Automated Theorem Provers: an introduction

Alessio Catanzaro

April 2019

1 Proving theorems

According to most mathematicians, proving theorems is an art. It requires a deep and complete knowledge of the field of investigation, a sound use of logic and a great deal of rigour. Yet these conditions are not sufficient as in most cases an intuition of the thesis is needed. Such fact seem to be a paradox, that in a rigorous logical framework sometimes non-trivial propositions are not proven to be true by constructive methods[1], but they are rather proven by intuition that the result is true[2].

Nevertheless the quite opposite idea that a complete mathematical theory could be entirely deducible step-by-step from a set of well chosen axioms dates back to the what we consider the beginning of logically sound mathematics, *i.e.* Euclid's *Elements of geometry*. In such work indeed, after giving basic definitions, postulates (axioms) and "common notions", a set of claims is made and proven, and each statement of the proof is logically justified by making reference to a previous proposition or postulate. This way of approaching maths one step at a time, *i.e.* making chains of logically connected propositions, each of them being trivially linked to the previous one, is extremely similar to one of the two mainstream mathematical school of thought of the 20^{th} century, that is to say Burbakism.

Such extremely influential school of reasoning and teaching may had been one of the drives for which, in the context of the quest for a logically sound base for the whole of mathematics during the first half of the 20^{th} century, the idea that proving propositions may be somehow a deterministic process revived, giving rise to what, in the modern language, is known as Automated Theorem Proving (ATP).

[1] By "Constructive methods" I mean that the proof actually realizes what the proposition claims (*e.g.* if it says that there exist a certain number it provides the way in which it may be computed), while non constructive methods are those that do not care about explicating objects but only giving proof of its existence.

[2] In this case, existence theorems may be proven by just giving an example of an object that satisfies the clauses of the thesis.

Automated Theorem Proving is the idea that given a logic system that comprehends suitable language and axioms, with the addition of rules of inference to be applied to the latter, there exist a mechanical way by which all propositions that are true in such system may be proven. In the words of David Duffy [4]:

> "Automated Theorem Proving is concerned with the task of mechanizing mathematical (or logical) reasoning. Proofs of mathematical theorems are performed by a computer, analogously to the way arithmetical problems are solved by a calculator."

The analogy between arithmetical calculators and theorem provers is what has been leading the research in the field of ATP since the beginning. In the following I will briefly sketch the state of the art basics of this field.

2 ATP fundamentals

2.1 First Order Logic

First order logic is the mainstream way to formalize some of the seminal results of modern mathematics. Among them Peano arithmetic and Zermelo-Frenkel set theory are to be found. It is also the main foundation of ATP, hence a brief introduction is necessary.

A First Order Logic is a logic system that allows to manipulate propositions that involve quantifiers and predicates concerning the variables of a taken set.

It consists of

- A set A of symbols (the alphabet).
- A set of Well-Formed Formulae (*wff*), made from S using given formation rules.
- A Proof Theory, which allows us to define the provable *wff*. It is made of a finite set of *axioms* and a finite set of relations on *wff* called *Inference Rules*. *Inference rules* are functions that map any set of *wff* into a single *wff*.

In such context, a *proof* is a sequence of *wff* such that each *wff* may be inferred from the previous one or is an axiom. A theorem is a *wff* for which there exists a proof.

2.1.1 The Alphabet

The elements that compose the alphabet may be classified into several categories

- Constant symbols (a, b, ...)
- Variable symbols (x, y, ...)

- Symbols described as results of a function (f(a), g(x), ...)
- Symbols that represents predicates (P, T, ...)
- Connectors ($\rightarrow$, $\wedge$, ...)
- Punctuation symbols ("(", ",", ...)
- Quantifiers ($\forall$, ...)

2.1.2 Formation Rules

A *term* is defined as a constant or variable symbol, or as the result of a function evaluation. An *atomic formula* is either a *term* or a predicate. Thanks to these atoms we may give a definition of all the *wff*, indeed, we say that

- Every *atomic formula* is a *wff*
- if F and S are *wff* then also

  $(F \rightarrow S) \quad (F \wedge S) \quad (F \vee S) \quad (\neg F) \quad (\neg S) \quad (\forall x F) \quad (\forall x S) \quad (\exists x F) \quad (\exists x S)$

  are well formed formulae.

2.1.3 Axioms

The axioms of first order logic may be divided into two kind: those which entail the logical structure of the system (logical axioms) and those which depend on the structure that may be given in addition (proper axioms). The latter may be absent from a FOL, in such case it is called a First Order *Predicate calculus*. Given that F, S and O are *wff* the logical axioms that any FOL has are

$$F \rightarrow (S \rightarrow F) \tag{1}$$

$$(F \rightarrow (S \rightarrow O)) \rightarrow ((F \rightarrow S) \rightarrow (F \rightarrow O)) \tag{2}$$

$$(\neg S \rightarrow \neg F) \rightarrow ((\neg S \rightarrow F) \rightarrow S) \tag{3}$$

$$(\forall x_i) F(x_i) \rightarrow F(t) \text{ if } F(x_i) \text{ is a } \textit{wff} \text{ with } t \text{ free term.} \tag{4}$$

$$(\forall x_i)(F \rightarrow S) \rightarrow (F \rightarrow (\forall x_i) S) \tag{5}$$

The (main) inference rules that apply to FOL are listed in figure 1.

2.2 ATP implementations basics

First Order Logic systems set a clear framework of reasoning. Indeed, the core idea behind the Automatic Theorem Proving is precisely to apply mechanically the inference rules to axioms to derive conclusions, the to do so with the obtained propositions and so on, until a given theorem is proven[3]. Such "provers" are called Stand-Alone Provers, as they require no external input to produce a result.

However, some issues do rise quite early following such an approach. Indeed the machine by inferring propositions indistinctly from axioms is widely exploring the whole space of possible theorems. If interested in a specific theorem to be proven, a long (possibly infinite) time will be taken to prove it.

In order to overcome such an issue some strategies may be taken. The first one involves human interaction (it is the case of Interactive Provers). As the inferring process goes one, it periodically ask for an input from a user to guide it to what may seem a short path of proving the theorem.

Another strategy to overcome the issue is "weighting" the path prior to the execution, in order to give directions for the machine to follow. By giving symbols a weight to be taken into account when computed the space of all theorems is not explored uniformly as would happen if all the weights were equals, but the search is steered towards a locus in which, hopefully, the theorem which we look for is. Needless to say that such a method works only if the weight are assigned correctly, but their correctness is hard to evaluate *a priori*.

Both these fixes however are human-involving, in the sense that the intervention of a human user is requested, prior or *in itinere* to the process, to speed up the proof making. Such fact is not to be neglected, and I will make some comments about that in the conclusion.

2.2.1 ATP main elements

Automatic Theorem Provers vary in their structure and features, however, the core elements are somehow common to all of them.

The first task to comply with is to find a way in which the information is to be represented into the language. Indeed, the language we are using must be capable of dealing with inputs and the problem we are taking into consideration. Common implementations of this are the "Clause language", in which well formed formulae are translated into clauses; the "Non Clausal language" [6]; the "General Mating" method [1]; the "Natural Deduction" where the well formed formulae are dealt with in their natural form.

Then the machine needs a way to implement the inference rules to draw conclusions from the axioms. One of the main issues in ATP is indeed the fact that inference rules should make deduction step not to be too small. In the figure 1 taken from [7] the main inference rules are listed.

[3]That is a chain of logically connected *wff* that begins with axioms and ends with the theorem.

TABLE 1 Rules of Inference.

Rule of Inference	Tautology	Name
p $p \rightarrow q$ $\therefore q$	$(p \wedge (p \rightarrow q)) \rightarrow q$	Modus ponens
$\neg q$ $p \rightarrow q$ $\therefore \neg p$	$(\neg q \wedge (p \rightarrow q)) \rightarrow \neg p$	Modus tollens
$p \rightarrow q$ $q \rightarrow r$ $\therefore p \rightarrow r$	$((p \rightarrow q) \wedge (q \rightarrow r)) \rightarrow (p \rightarrow r)$	Hypothetical syllogism
$p \vee q$ $\neg p$ $\therefore q$	$((p \vee q) \wedge \neg p) \rightarrow q$	Disjunctive syllogism
p $\therefore p \vee q$	$p \rightarrow (p \vee q)$	Addition
$p \wedge q$ $\therefore p$	$(p \wedge q) \rightarrow p$	Simplification
p q $\therefore p \wedge q$	$((p) \wedge (q)) \rightarrow (p \wedge q)$	Conjunction
$p \vee q$ $\neg p \vee r$ $\therefore q \vee r$	$((p \vee q) \wedge (\neg p \vee r)) \rightarrow (q \vee r)$	Resolution

Figure 1: Main inference rules of logic.

Implementing language and establishing inference rules are however related to the formal structure of the logical system. The inner core of of an ATP are the *Search Strategies*, *viz*. the way in which it explores the space, and the *Proving methodologies*, that is the way in which a theorem is proved. Just like in mathematics, there are two ways of doing so

- Direct Proof. This is the straight-forward case where a theorem may be proved by mechanically applying inference rules to axioms and deriving the chain that links the theorem to the postulates. Such chaining can be forward

or backward, *i.e.* it may start from the axioms and derive propositions until the theorem is reached or it may reduce the theorem to a set of simpler well formed formulae until the axioms are reached.

- Refutation approach. As in maths we have proof *ex absurdo*, the prover does not look for a proof of the theorem, it rather adds the negation of the theorem to the axioms and runs the inferring until it reaches a contradiction[4]. If the negation of a theorem is not consistent with the axioms, it follows that the theorem is to be true. An example of Theorem Prover based on refutation is PROLOG.

3 Final remarks

The fact that a brute force approach to axioms to derive proofs of theorems would be eventually effective in a arbitrarily long time is beyond question. However it may be worthwhile asking why this is not the way most theorem were discovered by mathematicians. Moreover the fact that often first demonstrations of theorems given by mathematicians are not the simplest or the shortest path to prove the thesis may lead to wonder why maths does not proceed in the seemingly straight-forward way that the framework of automated proving suggests. Conversely mathematics development is often discontinuous in its course, and intuition (a definitely anti-inference rule) is widely recognized to be a fundamental moment in the process of maths research. Why so? One of the most renowned mathematicians of the 20^{th} century, Vladimir Igorevic Arnold, once held a speech "On teaching mathematics" [2] in which he harshly criticized the approach to maths of the French contemporary schools, too abstract and far from application. He claimed that maths was being led astray from its natural course, that is to observe experimental facts[5].

As the dichotomy between those two approaches to maths persists and perhaps it always will, it is worthwhile to mention that the intuitionist approach to theorem proving is currently leading some researchers to build Automatic Theorem Provers whose heuristics to find proofs are already embedded into the machine thanks to machine learning [3]. Such an approach is to be regarded with interest from the intuitionist perspective.

References

[1] Peter B Andrews. "Theorem proving via general matings". In: *Journal of the ACM (JACM)* 28.2 (1981), pp. 193–214.

[4] The proposition $\neg F = F$

[5] Arnold had a quite wide idea of experimental fact, as he is famous for having said: "The Jacobi identity is an experimental fact in the same way as that the Earth is round. But it can be discovered with less expense."

[2] Vladimir I Arnol'd. "On teaching mathematics". In: *Russian Mathematical Surveys* 53.1 (1998), p. 229.

[3] James P Bridge. *Machine learning and automated theorem proving*. Tech. rep. University of Cambridge, Computer Laboratory, 2010.

[4] David A Duffy. *Principles of automated theorem proving*. John Wiley & Sons, Inc., 1991.

[5] Elliott Mendelson. *Introduction to mathematical logic*. Chapman and Hall/CRC, 2009.

[6] Neil V Murray. "Completely non-clausal theorem proving". In: *Artificial intelligence* 18.1 (1982), pp. 67–85.

[7] Kenneth H Rosen and Kamala Krithivasan. *Discrete mathematics and its applications: with combinatorics and graph theory*. Tata McGraw-Hill Education, 2012.