DISSERTATION PROPOSAL

BENJAMIN BERMAN

1. Introduction

A long time ago my cello teacher pointed out that the way to play a difficult passage of music is not simply to grit one's teeth and keep practicing but to figure out how to make playing those notes easy. The major goal of the proposed dissertation is to reapply the same idea in the context of interactive theorem proving with Coq: I intend to show ways to make the difficult task of using Coq easier by improving the user interface. As well as solving usability problems for an important and powerful tool, many of the techniques I am developing and testing are widely applicable to other forms of coding.

The research involved in this proposed dissertation, described more fully below, breaks down into two related main parts. Part one is the development of "CoqEdit", a new theorem proving environment for Coq, based on the jEdit text editor. CoqEdit will mimic the main features of the existing environments for Coq, but will have the important property of being easily extended using Java. Part two is the development and testing of such extensions.

There are several points that I hope will become clear as I describe the proposed research below. First is that the research will make a significant positive contribution to society. While Coq is a powerful tool, and is already being used for important work, its power has come at the cost of complexity, which makes the tool difficult to learn and use. Finding and implementing better ways to deal with this complexity in the user interface of the tool can allow more users to perform a greater number, and a greater variety, of tasks.

In addition, the complexities of the tool in fact provide greater opportunities for research on user interfaces for coding in general. This is because a) the complexities are partly the result of the variety of tool features and tasks for which the tool may be used and b) the difficulties caused by the complexity may make the effects of good user interface designs more apparent. Furthermore, although Coq has properties that make it very appealing for developing programs (in particular, programs that are free of bugs), it also pushes at the boundaries of languages that programmers may consider practical. However, if, as in the proposed research, we design user interfaces that address the specific problems associated with using a language, perhaps making the user interface as integral to using a language itself as its syntax, these boundaries may shift outward. In other words,

Furthermore, one of the fundamental questions that general research on user interfaces for coding asks

–opportunities that would not be available were the research to be done with a simpler tool.

In writing this proposal, I seek to make clear several points.
Why is this work important to society?
Why is this work an intellectual contribution?
Why is this work doable?

I would like to point out that this task of improving user interfaces for theorem provers is a natural extension of work on symbolic logic and of proof assistants. Symbolic logic can make working with statements easier. Proof assistants can make working with symbolic logic easier. User interfaces can make working with proof assistants easier. Though the techniques differ, the goals are shared.

2. Ideas

2.1. **jEdit as an Interactive Theorem Proving Environment.** Harness the power of Java/Piccolo/jEdit. jEdit also used with Isabelle (CITE).

2.2. **Proof Previews.**

2.3. **Proof Tree Visualization.**

2.4. **Proof Transitions.**

2.5. **Syntax Tree Visualization.**

2.6. **Keyboard-Card Menus.**

3. Implementation

3.1. **Current.**

3.2. **Planned.**

4. Description of User Studies

4.1. **Keyboard-Card Menu Study Results.**

5. Literature Review

6. Timeline

7. Conclusion

I hope to have made several points in this proposal. First, that this is important work, both because the Coq interactive theorem prover is an important tool that could benefit significantly from improved user interfaces and because many of the ideas generalize to other forms of coding. Second, that as an intellectual challenge this work is non-trivial, not only because of the normal programming problems that must be overcome but because designing good user interfaces for complicated systems, which includes the identification of tractable problems and the testing of potential solutions, is non-trivial. Finally, that, despite this non-trivial nature, the work can be accomplished.