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 in recent years. This paper compares two widespread tools for automated theorem proving, Coq [1] and Isabelle [2], with respect to the power of expressiveness and usability. For this reason, it firstly gives a brief introduction to the bases of formal systems and automated deduction theory, its main problems and challenges.

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

1 Introduction

Nowadays, the search for foundations of mathematics has become one of the key questions in philosophy of mathematics, which eventually have an impact on numerous problems in modern life. Basically, this search has led to the development of *formal approach*, a methodology for manipulating the abstract essences according basic rules in a verifiable way. In other words, it is possible to follow the sequence of such manipulations in order to

check the validity of each statement and, as a result, of a system at whole. Moreover, automating such a verification process can significantly increase reliability of formal models and systems based on them.

At present, a large number of tools have been developed to automate this process. In particular, the systems *Isabelle/ZF* [2], *Coq* [1], *PVS* [3], *ACL2* [4], sometimes called *proof assistants* or *automated proof checkers*, validate the input statement (used-defined theorem) with respect to the sequence of inference transitions (user-defined proof) according to set of rules. Another goal of constructing the formal system is having the computer to automatically *discover* the formal proof, which can rely either on induction, on meta argument, or on higher-order logic. McCune's systems *Otter* [5] and *Prover9* [6] are commonly recognized as the state-of-the-art tools. In this paper only the systems that aim the first goal were considered, since they can be sufficiently applied in model checking and software verification.

This paper is organised as follows. Sections 2 and 3 provide a brief history of logic, thorough definitions, basic properties and theoretical limitations of the formal systems. Section 4 covers general methods for automated reasoning. Section 5 enumerates possible application areas for automated theorem provers. Finally, Section 6 presents the comparison of two aforementioned theorem provers, Coq and Isabelle, with respect to criteria s.a. expressiveness, computation power and usability.

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

2 History of formal approach

The formal approach appeared in the beginning of previous century when mathematics experienced deep fundamental crisis caused by the need for a formal definition of the very basis. At that time, multiple paradoxes in several fields of mathematics have been discovered. Moreover, the completely new theories appeared just by modification of the set of axioms, e.g., reducing the parallel postulate of Euclidean geometry has lead to completely different non-Euclidian geometries s.a. Lobachevsky's hyperbolic geometry or Riemman's elliptic geometry, that eventually have a large number of applications in both natural sciences and engineering.

In general, a *formal proof* is the sequence of statements, based on a finite set of fundamental axioms and satisfying the rules of logical inference. *An axiom* is a statement evidently claimed to be true. *A logical inference* is a 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, thereby, the result formal system represents the abstract model describing part of real world.

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)

The Coq is a computer tool for verifying theorem proofs. These theorems may concern usual mathematics, proof theory, or program verification

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

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

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

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

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

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