and sets up patterns for organizing and justifying which can then be related to the more nebulous ideas of

presenting and explaining concepts and making informal arguments.

- The vocabulary of *goals*, *subgoals*, *rules*, *tactics*, and *lemmas* enables us to discretize and quantify some of the learning, a major contribution to pedagogy.
- Many of the exercises are *self-checking*: either the theorem prover accepts the inference step, or it does not. This enables students to work at their own pace.
- The material is based on a library of reference material annotated and connected by a hypertext web which enables students to make personal contributions to the course knowledge base and thus engages them with the material.
- Since Nuprl's formal language can express virtually any mathematical concept, we can begin to teach understanding as a process of making connections between topics.

## The Nuprl Computational Mathematics Medium

A Nuprl *library* is a sequence of various kinds of objects, including definitions, theorems and comments. Libraries are divided into sections called *theories*. Here we give a partial listing of a Nuprl theory covering basic divisibility theory over the integers  $\mathbb{Z}$ , showing the definitions and theorems that lead up to a theorem about the existence of GCD's (greatest common divisors).

The definition of the 'divides' relation is:

*A divides	$b \mid a \equiv \exists c: \mathbb{Z}. a = b \cdot c$
*T divisor_of_sum	$\forall a, b_1, b_2: \mathbb{Z}. a \mid b_1, a \mid b_2 \Rightarrow a \mid b_1 + b_2$
*A gcd_p	$y \text{ Is\_a\_GCD\_of } a \text{ and } b \equiv y \mid a \wedge y \mid b \wedge \forall z: \mathbb{Z}. (z \mid a \wedge z \mid b) \Rightarrow z \mid y$
*T gcd_exists_n	$\forall b: \mathbb{N}, a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b)$

In the library listing format shown here, each object description starts with a symbol indicating the object's status and a letter indicating the kind of object. Here, the \* shows that the object is complete and the ? shows that this is an *abstraction*, Nuprl's name for a definition.

Many sample libraries can be viewed on the WorldWideWeb @www.cs.cornell.edu/Info/Projects/Nuprl.

### Hypertext, Terms, and Proofs

Typical lesson material consists of documents blending formal with informal text which are intended to be read with Nuprl's interactive editor. One particularly useful hypertext link is one that points to a proof of a theorem. One may, of course, create an ordinary hypertext link to a proof, but often it is more convenient to show the content of the theorem at its point of use, and so Nuprl includes links that display content. For example:

THM\*  $\forall b: \mathbb{N}, a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b)$

When the user clicks on this, a window is opened on the proof of the displayed theorem. Such a link to a theorem may be embedded in informal hypertext, and may also be used within proofs of other theorems that use it as a lemma.

### Reading Proofs

Proofs are presented as trees of inference steps through which one walks with the editor; the top of the proof pointed to by the link shown above looks like:

```
* top
Ã   $\forall b: \mathbb{N}, a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b)$ 
BY Induction
1* 1.  $b: \mathbb{N}$ 
    2.  $\forall b_1: \mathbb{N}. b_1 < b \Rightarrow \forall a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b_1)$ 
       ...Ind Hyp
       Ã   $\forall a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b)$ 
```

The “\* top” indicates that the proof is complete and that this is the top of the proof tree. The goal is followed by the tactic used for the inference, and the numbered list of subgoals automatically generated from the goal and the tactic. If the user wishes to see the proof of a subgoal, clicking on it will show:

```
* top 1
1.  $b: \mathbb{N}$ 
2.  $\forall b_1: \mathbb{N}. b_1 < b \Rightarrow \forall a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b_1)$ 
   ...Ind Hyp
+  $\forall a: \mathbb{Z}. \exists y: \mathbb{Z}. \text{GCD}(y; a, b)$ 
BY AnalyzeConcl THEN CaseSplitOn  $b = 0$ 
1* 3.  $a: \mathbb{Z}$ 
    4.  $b = 0$ 
    +  $\exists y: \mathbb{Z}. \text{GCD}(y; a, b)$ 
2* 3.  $a: \mathbb{Z}$ 
    b.  $-b = 0$ 
    +  $\exists y: \mathbb{Z}. \text{GCD}(y; a, b)$ 
```

The “\* top” indicates that the proof of this goal is complete, and also indicates its location within the tree of inference steps, namely, the first subgoal below the top of the proof. The user would simply continue reading down the various branches of the proof until steps with no subgoals are reached or until he or she is satisfied.

### Definitions and Blending Materials

The ability to find the definition of an operator from any use of it throughout the system is a key feature of the Nuprl editor. For example, if the user wanted to see the definition of “divides,” he or she could click on any occurrence of it, such as the “ $n \cdot a \mid n \cdot b$ ”, and the definition would be raised, which looks like this:

$b \mid a \equiv \exists c: \mathbb{Z}. a = b \cdot c$

←

Read “b divides a”. VIEW → IntegerDivisionNotes

The comment indicates the plain language reading, and contains a hypertext link to a document about integer division. The occurrences of the “divides” operator we have been discussing are formal no matter where they occur, either in proofs or in informal documents. Besides the benefit of being able to find the definition, if the user wants to change the way an operator is displayed, say to “x divides y” instead of “ $x \mid Y$ ”, this change will be automatically adopted throughout the system since it is the abstract expression rather than the text “ $x \mid y$ ” that is used.

Formal and informal material can be freely mixed. If this document had been presented on the Nuprl system, for example, you could actually have clicked on the theorem references and operator occurrences occurring above with the described effects. So one can write informally about a subject but use the formal notations whenever desired, thus allowing the user full access to relevant formal materials such as definitions and proofs, and any other documentation that has been attached to the formal materials. Conversely, one can insert commentary inside the formal objects of the system, as was trivially exemplified by the comment in the definition shown above.

Nuprl can also refer to external on-line documents with hypertext-like links that allow the user to initiate independent external processes such as Web browsers and document viewers. This means that Nuprl documents can effectively incorporate on-line material that it would not be economical to build internally in Nuprl, such as drawings, textbooks, and papers.

## Educational Aspects

The main application of Nuprl is in creating the interactive formal courseware; this will solve a number of the problems common to any technical subject and to mathematics in particular, such as

- the *omission* of key justification steps in proofs, or of entire proofs;
- the *difficulty of access* to resources for finding key definitions, lemmas, definitions, or remarks;
- the *ambiguity* of expressions that could be interpreted in more than one way;
- *missing motivation* for proof steps.

Since the power of PC's is now adequate to run Nuprl, these problems can be addressed directly, thereby not only solving the problems but also integrating the techniques proven in logic teaching.

Additional advantages include the possibility of anonymously monitoring the course material to see how students use it either to solve certain problems or to study for exams, whether they use the material as we anticipated. Do they follow definition chains? Do they consult the formal proof when encountering obstacles to understanding? Do they often execute the algorithms associated with computational theorems in order to produce more examples of a result? We find out which definitions are referred to most often, which expressions must be fully disambiguated, which lemmas are most often reviewed. These quantifiable measures give us a map of the difficulty of the "cognitive terrain." And last, the existence of extensive libraries of formal mathematics within the system opens the possibility that interested students will explore on their own and even create new sub-libraries.

## Summary and Impact

The implementation of this method is in its early stages, and it is possible that unforeseen problems could arise, but even at this point we can guarantee that we are generating a useful body of reference material, which can be used to greatly inform, and perhaps transform, the teaching of various topics in college mathematics and computer science. As more and more subjects are connected, students will see that one major component of understanding is linking new knowledge tightly into a solid base of fundamental concepts; they will be challenged to understand ideas that we cannot yet formalize precisely; and, they will come to know very concretely the value of continuous learning and reflection.

## Acknowledgments

I would like to thank Pauline Cameron for editing this article with me, Karla Consroe for preparing the manuscript, and Stuart Allen for the discussions concerning this topic.

## References

1. Anderson, J.R., C. F. Boyle, R. Farrell, and B. J. Reiser. "Cognitive principles in the design of computer tutors," in P. Morris, editor, *Modeling Cognition*, Wiley, 1987.
2. Barwise, John, and John Etchemendy. *Hyperproof*. Center for the Study of Language and Information, Stanford University, Stanford, California, 1994.
3. Bates, J. L., and Robert L. Constable. "Proofs as Programs," *ACM Trans. Program. Lang. and Syst.*, Vol. 7, No. 1, pp. 53-71, 1985.
4. Bruner, J. S., *The Process of Education*, Harvard University Press, Cambridge, Massachusetts, 1960.
5. Confrey, Jere. "A review of the research on student conceptions in mathematics, science, and programming." In Courtney Cazden, editor, *Review of Research in Education*, chapter 1, pp. 3-56. American Education Research Association, 1990.
6. Constable, Robert L. et al. *Implementing Mathematics with the Nuprl Development System*. Prentice-Hall, New Jersey, 1986.
7. Davis, James, and Daniel Huttenlocher. "Shared Annotation for Cooperative Learning," CSCL'95 Proceedings.
8. Goldenson, D. R. Teaching introductory programming methods using structure editing: Some empirical results. In W. C. Ryan, editor, *Proceedings of the National Educational Computing Conference 1989*, pages 194-203, Eugene, Oregon, 1989. University of Oregon, International Council on Computers in Education.
9. Greenleaf, N. "Bringing mathematics education into the algorithmic age," In J. P. Myers and M. J. O'Donnell, editors, *Constructivity in Computer Science*, pp. 199-217, New York, June 1991. Lecture Notes in Computer Science, Springer-Verlag.
10. Lakatos. I *Proofs and Refutations: The Logic of Mathematical Discovery*. Cambridge University Press, Cambridge, Massachusetts, 1976.
11. Newell, A. and Simon, H. A. *Human Problem Solving*, Prentice Hall, Englewood Cliffs, New Jersey, 1972.
12. Simon, Herbert A. "Problem solving and education," in D. Tuma and F. Reif, editors, *Problem Solving and Education: Issues in Teaching and Research*. Erlbaum, Hillsdale, New Jersey, 1980.
13. Sleeman, D. and J. S. Brown, editors. *Intelligent Tutoring Systems*. Academic Press, London, 1982.
14. Solow, Daniel, *How to Read and Do Proofs: An Introduction to Mathematical Thought Process*, John Wiley & Sons, New York, 1982.
15. Suppes, Patrick. *University-Level Computer-Assisted Instruction at Stanford: 1968-1980*, volume IMSSS. Stanford University, Stanford, California, 1981.
16. Wertheimer, R. "The geometry proof tutor: An 'intelligent' computer-based tutor in the classroom," *Mathematics Teacher*, pp. 308-313, 1990.