Introducing Logic and Formal Systems with Coq

Martin Henz* and Aquinas Hobor**

National University of Singapore

Abstract. During the past three years we have been integrating mechanized theorem proving into a traditional introductory course on formal methods. We explain our goals for adding mechanized provers to the course, and give an explanation of how we have integrated the provers into our syllabus to meet those goals. We also document some of the teaching materials we have developed for the course to date, and what our experiences have been like.

1 Introduction

National University of Singapore's School of Computing (SoC) has taught introductory formal methods to computing students for many years as course numbers CS3234 (undergraduate) and CS5209 (graduate). The first author has been teaching both CS3234 and CS5209 several times, using a fairly traditional approach, and using standard undergraduate textbooks, including *Mathematical Logic for Computer Science*, by Mordechai Ben-Ari [1], and *Logic in Computer Science*, by Michael Huth and Mark Ryan [2]. The results were, sad to say, equally "traditional":

1. A good appreciation for logic requires an understanding of an unusual number of innovative concepts, making for a "hard" module, in the eyes of students and instructors.
2. Students perceive logic as being "formalistic", studying formal methods for their own sake, rather than to achieve particular goals.
3. Average students usually find paper-and-pen-based excercises and tutorials "dry" and "boring".

The first point makes for a steep learning curve, the second decreases the motivation of students to climb the curve, and the third poses further obstacles to the desired learning outcome for those students who have enough motivation to even try.

In this situation, the second author joined the team, which proceeded to address the second and third problem (after declaring the first one as inevitable). The motto was to learn from other subject areas in computer science and successful computer science teaching projects in computer graphics (special effects), and programming languages (compilers). Learning goals can be achieved in these areas when the students can be motivated by practically relevant and appealing applications, and by shortening the gap between theory and through the use of adequate didactic tools, thereby implementing a "learning-by-doing" approach.

* Supported in part by ???.

** Supported by a Lee Kuan Yew Postdoctoral Fellowship.

Several tools are being used in the teaching of logic in computer science, including logic programming systems, model checkers and SAT solvers (and other tools for propositional logic). We found that each requires a certain learning overhead, which is often not justified, given their limited used for only one or two sections of the module. Ideally, the same tool would be used *throughout* the module, reducing the mentioned overhead to a minimum.

The team considered several choices of possible tools, and finally settled on the proof assistant Coq to be used throughout the module. Initial results were encouraging; the interactive discovery of proofs using Coq provided a useful reinforcement of the conceptual material. While not having been developed specifically for didactical use, Coq's basic concepts and metaphors proved to be sufficiently easy for third year undergraduates (and of course graduate students).

Remainder of paper. We next go through the syllabus of our course, focusing for each topic on how we have added mechanized proving to a more traditional curriculum. The initial time we ran the course, our textbook was the popular Huth and Ryan [**?**], but more recently we have been developing a considerable amount of new material [**?**], both in Coq and on paper, and the most recent time we ran the course this new material was the primary reference. When appropriate in the syllabus we try to provide pointers into this material, but readers should keep in mind that the supplementary material is very much a work in progress. We eventually hope to package this material into some kind of book. After discussing our syllabus, we describe the resulting course format, and explain its rationale. We conclude with a discussion of our experiences, both positive and negative, in developing this course.

2 Overview of Course Material

3 Course Format

Both CS3234 and CS5209 take one semester (13 weeks), and have two hours of instruction per week. In addition, the undergraduate version has a tutorial of one hour per week. The most recent time we ran CS3234, we added two hours of supervised laboratory time per week as well. Although we did not take attendance in the class or in the tutorials, we did for the labs: this allowed us to ensure that students were spending enough time working on the assignments, and allowed us to quickly address questions relating to the nitty-gritty pain of learning Coq. We feel that the labs were extremely helpful, and look to include them going forward. Graduate students might also benefit with the addition of mandatory laboratory time.

The basic content for both courses is the same: syntax and semantics of basic mathematical logics; proof methods (*e.g.*, natural deduction, induction); and applications of the above to computer science. A variety of other topics, depending on the preferences of the instructors, have also been included (*e.g.*, type theory).

We have discovered that to teach formal methods and mechanized theorem proving requires *a lot* of work for the students. The last time we taught CS3234 we required:

- 7 paper assignments (at 2% each)

- 5 Coq assignments (at 2% each)
- 6 20 minute Coq quizzes (at 2% each)
- A 1 hour paper midterm (10%)
- A 2 hour final with both Coq and paper problems (22% in Coq, 32% on paper)

As one might imagine, preparing and grading this many assignments requires a serious commitment on the part of the instructors as well. Fortunately, our department was able to allocate two teaching assistants to help giving the tutorials/laboratories and doing some of the grading; we ended up having the highest support/student ratio in the department. However, the previous year we did it all ourselves, and we had very little time other than preparing for the course. When we last taught CS5209, we tried to assign less homework, hoping that graduate students would be able to learn the material without as much supervision. We were mistaken; quite a few of our graduate students had a very hard time with Coq, which we believe was related to the lesser amount of homework. In the future we will assign more work in CS5209.

4 Discussion