Theory of Automated Reasoning
An Introduction

Antti-Juhani Kaijanaho

Intended as compulsory reading for the Spring 2004 course on Automated Reasononing at Department of Mathematical Information Technology, University of Jyväskylä.

CHAPTER 1

Introduction

In the 17th Century, Gottfried Leibniz dreamt of building a system where the truth of any assertion could be determined by calculation. Leibniz wrote:

> If controversies were to arise, there would be no more need of dispute between two philosophers than between two accountants. For it would suffice to take their pencils in their hands, and say to each other: Let us calculate.

Leibniz's dream was a machine that could decide any question about what is and what is not. His dream has been shared by many, although they did not know of Leibniz's ideas. In the latter part of the 19th Century, modern logic emerged: George Boole created his algebra, Georg Cantor discovered magnitudes of infinity and developed an intuitive set theory, Gottlob Frege developed his unrestricted predicate calculus, Bertrand Russell crushed it by discovering the antinomy now known as Russell's paradox[1] and then in the first decade of the 20th Century wrote *Principia Mathematica* with Alfred Whitehead attempting to patch it up. In the first three decades of the 20th Century, research in logic was furious and fruitful, resulting in the modern approach to logic. Though this research was primarily motivated by interest in fundamentals of mathematics, it was critically important in realizing the dream of the reasoning machine. And such dream was also in the minds of the researchers.

In 1900, at the International Congress of Mathematicians in Paris, David Hilbert delivered his famous speech on mathematical problems. He described 23 open problems for the new century; one of them is particularly relevant to our topic.

Hilbert's *second problem* concerned the validity of formal arithmetic. In order to understand it, one must understand the point of view that it was posed from. Hilbert wanted to reduce all of mathematics to finite reasoning from a set of self-evident axioms. By that time, many important mathematical fields of study, such as geometry, had been reduced to the study of arithmetic. It was then critically

[1] Russell's paradox can be paraphrased as the barber's problem: A village has a bearded barber, who swears to shave every bearded man in the village who does not shave himself. Does the barber shave himself?

important for Hilbert's program to find a proof that arithmetic was consistent. As Hilbert himself said in his address:

> If contradictory attributes be assigned to a concept, I say, that *mathematically the concept does not exist*. So, for example, a real number whose square is -1 does not exist mathematically. But if it can be proved that the attributes assigned to the concept can never lead to a contradiction by the application of a finite number of logical processes, I say that the mathematical existence of the concept (for example, of a number or a function which satisfies certain conditions) is thereby proved.

The second problem, therefore, was "*[t]o prove that [the axioms of arithmetic] are not contradictory, that is, that a definite number of logical steps based upon them can never lead to contradictory results.*" Hilbert required that any such proof use only so called *finitary methods*: methods whose correctness can in principle be checked by exhaustive search in a finite amount of time. From the point of view of a reasoning machine builder, the question is equally important, for it is necessary to have a sound basis for any reasoning the machine would do; however, a reasoning machine builder does not require the use of finitary methods in these proofs — ey has no problems assuming that mathematical reasoning in general is sound.

In 1929, Mojzesz Presburger provided a partial solution to Hilbert's problem. He proved that natural number arithmetic with only addition and no multiplication is consistent (void of contradiction) and complete (capable of proving all valid statements).

It was a major blow to Hilbert's program and a huge discovery in itself when Kurt Gödel proved in 1931 that there is no proof of consistency for any axiomatic theory that is capable of expressing general arithmetic (equality, addition and multiplication of natural numbers). In fact, if such proof were to be found, it would mean that arithmetic is inconsistent and therefore, in Hilbert's words, mathematically arithmetic would not exist (and since the consistency of every other major branch of mathematics is dependent on the consistency of arithmetic, neither would they exist mathematically).

There was another problem that Hilbert posed, this time in 1928 (three years before Gödel's devastating discovery) with Wilhelm Ackermann, that was an important link in his program. It became known as the *Entscheidungsproblem* (German for "decision problem"): to find a general mechanical procedure that would be able to decide, whether any given formula of first-order logic is a theorem. A solution for this problem would immediately give a recipe for creating a reasoning machine, and if the problem was solved, the bulk of these lecture notes would consist of presenting that solution.

The first step in solving the Entscheidungsproblem would be defining what exactly is a mechanical procedure. Alonzo Church did that by defining his λ-calculus and arguing that it is a model of mechanical calculation. At about the same time, unaware of Church's work, Alan Turing created his machine and similarly argued that it is a model of mechanical calculation. Church published first, and so Turing was able to compare the two models, and as he presented his machine, he also gave a proof that it is equivalent to λ-calculus.

Both Church and Turing solved the Entscheidungsproblem by proving that there is no mechanical procedure that can decide whether an arbitrary first-order sentence is a theorem or not. *First-order logic was found to be undecidable.* This finding was another severe blow to Hilbert's program, and also seems to make this booklet unnecessary.

There is an old joke about a shop that promises immediate delivery on all orders, except for impossible ones, for which the delivery will take up to two weeks. In similar spirit, these discoveries have only slowed down the search for the reasoning machine. In the fifties, when computers were shiny and new, many researchers turned to the problem of making the computer prove theorems.

Even though first-order logic was now know to be undecidable in general, it was also known that there are particular first-order theories that are decidable. Presbuger's arithmetic (natural numbers with addition but without multiplication, as mentioned above) has a decision procedure. In 1954, Martin Davis programmed Presburger's arithmetic. It did not perform well; as Davis later wrote: "*Its great triumph was to prove that the sum of two even numbers is even.*" This is not surprising in retrospect, as it was proven in the 1970s that any decision procedure for Presburger arithmetic has super-exponential complexity.

Although there were several attempts at making a general automatic theorem prover (notably by Davis and Putnam), the first major breakthrough was John Alan Robinson's combination of unification and the resolution principle[2]. Since then, a lot of improvements have been made to both techniques. Today, most general-purpose automatic reasoning systems are based on resolution, and nearly all of them use unification. Another main technique, semantic tableaux, has emerged as an important technique in the last ten years, although it has been around since 1960's, developed by Raymond Smullyan[3] based on the work of E. W. Beth and Jaakko Hintikka in the 1950's.

---

[2] J. A. Robinson: A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, vol. 12 no. 1 (January 1965).

[3] Raymond M. Smullyan: Trees and Nest Structures. *The Journal of Symbolic Logic*, vol. 31 no. 3 (September 1966).

While the original motivation for a reasoning machine was philosophical and metamathematical, its greatest application today is in verification of software and hardware systems. Reasoning systems have also solved several open mathematical problems. Automated reasoning also has applications in artificial intelligence, database systems and programming.

This booklet develops the basic theory of automated reasoning. It assumes that the reader has the maturity of a post-baccalaureate graduate student in (theoretical) computer science. In particular, it assumes a knowledge of formal logic, theory of computation and formal languages as they are taught in computer science undergraduate (pre-baccalaureate) curricula. This booklet is a companion to the author's lectures in the Spring 2004 course on Automated reasoning at University of Jyväskylä, focusing on theory, while the lectures focus on intuitive understanding of concepts, issues and implementation techniques.

We start with a precis of formal logic. The rest of this booklet covers important techniques such as normal form transformations, term unification, semantic tableaux and resolution.

The main references used in writing this booklet are

- Melvin Fitting: *First-Order Logic and Automated Theorem Proving*, Second Edition, New York, Springer (Graduate Texts in Computer Science), 1996, and
- Alan Robinson and Andrei Voronkov, editors: *Handbook of Automated Reasoning*, Amsterdam, North-Holland (Elsevier) and Cambridge (Massachusetts), MIT Press, 2001.

Other papers and books used as sources are referred to in footnotes.