Comparison of theorem provers

Artem Yushkovskiy
artem.yushkovskiy@aalto.fi

Tutor: Stavros Tripakis

Abstract

One of the useful applications mathematical logic theory is the Automated theorem proving. This is a set of techniques that allow one to verify mathematical statements mechanically using logical reasoning. Although, it can be also used to solve engineering problems, for instance to prove security properties for a software system or an algorithm. Furthermore, automated theorem proving is an essential part of the Artificial Intelligence theory, which became highly evolving these days. In this paper, we describe bases of formal systems and automated deduction theory, and compare two widespread tools for automated theorem proving, Coq [1] and Isabelle [2].

KEYWORDS: logic, automated theorem prover, Coq, Isabelle

1 Introduction

In general terms, *formal proof* is the sequence of statements, based on finite set of fundamental axioms and satisfying the rules of logical inference. *The axiom* is a statement claimed to be true evidently. *The logical inference* is the transfer from one statement (premise) to another (consequence), which preserve truth, while the rule of logical inference is a principle that allows one to infer the validity of such transfer. In formal logic, inference is based entirely on the structure (i.e., form) of those statements, which allows

one to apply basic logical rules to any type of proof and thus construct the formal system.

The main goal of the formal system is to be verifiable, i.e. one could *check* its validity. At present, a lot of tools are being developing to automate the process of such checking to run it on the computer. In particular, the systems *Isabelle/ZF* [???], *Coq* [???], PVS [???], ACL [???] work in a form of axiomatic set theory and allow the user to enter theorems and proofs into the computer, which then verifies that the proof is correct (these are also called sometimes "proof assistants"). Another goal of constructing the formal system is having the computer to *discover* formal proof. This goal is different from the previous one since the system must be optimised for efficient search. The output proofs can rely on induction, or on meta argument, or on higher-order logic. McCune's systems *Otter* [???] and *Prover9* [???] are commonly recognized as the state-of-the-art tools [Com00].

In current paper we consider only the systems, which are built to achieve the first goal, i.e. to verify existing proof, since <...>. Two aforementioned theorem provers Coq and Isabelle are examined for the purpose of revealing expressiveness, computation power and usability. These properties are described in detail in section <???>. Section 2 gives an overview of the history of logic providing thorough definitions, typology and properties of formal system and formal proof. Issues related to theoretical limitations of formal systems are discussed further as well. <... about comparison, results and author's personal contrubution>

2 Theory of logical calculi

// TODO

Basics: set theory, its problems (paradoxes), Zermelo–Fraenkel, propositional and 1st ordered logic (all with unified formal notation), formal languages, ...

MOCK from plato: A third important consideration in the building of an automated reasoning program is the selection of the actual deduction calculus that will be used by the program to perform its inferences. As indicated before, the choice is highly dependent on the nature of the problem domain and there is a fair range of options available: General-purpose theorem proving and problem solving (first-order logic, simple type theory), program verification (first-order logic), distributed and concurrent systems (modal and temporal logics), program specification (intuitionistic logic), hardware verification (higher-order logic), logic programming (Horn logic), and so on.

MOCK from wiki: A formal system or logical calculus is any well-defined system of abstract thought based on the model of mathematics. A formal system need not be mathematical as such; for example, Spinoza's Ethics imitates the form of Euclid's Elements. Spinoza employed Euclidiean elements such as "axioms" or "primitive truths", rules of inferences etc. so that a calculus can be built using these. For nature of such primitive truths, one

can consult Tarski's "Concept of truth for a formalized language".

Foundational crisis in mathematics (Intuitionism, Logicism, Formalism,) Hilbert's program

2.1 Basic properties of logical system

// TODO

According to Formalists: Independence, Consistency, Completeness, Decideability

Formalists defined four basic properties which every logical system must have: 1. Independence, which means that there aren't any superfluous axioms. There's no axiom that can be derived from the other axioms. 2. Consistency, which means that no theorem of the system contradicts another. 3. Completeness, which means the ability to derive all true formulae from the axioms. 4. Decideability, the Entscheidungsproblem, which asks for an algorithm that takes as input a statement and answers "Yes" or "No" according to whether the statement is universally valid, i.e., valid in every structure satisfying the axioms.

2.2 Limitations of logic systems

// TODO

completenes, Gödel's incompleteness theorems

3 Methods for automated reasoning

// TODO

techniques in common words (and in introduced previously notation): clause rewriting, - Resolution - Sequent Deduction - Natural Deduction - The Matrix Connection Method - Term Rewriting (+lambda calculus) - Mathematical Induction

4 Some automated theorem provers

// TODO

in two words: here we consider Coq and Isabelle, bla-bla

4.1 The Coq theorem prover

Describe Coq features

4.2 The Isabelle theorem prover

Describe Isabelle features

5 Comparison of selected theorem provers

// TODO

in table:

- expressiveness of logic used - time of proving - num of supporting theories - set of techniques to prove automatically - V of theorem code - num of user interaction steps - usability - etc ...

6 Results

// TODO

Conclusions of comparison

7 Future work

// TODO

to apply this survey to software verification

References

[1] Douglas E. Comer. *Internetworking with TCP/IP, Volume I*. Prentice-Hall Inc, 4th edition, 2000.

[2] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). RFC 2408, The Internet Engineering Task Force, November 1998. http://ietf.org/rfc/rfc2408.txt.

[3] Pekka Nikander. *An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems*. PhD thesis, Helsinki University of Technology, March 1999.

[4] Sanna Suoranta. An Object-Oriented Implementation of an Authentication Protocol. Master's thesis, Helsinki University of Technology, November 1998.

[5] Tim Hsin-ting Hu and Binh Thai and Aruna Seneviratne. Supporting Mobile Devices in Gnutella File Sharing Network with Mobile Agents. In *IEEE International Symposium on Computers and Communication*, volume 3, pages 1530–1346, April 2003.