Automated theorem proving for the two variable fragment in the First Order Logic

Author: Patrick Cătălin Alexandru Sava

Supervisor: Dr. Ian Pratt-Hartmann

Abstract

First-order logic has few fragments which are very interesting in terms of decidability. One of them is the two-variable fragment with no function symbols. Even though the two-variable fragment could be very rich in terms of potential discoveries, there are few papers written in this regard by now. One of them is Hans de Nivelle and Ian Pratt-Hartmann’s paper of 2001 which, by no coincidence, represents namely the goal of the project. Being the first of its kind, this paper describes a decision procedure for the two-variable fragment with equality (prior works describe only the decision procedure for the case without equality). Implementing this algorithm embodies the project’s aim, nonetheless, the roadmap for achieving this objective involved moreover the implementation of a theorem prover for the general case.

Acknowledgements

Firstly, the project would not have been possible without the continuous help and support of my supervisor, Dr. Ian Pratt-Hartmann. I am very grateful for the satisfying experience of working together on such a challenging, yet rewarding project. I would like to highlight my appreciation for all of the excellent advice I was receiving from him throughout the entire project.

Secondly, there is no doubt that my self-development, knowledge and capabilities were constantly enhanced by the world of Computer Science which has been an indispensable part of my life ever since. Thus, I would like to extend my gratitude to the programming community, my teachers and friends who have been a part of this great journey.

Finally, I would like to thank my family (especially my mother, my grandmother and my partner, Yoana) for all of their trust, support and love during all of these university years and not only.

[**Introduction**](#_df8vv4rcxc1p) **6**

[Background](#_orz03eqchgg4) 6

[Motivation](#_8k4aygsvrx9q) 6

[Project aim](#_ax4updv1mq93) 6

[Project Plan](#_r2ujw6ayztbf) 6

[Methodology](#_oizawpbvlrdq) 6

[Report Structure](#_2oarfns66zbh) 6

[Impact of Covid-19](#_xwhv18ieh44j) 6

[**Context**](#_9zd57b4duhd5) **6**

[First-Order Logic](#_dy6konftaet3) 7

[Definitions](#_djdrgmdkxmng) 7

[Operators](#_2vspvqs2p7le) 7

[Special case: equality](#_wnin7kxxh15l) 7

[Assumptions](#_3641xxj7pn58) 7

[Clause Normal Form (CNF)](#_r0vwamua472s) 7

[Automated Theorem Proving](#_f3q4g59ixn98) 7

[Unification](#_c8vaumr6n4pg) 7

[Overview](#_w55scof3yhil) 7

[Most general unifier](#_lsd5mg6w3z1g) 7

[Resolution rule](#_aqodhlrppqkq) 7

[Factoring rule](#_4p8p51x1sbok) 7

[Refinements](#_ifehuyerqdp6) 7

[Tautology Removal](#_2uipkhcq6lh9) 7

[Subsumption](#_1tbv1tqm568p) 7

[Unification within the same clause, on literals having the same sign](#_77q9rj31lmho) 7

[Depth-ordered resolution](#_e7rwvwdaw2yp) 7

[**Design**](#_xxpev9n2lwjw) **7**

[**Implementation**](#_gfjtingyzhcz) **8**

[Environment](#_l1vcfc3nbwqf) 8

[Front-end](#_d5si54q21z42) 8

[Tokenizer](#_cn3wt633rot0) 8

[Parse Tree](#_l6i6odkebjow) 8

[Intermediate Representation](#_e3c2dgf365go) 8

[The approach for disambiguating the given formulas](#_vrbua8pgemx8) 8

[The approach for resolving the precedence](#_oja1m8ihg9eo) 8

[The approach for eliminating double Implications](#_ebd5oqbkz1ac) 8

[The approach for eliminating implications](#_hfjgzsolr963) 8

[Basic reduction](#_9xc26xr5lcjq) 8

[Skolemization](#_bxgow04gybs6) 8

[Back-end](#_61zrhg2p0mft) 8

[Clause Form](#_apo4ra7ygpa6) 8

[Basic Theorem Prover](#_5shpuwpg832v) 8

[Overview](#_gjbi0aoldqn4) 8

[Multithreading](#_s1hwhpfhlpg6) 8

[Depth-Ordered Theorem Prover](#_kqflx93o42gg) 8

[Two-Variable Theorem Prover](#_2qdj0scw4dg3) 8

[Without Equality](#_wmmcmnyjbk25) 8

[With Equality](#_e5cregib9bw3) 9

[**Evaluation**](#_wb83b0x8anmn) **9**

[Unit testing](#_xmp6tw7l5j9v) 9

[End-to-End testing](#_b1rwdin7qwlu) 9

[Using the problems found on tptp.org](#_mlszyojem44) 9

[Using general formulas](#_2bj5s1nzowqv) 9

[Testing against Vampire](#_gs6mdb6o2163) 9

[**Experiments**](#_w8eibc75csft) **9**

[**Reflection**](#_vct7big78d3b) **9**

[**Conclusion**](#_l2sgo3qyqxj8) **9**

# 

# 

# 

# 

# 

# 

# 

# Introduction

## Background

First-order logic is characterized by a strong richness in terms of expressivity. Currently the world is aware of its presence behind many fields such as Mathematics, Philosophy or Computer Science. In Computer Science it could be present as part of sophisticated solutions in the subfield of databases or the one of natural language processing.

In comparison with propositional logic, the first-order logic has a stronger power due to the predicates and the quantifiers. Symmetrically, higher-order logic has a stronger power than the first-order logic, but it does not provide a sound and complete proof calculus. [reference <https://en.wikipedia.org/wiki/Higher-order_logic>]

In essence, first-order logic lies at the intersection of high expressiveness with soundness and completeness and this is one of the reasons why it is valued by the research world.

## Motivation

During the course COMP24412 I became particularly attracted to using Prolog and Vampire. I have then realised that this is an area of research which I have not had the chance to explore well enough so I was looking for a third-year project liying in this sub-field.

At the same time, I was looking to grasp my knowledge in the sub-field of compilers, which the implementation of a theorem-prover was able to offer to me since I strongly preferred to write the front-end of it by myself.

Ultimately, the curiosity of seeing how well I could implement a piece of software which is going to compete directly with an absolut champion like Vampire, encouraged me to accept this challenge.

## Project aim

The main goal of the project was to implement the first ever decision procedure for the two-variable fragment with equality of the first order-logic which was priorly described in a paper written by Hans de Nivelle and Ian Pratt-Hartmann in 2001, but for which there is no evidence currently that it was actually implemented by someone else in the meantime. On the top of that, an additional goal of the project was to compare the algorithm with Vampire and to produce some satisfiability charts in the case in which the input is a randomly generated formula.

## Project Roadmap

The project roadmap was quite well defined:

* Semester one
  + The first six weeks (including week 0) were dedicated to familiarising myself more to the topic, that meaning mostly reading papers, books and additional materials
  + The next three weeks (up to week 8) were mostly invested in writing the parser for the formulas
  + The last weeks up to the Christmas holiday were spent coding a general theorem prover
  + The Christmas holiday and the exam session were dedicated for testing the existing functionality, coding the depth-ordered theorem prover and the two-variable theorem prover for the case without equality
* Semester two
  + The first three weeks were spent on reading some papers, coding the two-variable theorem prover for the case with equality, optimizing, testing and debugging the code
  + The next three to four weeks were mostly invested in optimizing, testing and debugging the code. On the top of that, two random generators for formulas and some scripts of comparing my work against Vampire were written.

## Methodology

One of the most important aspects of the project was definitely the time management. Coordinating with my supervisor while both researching and implementing the theorem prover was challenging and required a good amount of prioritization. My supervisor suggested keeping track of everything I am doing in a log book, which I found to be very useful not only for writing now this report, but also for organizing my thoughts and ideas better. Even though I did not feel it as being useful at all at the beginning, this idea proved as being brilliant in time, especially when the size of the codebase for the project hit 5000 lines of code and even remembering what my own lines of code are doing was quite problematic.

## Report Structure

The structure of the report is chosen in a way in which the reader is presented the context first, then the high-level idea of the project and further details of smaller granularity. Hence, the next section is reserved for reiterating the basic notions of first-order logic and the formal tools which are going to be part of the algorithm. The next section is going to be dedicated to the design decisions, included but not limited to the high-level presentation of the two-variable theorem prover. It follows the section dedicated to the implementation, which indeed underlines the low-level details of it. Further there is a section dedicated to the methods of evaluating the project, one dedicated to the experiments, one dedicated to the reflection and finally the one which concludes the report.

## Impact of Covid-19

Fortunately, I am in a position in which I could say that the whole situation caused by Covid-19 did not create any major disruptions, and that’s mostly because my project was completely independent of any University equipment. The communication with my supervisor was excellent and I feel grateful that the overall quality of the project was not affected by this matter.

# Context

Highlight the assumptions related to the knowledge of the reader here!

## First-Order Logic

### Definitions

### Operators

#### Special case: equality

### Assumptions

## Clause Normal Form (CNF)

## Automated Theorem Proving

### Unification

#### Overview

#### Most general unifier

### Resolution rule

### Factoring rule

### Refinements

#### Tautology Removal

#### Subsumption

#### Unification within the same clause, on literals having the same sign

#### Depth-ordered resolution

# Design

# Implementation

## Environment

## Front-end

### Tokenizer

### Parse Tree

## Intermediate Representation

### The approach for disambiguating the given formulas

### The approach for resolving the precedence

### The approach for eliminating double Implications

### The approach for eliminating implications

### Basic reduction

### Skolemization

## Back-end

### Clause Form

### Basic Theorem Prover

#### Overview

#### Multithreading

### Depth-Ordered Theorem Prover

### Two-Variable Theorem Prover

#### Without Equality

#### With Equality

# Evaluation

## Unit testing

## End-to-End testing

### Using the problems found on tptp.org

### Using general formulas

## Testing against Vampire

# Experiments

# Reflection

# Conclusion