Comparison of theorem provers

Artem Yushkovskiy
artem.yushkovskiy@aalto.fi

Tutor: Stavros Tripakis

Abstract

One of the useful applications of mathematical logic theory is Automated theorem proving. This is a set of techniques that allow one to verify mathematical statements mechanically using logical reasoning. Although, it can be used to solve engineering problems as well, 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 is highly evolving these days. In this paper, the bases of formal systems and automated deduction theory are described. Then, two widespread tools for automated theorem proving, Coq [1] and Isabelle [2], are compared with respect to expressive power and usability.

KEYWORDS: logic, formal method, proof theory, automated theorem prover, Coq, Isabelle.

1 Introduction

Modern world cannot be imagined without mathematics. Almost every human technical achievement is based on the groundwork of mathematical methods. Such formal methods allow one both to describe existent phenomena and to develop new tools in a verifiable way, using strict logical *proof*.

Informally, *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* [2], *Coq* [1], *PVS* [6], *ACL2* [7] 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 automatically *discover* the formal proof, which can rely either on induction, or on meta argument, or on higher-order logic. McCune's systems *Otter* [8] and *Prover9* [9] are commonly recognized as the state-of-the-art tools.

In current paper we consider only the systems, which are built to achieve the first goal, i.e. to verify existing proof, since they can be sufficiently applied in software verification, the research area in which both authors are being involved. 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 6. In Section 2, one can find an overview of the history of logic. Then, Section 3 gives thorough definitions, typology and basic properties of formal system and formal proof; issues related to theoretical limitations of formal systems are discussed there as well. General methods for automated reasoning are given in Section 4. Possible applications areas for automated theorem provers are enumerated in Section 5. The comparison of selected theorem provers is given in Section 6.

<... results and author's personal contribution >

2 History of formal approach

Although in most cases our formal models work efficiently, in previous century mathematics experienced deep fundamental crisis caused by the need for a formal definition of the very basis of mathematics. That time many paradoxes in some mathematics have been discovered, so, in 1920s, the three main confrontation schools appeared: the *logicism* based on work of Gottlob Frege, Bertrand Russell, and Alfred Whitehead, which stated that mathematics is an extension of logic and therefore it can be reduced to logic partly or fully; the *formalism* led by David Hilbert, also called the 'proof theory', which advocated classical mathematics with its axiomatic approach; and the radical *intuitionism* led by Luitzen Brouwer, which rejected this formalisation as unnecessary and even meaningless, claiming

that only the objects, which we know how to construct, may exist. Although the formalism was the leading school, all these schools have contributed important ideas and techniques to mathematics, philosophy and many other scientific and engineering areas [3].

One solution for the foundational crisis was proposed by the school of formalism, it is known as Hilbert's program, and aimed to base all existing theories on finite set of axioms, and prove that these sets are consistent [4]. Thus Hilbert proposed to reduce the consistency of all mathematics to basic arithmetic. Unfortunately, these intentions turned out to be rather unrealisable, when in 1931 Kurt Gödel published his two famous incompleteness theorems, that demonstrated the limitations of any formal axiomatic system containing basic arithmetic [5]. In particularly, he proved that all consistent axiomatic formulations of number theory are incomplete (such that there will always be statements which cannot be proved or disproved within that formulation), and that such formal system can not prove that itself it is consistent (i.e., it is unable to prove that a statement and its negative are both true). Notwithstanding the fact that the whole mathematics can not be described in a single axiomatic system, the formal approach allows one to build though restricted, but fine, concise and verifiable theories.

3 Logical foundations

// TODO (Paragraph is still in progress)

- what the logical system is (formally: set of axioms, inference rules)
- (?) a brief history of axiomatic approach (Euclid, Hilbert)
- what the truth is: completeness
- intro to set theory (for notation) +Zermelo–Fraenkel set theory (ZFC)
- intro to formal languages
- intro to propositional and 1st ordered logic (introduce notation here. Maybe Higher-Order Logic, Non-classical Logics)
- type systems (+dependent type, where type checking although may be undecidable, but it verifies the correctness of), Nominal vs. structural type system. **Curry–Howard correspondence** (the direct relationship between computer programs and mathematical proofs)
- else?

According to Formalists: Independence, Consistency, Completeness, Decidability.
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. Decidability, 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.

Formula (i.e. statement, program) may be:

- provable (either true or false, according to selected set of axioms)
- valid
- sound
- ...

4 Methods for automated reasoning

// TODO (Paragraph is still in progress)
techniques in common words (and in introduced previously notation), e.g.:

- Clause rewriting
- Resolution
- Sequent Deduction
- Natural Deduction
- The Matrix Connection Method
- Term Rewriting (+lambda calculus)
- Mathematical Induction

5 Some applications of theorem provers

// TODO (Paragraph is still in progress)
Describe possible applications of formal methods:

1. Interactive theorem proving: construct a formal axiomatic proof of correctness,
2. verifying that a mathematical statement is true.
3. verifying that a circuit description, an algorithm, or a network or security protocol meets its specification:
    - program verification (first-order logic),

- distributed and concurrent systems (modal and temporal logics),
- program specification (intuitionistic logic),
- Model checking: reduce to a finite state space, and test exhaustively.
- hardware verification (higher-order logic),
- logic programming (Horn logic),
- and so on.

6 Comparison of some theorem provers

//TODO (Paragraph is still in progress)

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

6.1 The Coq theorem prover

// TODO (Paragraph is still in progress)

Coq is a formal proof assistant system. It provides a formal language to write mathematical definitions, executable algorithms and theorems together with an environment for semi-interactive development of machine-checked proofs [1]. Coq uses the Calculus of Construction, a higher-order formalism for constructive proofs in natural deduction style, developed by Thierry Coquand [10]. The Calculus of Construction can be considered as an extension of the Curry–Howard isomorphism.

(add some core features of Coq ...)

(Coq has been used to formalize ...)

6.2 The Isabelle theorem prover

// TODO (Paragraph is still in progress)

The Isabelle is an interactive theorem prover, which relies on higher-order logic. It allows mathematical formulas to be expressed in a formal language and provides tools for proving those formulas in a logical calculus [2]. Isabelle's main proof method is a higher-order version of resolution, based on higher-order unification.

(add some core features of Isabelle ...)

(Isabelle has been used to formalize ...)

6.3 Joint comparison

//TODO: in table:

- expressiveness of logic used

- time of proving
- num of supporting theories
- set of techniques to prove automatically
- Volume of proof (as text)
- num of user interaction steps
- usability
- etc ...

7 Results

// TODO (Paragraph is still in progress) conclusion of comparison

8 Future work

// TODO (Paragraph is still in progress)
< in future, we want to apply this survey to software verification >

References

[1] "Coq proof assistant." https://coq.inria.fr/.

[2] "Isabelle, a generic proof assistant." https://www.cl.cam.ac.uk/research/hvg/Isabelle/.

[3] J. Ferreirós, "The crisis in the foundations of mathematics," 2008.

[4] R. Zach, "Hilbert's program then and now," in *Philosophy of Logic. Handbook of the Philosophy of Science*, vol. 5, p. 411–447, Amsterdam: Elsevier, 2006.

[5] P. Raatikainen, "Gödel's incompleteness theorems," 2015.

[6] "Pvs specification and verification system." http://pvs.csl.sri.com/.

[7] "Acl2: a computational logic for applicative common lisp." http://www.cs.utexas.edu/users/moore/acl2/.

[8] W. McCune, "Otter and mace2," 2003. https://www.cs.unm.edu/~mccune/otter/.

[9] W. McCune, "Prover9 and mace4."

[10] G. H. T. Coquard, "The calculus of constructions," 1986.