COMP2560 - Project Summary

Alexander Soen

September 1, 2017

Project

Title

Automated Synthesis of a Tableaux Theorem Prover for Classical Propositional Logic using Coq.

Supervisor

Rajeev Gore.

Objective

It is essential to prove or disprove that a formula is a theorem in many areas of applied logic. This is done through automatic reasoning, theorem provers. One such versatile and successful type of theorem provers are tableau-based theorem provers. These are theorem prover that utilise the semantic tableau, a procedure which can determine if a formula is satisfiable, and hence if the formula is a theorem. We aim to build a tool that given a set of tableau rules, among other parameters, it will automatically generate a corresponding tableau-based theorem provers which is proven to be correct. We aim to do this for classical propositional logic first in the hopes that this can be extend for more complex logical systems in the future.

Current Limitations

Currently, the Tableau Work Bench [1], implemented in O'Caml, exists as a "user-friendly framework for building automated tableau-based theorem provers". It allows users to give an input set of rules for their custom logical system in propositional logic to generate a corresponding tableau-based theorem prover. However, the generated tableau-based theorem provers generated are not guaranteed correctness.

Apart from the Tableau Work Bench, to implement a theorem prover for a custom logic system, assuming there is no pre-existing one already, one would have to implement and program the theorem prover from scratch.

Furthermore, if one wants to prove that a logical set of rules corresponds to a particular logical system they would have to prove this separately from their corresponding implementation of a theorem prover. This is a major issue on knowing if the implementation of the theorem prover correctly corresponds to the logical system the user wants it to describe.

Approach

Instead of implementing a framework for building automated tableau-based theorem provers in O'Caml like the Tableau Work Bench, we instead implement a similar system in Coq, a theorem prover itself. By using Coq to implement the tool, we are able to prove properties about the process of proving or disproving if a formula is a theorem.

This allows users to complete a completeness and soundness proof of their logical rules (with respect to what they want these rules to describe) inside Coq (proving the rules in-fact describe what they believe they do). This approach gives us a tool which is capable of accepting an input to describe a custom logical calculi, generating a corresponding tableau-based theorem prover, and allows the user to be sure that the generated tableau-based theorem prover does in fact correspond to their described logical calculi through proof.

Motivation

The reason we want a tool to build automated tableau-based theorem provers is so that researchers who are interested in experiment with automated theorem provers for calculi they are designing do not have to program their own theorem prover. This allows users with limited programming experience to create theorem provers for their calcui. By formally verifying the process of generating the tableau-based theorem provers, the users of the tool are guaranteed that the produced theorem prover expresses the calculi they described.

Currently, we require the user to prove that their logical rules are sound and complete within Coq to ensure that the generated theorem prover expresses the calculi the user describes (otherwise it generates a theorem prover with no guarantee). However, as we have implemented the generation of the theorem prover in Coq, the user can prove the properties of soundness and completeness of theorem in Coq. Thus, we couple the proof of correctness of the logical system with the creation of the corresponding theorem prover

Coupling proof and implementation of a theorem prover is a step up from current practices of trusting implementations, where the user need to check that the theorem prover is correct (automatically done by us once the soundness and completeness is proven). This may seem like additional work, that is to prove soundness and completeness, but for those who want to test custom logical systems, these proofs are necessary for their logical system to be useful.

Milestones

Milestone	Date of Completion\to Complete
Background Learning: How to use Coq	10 July, Semester Break
Background Learning: How a Tableau Proof Works	17 July, Semester Break
Background Learning: How Sequent Calculus Works	24 July, Week 1
Encode Sequent Calculus and Tableau Calculus in Coq	31 July, Week 2
Prove Equivalence of Sequent Calculus and Tableau Calculus in Coq	07 August, Week 3
Prove Admissibility of Structure Rules in Sequent Calculus	14 August, Week 4
Implement Partition Application of Tableau Rules in Coq	21 August, Week 5
Implement Rule Application of a Tableau Node in Coq	28 August, Week 6
COMP2560: Project Summary	01 September, Week 6
Implement Tactic Language for Rule Application in Coq	04 September, Mid-Semester Break
Implement Tree Search and Prove Correctness in Coq	11 September, Mid-Semester Break
COMP2560: Paper Draft	10 October, Week 10
COMP2560: Paper Reviews	17 October, Week 11
COMP2560: Poster, Rebuttal	24 October, Week 12
COMP2560: Presentation	27 October, Week 12
COMP2560: Final Paper	9 November, Exam Period

Bibliography

[1] Pietro Abate and Rajeev Gor. The tableau workbench. *Electronic Notes in Theoretical Computer Science*, 231:55 – 67, 2009. Proceedings of the 5th Workshop on Methods for Modalities (M4M5 2007).