Future directions of automated deduction: Strategy analysis for theorem proving

Maria Paola Bonacina *
Department of Computer Science
University of Iowa
Iowa City, IA 52242-1419, USA
bonacina@cs.uiowa.edu

Keywords: theorem proving, search strategies, complexity measures, strategy analysis.

A new direction for research in automated deduction is the development of mathematical tools for the analysis and evaluation of theorem-proving strategies. We propose the name *strategy analysis*, by analogy with algorithm analysis. In the following, we give some motivation, mention some of the difficulties of this research, describe briefly current and future work, and comment on its impact on the field.

Motivation The motivation is a long-standing gap between the theory and the practice of theorem proving. On one hand, there are indeed theorem-proving strategies whose implementations are capable of solving significant problems (e.g., [1, 2, 4, 5, 6]). On the other hand, there is no theory of "strategy analysis", comparable to algorithm analysis, for evaluating these strategies. As a consequence, theorem-proving strategies are usually evaluated by benchmarking of their implementations. While benchmarking is eventually necessary, it is unsatisfactory that it is the only available approach, because its results are specific (depending on the input data, the implementation, the software and hardware environment), and coarse (theorem provers are complex objects made of many components, and it is difficult to establish quantitatively how different components contributed to the observed performances).

The challenge of strategy analysis Theorem-proving problems usually have *infinite* search spaces. The main reason for the absence of "strategy analysis" is the lack of formal tools to analyze the complexity of problems involving search in an infinite search space. There are several obstacles in analyzing the complexity of search in an infinite space, including the following:

- The methodology of traditional complexity analysis is not suitable, because it is concerned mainly with the asymptotic analysis of *finite* objects, with *time* and *space* the two dominating measures. Given that the search space is infinite, it is no longer meaningful to discuss about average case analysis, much less worst case.

*Supported in part by the National Science Foundation with grant CCR-94-08667.

- In a finite problem the time and space complexities can usually be treated as functions of a measure of the input. But for first-order logic, for instance, the difficulty of finding a proof is not related to the size of the input set of clauses. A source of the problem is that a set of first-order clauses represents more than itself: it represents the infinite set of the ground instances of the clauses. Thus, any measure based on the input alone is not sufficient.

- Neither is the complexity of a search strategy related to its output. In theorem proving, the output is the computed proof, if a proof is produced. The size of the proof is generally not indicative of the difficulty of *finding* the proof, since one may find a short proof after an extensive traversal of the search space.

Thus, a more accurate notion of complexity should be how difficult the *process* of finding the proof is, rather than either the input or the output of the computation.

An approach towards strategy analysis In recent and continuing work [3], we have proposed a new way of analysis to reason about and to compare theorem-proving strategies. Our methodology is based on a model for representing search in theorem proving. A primary goal in designing this model has been to accurately describe the behaviour of *contraction* inferences, such as subsumption and simplification, which have been used successfully in theorem proving. Unlike the usual expansion inference rules, such as resolution, which merely visit the search space, contraction inferences visit and *modify* the search space at the same time. Our model captures this dynamic behaviour by introducing a notion of *marking* into the search graph, which allows us to describe the search plan of a strategy as well as its inference mechanism.

Based on the concept of marked search graph, we have introduced a notion of complexity measure for analyzing and evaluating the behaviour of strategies. Unlike the conventional complexity measures which work with finite objects, we need to deal with infinite search graphs and derivations which may not halt. Our notion of complexity analysis, based on well-founded orderings instead of natural numbers, captures both the present and the future of a possibly infinite derivation. To do this properly, we have defined notions of *ancestor-graph* and *dynamic distance* to replace the conventional notions of path and path-lengh. This allows us to "finitize" the future, the portion of the infinite search graph not yet discovered, into a sequence of finite search graphs within bounded distances. We have shown the applicability of our framework by using it to compare contraction-based strategies of different contraction power, showing how they affect the evolution of the respective search spaces during the derivation.

Future work Since our work has only recently shown even the possibility of strategy analysis in infinite spaces, the development of this subarea of automated deduction lies almost completely in the future. Possible directions include the following:

- *Extension to other classes of strategies.* The analysis in [3] concentrated on forward-reasoning, contraction-based strategies. We feel that the main ideas in our work may capture essential aspects of infinite search, and therefore may be applied to other classes of strategies, such as backward-reasoning strategies, in automated deduction or artificial intelligence.

- *Analysis of search plans.* Theorem-proving strategies may differ in many parameters. As a first cut, one may distinguish between comparing strategies that have the same inference system and different search plans, and comparing strategies that have the same search plan and different inference systems. In [3], we considered an instance of the second type of problem. Thus, another direction is to compare strategies that have different search plans.

- *Analysis of parallel/distributed strategies.* An additional motivation to work on the modelling of the complexity of search came to us from our work on distributed automated deduction. Strategy analysis is a prerequisite to address the question of how parallelism may reduce the search complexity of theorem proving.

Impact The impact of developing a theory of strategy analysis may be far-reaching. Classical algorithm analysis applies to decidable problems. On the other hand, many subfields of computer science deal with problems that are not decidable in their general formulation. Nonetheless, non-trivial classes of instances of such problems are solved mechanically, witness the field of theorem proving. This is mainly because the existence of an infinite search space may not require an infinite amount of computational resources, since it is not necessary to traverse the entire search space to find a solution. In principle, the availability of tools for strategy analysis may have on all these problems an impact similar to that of algorithm analysis on finite problems. For instance, if strategy analysis establishes that strategy A has smaller search complexity than strategy B on the problems in a certain class, then researchers may concentrate on improving and implementing strategy A, or, at least, strategies with the features that differentiate A from B. Better and more specialized strategies will then impact positively all applications of automated deduction.

References

[1] Anantharaman, S. and Hsiang, J., Automated proofs of the Moufang identities in alternative rings, *J. Automated Reasoning* **6**(1) (1990) 76–109.

[2] Astrachan, O. L. and Loveland, D. W., METEORs: High performance theorem provers using model elimination, in R. S. Boyer, (ed.), *Automated Reasoning: Essays in Honor of Woody Bledsoe*, Kluwer Academic Publisher, Dordrecht, The Netherlands, 1991, pp. 31–60.

[3] Bonacina, M. P. and Hsiang, J., On the modelling of search in theorem proving – Towards a theory of strategy analysis, submitted for publication and available as technical report, Department of Computer Science, University of Iowa, December 1995.

[4] Kapur, D. and Zhang, H., RRL: a Rewrite Rule Laboratory, in E. Lusk, R. Overbeek (eds.), *Proc. Nineth International Conference on Automated Deduction*, Argonne, Illinois, May 1988, Lecture Notes in Computer Science 310, Springer-Verlag, New York, 1988, pp. 768–770.

[5] McCune, W. W., Otter 3.0 Reference Manual and Guide, Technical Report ANL-94/6, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, January 1994.

[6] Stickel, M. E., A Prolog Technology Theorem Prover: Implementation by an Extended Prolog Compiler, *J. Automated Reasoning* **4** (1988) 353-380.