

# Declarative Theorem Proving Project Plan

Jimmy Thomson (u4308384)

August 15, 2008

## Background

Automated theorem provers are useful tools, especially for verifying systems [2]. However, most systems including HOL are not truly automated: They require a human to guide the proof. This means that humans must create and maintain proofs of theories.

Currently the HOL theorem prover provides the operator with a collection of ML functions [1], which can be sequenced together to describe how to prove theorems. These functions, known as tactics, are procedural in nature and require the proof to be performed one step at a time. In order for a human to understand such a proof, they must perform each step (either themselves or interactively in HOL) before they know what state the next step acts on.

Such a proof, especially as they become longer and more complex, is difficult for humans to follow, and the actions involved are different from how a human would normally prove the same theorem.

## Task Description

My task is to design a new input language which is declarative rather than procedural, and easier for humans to use. In addition to this, I will create a type checker, to find typing errors without having to perform the whole proof, and a conversion from this input language to HOL, to allow the language to be used to prove theorems.

The input language will allow for theorems to be proved using the standard theories of HOL, and using other theorems proved in the input language.

The input language will also allow for simple definitions in the logical domain. The input language will not allow for user-defined justification steps originally, but it may be added as an extension.

The type checker will check the entirety of the input file and identify type conflicts to the user.

The prover will convert the input language into HOL ML functions to perform the proof, and result in a collection of theorems, proofs and reasons why some proofs failed.

## Plan

The proposed project plan is as follows (Note that the Week column includes the lecture break in the week count, so after week 10 the count is two weeks ahead):

Week	Date	Milestone
4	15/08	Project Plan
6	29/08	Language design finalised
8	12/09	Parser and Lexer finished
10	26/09	Term 3 ends, Type checker tool finished
13	13/10	Term 4 begins, Prover tool finished
15	27/10	Final Report finished
16	03/11	Final Presentation finished

Problems are most likely to be encountered in the tool creation stages, either the type checker or the prover, but the parser and lexer are likely to not take as long as planned. If the tools do take longer, the period between week 16/09 and 13/10 is lecture free, so more time can be allocated to the project during that time.

If the tools are finished before the end of week 13, then I will attempt to include further components listed as optional in the task description above.

## References

- [1] *The HOL System Description*, 2007.
- [2] Anduo Wang, He Fei, Ming Gu, and Xiaoyu Song. Verifying java programs by theorem prover hol. *compsac*, 1:139–142, 2006.