Journal of Advances in Science and Technology

Vol. IV, No. VIII, February-2013, ISSN 2230-9659

SURVEY ON DOWING GRAPHS AND GRAPH-THEORETIC DATA STRUCTURES

www.ignited.in

Journal of Advances in Science and Technology
Vol. IV, No. VIII, February-2013, ISSN 2230-9659

Survey on Dowing Graphs and Graph-Theoretic Data Structures

Vilas Dnyanu Surve[1] Dr. Pradeep Goel[2]

[1]Research Scholar, CMJ University, Shillong, Meghalaya

[2]Associate Professor, M.M. College, Fatehabad

Abstract - However, invalid formulas (those that are not entailed by a given theory), cannot always be recognized. In addition, a consistent formal theory that contains the first-order theory of the natural numbers (thus having certain "proper axioms"), by Gödel's incompleteness theorem, contains true statements which cannot be proven. In these cases, an automated theorem prover may fail to terminate while searching for a proof. Despite these theoretical limits, in practice, theorem provers can solve many hard problems, even in these undecidable logics.

A simpler, but related, problem is proof verification, where an existing proof for a theorem is certified valid. For this, it is generally required that each individual proof step can be verified by a primitive recursive function or program, and hence the problem is always decidable.

Since the proofs generated by automated theorem provers are typically very large, the problem of proof compression is crucial and various techniques aiming at making the prover's output smaller, and consequently more easily understandable and checkable, have been developed.

Key words; incompleteness theorem, theoretical limits, in practice, recursive function.

----------♦----------

INTRODUCTION

Automated theorem proving (also known as **ATP** or **automated deduction**) is a subfield of automated reasoning dealing with proving mathematical theorems by computer programs. Automated reasoning over mathematical proof was a major impetus for the development of computer science.

Logical foundations

While the roots of formalised logic go back to Aristotle, the end of the 19th and early 20th centuries saw the development of modern logic and formalised

REVIEW OF LITERATURE

Frege's *Begriffsschrift* (1879) introduced both a completepropositional calculus and what is essentially modern predicate logic.[1] His*Foundations of Arithmetic*, published 1884,[2] expressed (parts of) mathematics in formal logic. This approach was continued by Russell andWhitehead in their influential *Principia Mathematica*, first published 1910-1913,[3] and with a revised second edition in 1927.[4] Russell and Whitehead thought they could derive all mathematical truth using axioms and inference rules of formal logic, in principle opening up the process to automatisation. In 1920, Thoralf Skolem simplified a previous result by Leopold Löwenheim, leading to the Löwenheim–Skolem theorem and, in 1930, to the notion of a Herbrand universeand a Herbrand interpretation that allowed (un)satisfiability of first-order formulas (and hence the validity of a theorem) to be reduced to (potentially infinitely many) propositional satisfiability problems.[5]

In 1929, Mojżesz Presburger showed that the theory of natural numbers with addition and equality (now called Presburger arithmetic in his honor) is decidable and gave an algorithm that could determine if a given sentence in the language was true or false.[6][7] However, shortly after this positive result, Kurt Gödel published *On Formally Undecidable Propositions of Principia Mathematica and Related Systems* (1931), showing that in any sufficiently strong axiomatic system there are true statements which cannot be proved in the system. This topic was further developed in the 1930s by Alonzo Church and Alan Turing, who on the one hand gave two independent but equivalent definitions of computability, and on the other gave concrete examples for undecidable questions.

First implementations

Shortly after World War II, the first general purpose computers became available. In 1954, Martin Davis programmed Presburger's algorithm for a JOHNNIAC vacuum tube computer at the Princeton Institute for Advanced Study. According to Davis, "Its great triumph was to prove that the sum of two even numbers is even".[7][8] More ambitious was the Logic Theory Machine, a deduction system for the propositional logic of the *Principia Mathematica*, developed by Allen Newell, Herbert A. Simon and J. C. Shaw. Also running on a JOHANNIAC, the Logic Theory Machine constructed proofs from a small set of propositional axioms and three deduction rules: modus ponens, (propositional) variable substitution, and the replacement of formulas by their definition. The system used heuristic guidance, and managed to prove 38 of the first 52 theorems of the *Principia*.[7]

MATERIAL AND METHOD

The "heuristic" approach of the Logic Theory Machine tried to emulate human mathematicians, and could not guarantee that a proof could be found for every valid theorem even in principle. In contrast, other, more systematic algorithms achieved, at least theoretically, completeness for first-order logic. Initial approaches relied on the results of Herbrand and Skolem to convert a first-order formula into successively larger sets of propositional formulae by instantiating variables with terms from the Herbrand universe. The propositional formulas could then be checked for unsatisfiability using a number of methods. Gilmore's program used conversion to disjunctive normal form, a form in which the satisfiability of a formula is obvious.[7][9]

Depending on the underlying logic, the problem of deciding the validity of a formula varies from trivial to impossible. For the frequent case of propositional logic, the problem is decidable but Co-NP-complete, and hence only exponential-time algorithms are believed to exist for general proof tasks. For a first order predicate calculus, with no ("proper") axioms, Gödel's completeness theorem states that the theorems (provable statements) are exactly the logically valid well-formed formulas, so identifying valid formulas is recursively enumerable: given unbounded resources, any valid formula can eventually be proven.

However, *invalid* formulas (those that are *not* entailed by a given theory), cannot always be recognized. In addition, a consistent formal theory that contains the first-order theory of the natural numbers (thus having certain "proper axioms"), by Gödel's incompleteness theorem, contains true statements which cannot be proven. In these cases, an automated theorem prover may fail to terminate while searching for a proof. Despite these theoretical limits, in practice, theorem provers can solve many hard problems, even in these undecidable logics.

Related problems

A simpler, but related, problem is **proof verification**, where an existing proof for a theorem is certified valid. For this, it is generally required that each individual proof step can be verified by a primitive recursive function or program, and hence the problem is always decidable.

Since the proofs generated by automated theorem provers are typically very large, the problem of proof compression is crucial and various techniques aiming at making the prover's output smaller, and consequently more easily understandable and checkable, have been developed.

Proof assistants require a human user to give hints to the system. Depending on the degree of automation, the prover can essentially be reduced to a proof checker, with the user providing the proof in a formal way, or significant proof tasks can be performed automatically. Interactive provers are used for a variety of tasks, but even fully automatic systems have proven a number of interesting and hard theorems, including some that have eluded human mathematicians for a long time.[10][11] However, these successes are sporadic, and work on hard problems usually requires a proficient user.

Another distinction is sometimes drawn between theorem proving and other techniques, where a process is considered to be theorem proving if it consists of a traditional proof, starting with axioms and producing new inference steps using rules of inference. Other techniques would include model checking, which, in the simplest case, involves brute-force enumeration of many possible states (although the actual implementation of model checkers requires much cleverness, and does not simply reduce to brute force).

There are hybrid theorem proving systems which use model checking as an inference rule. There are also programs which were written to prove a particular theorem, with a (usually informal) proof that if the program finishes with a certain result, then the theorem is true. A good example of this was the machine-aided proof of the four color theorem, which was very controversial as the first claimed mathematical proof which was essentially impossible to verify by humans due to the enormous size of the program's calculation (such proofs are called non-surveyable proofs). Another example would be the proof that the game Connect Four is a win for the first player.

Industrial uses

Commercial use of automated theorem proving is mostly concentrated in integrated circuit design and verification. Since the Pentium FDIV bug, the complicated floating point units of modern microprocessors have been designed with extra

scrutiny. Nowadays[when?],AMD, Intel and others use automated theorem proving to verify that division and other operations are correctly implemented in their processors.

First-order theorem proving

First-order theorem proving is one of the most mature subfields of automated theorem proving. The logic is expressive enough to allow the specification of arbitrary problems, often in a reasonably natural and intuitive way. On the other hand, it is still semi-decidable, and a number of sound and complete calculi have been developed, enabling *fully* automated systems. More expressive logics, such as higher order logics, allow the convenient expression of a wider range of problems than first order logic, but theorem proving for these logics is less well developed.

Benchmarks and competitions

The quality of implemented system has benefited from the existence of a large library of standard benchmark examples — the Thousands of Problems for Theorem Provers (TPTP) Problem Library[12] — as well as from the CADE ATP System Competition(CASC), a yearly competition of first-order systems for many important classes of first-order problems.

CONCLUSION

- E is a high-performance prover for full first-order logic, but built on a purely equational calculus, developed primarily in the automated reasoning group of Technical University of Munich.

- Otter, developed at the Argonne National Laboratory, is the first widely used high-performance theorem prover[citation needed]. It is based on first-order resolution and paramodulation. Otter has since been replaced by Prover9, which is paired with Mace4.

- SETHEO is a high-performance system based on the goal-directed model elimination calculus. It is developed in the automated reasoning group of Technical University of Munich. E and SETHEO have been combined (with other systems) in the composite theorem prover E-SETHEO.

- Vampire is developed and implemented at Manchester University by Andrei Voronkov and Krystof Hoder, formerly also byAlexandre Riazanov. It has won the CADE ATP System Competition in the most prestigious CNF (MIX) division for eleven years (1999, 2001–2010).

- Waldmeister is a specialized system for unit-equational first-order logic. It has won the CASC UEQ division for the last fourteen years (1997–2010).

- SPASS is a first order logic theorem prover with equality. This is developed by the research group Automation of Logic, Max Planck Institute for Computer Science.

REFERENCES

1. ^ Richard Johnsonbaugh, *Discrete Mathematics*, Prentice Hall, 2008.

2. ^ Weisstein, Eric W., "Discrete mathematics" from MathWorld.

3. ^ Norman L. Biggs, *Discrete mathematics*, Oxford University Press, 2002.

4. ^ Brian Hopkins, *Resources for Teaching Discrete Mathematics*, Mathematical Association of America, 2008.

5. ^ [a] [b] Wilson, Robin (2002). *Four Colors Suffice*. London: Penguin Books. ISBN 978-0-691-11533-7.

6. ^ Trevor R. Hodkinson; John A. N. Parnell (2007). *Reconstruction the Tree of Life: Taxonomy And Systematics of Large And Species Rich Taxa*. CRC PressINC. p. 97. ISBN 978-0-8493-9579-6.

7. ^ "Millennium Prize Problems". 2000-05-24. Retrieved 2008-01-12.

8. ^ A. S. Troelstra; H. Schwichtenberg (2000-07-27). *Basic Proof Theory*. Cambridge University Press. p. 186. ISBN 978-0-521-77911-1.

9. ^ Samuel R. Buss (1998). *Handbook of Proof Theory*. Elsevier. p. 13. ISBN 978-0-444-89840-1.

10. ^ Franz Baader; Gerhard Brewka, Thomas Eiter (2001-10-16). *KI 2001: Advances in Artificial Intelligence: Joint German/Austrian Conference on AI, Vienna, Austria, September 19-21, 2001. Proceedings*. Springer. p. 325. ISBN 978-3-540-42612-7.

11. ^ Brotherston, J.; Bornat, R.; Calcagno, C. (January 2008). "Cyclic proofs of program termination in separation logic". *ACM SIGPLAN Notices* **43** (1). CiteSeerX: 10.1.1.111.1105.

12. ^ Graphs on Surfaces, Bojan Mohar and Carsten Thomassen, Johns Hopkins University press, 2001