Measuring progress to predict success: Can a good proof strategy be evolved? (Extended Abstract)

Giles Reger[1] and Martin Suda[2]

[1] University of Manchester, Manchester, UK
[2] TU Wien, Vienna, Austria

One of the main parameters of the superposition calculus employed by Automated Theorem Provers (ATPs) is the simplification ordering, and the choice of an ordering can have a huge impact on the success of a theorem proving attempt. However, it is difficult to choose a good ordering in advance and ATPs typically provide only a few heuristical schemes for determining an ordering from a large space of possibilities. The aim of this work is to establish to what extent the space of possible orderings can be better utilised during the construction of new successful proving strategies.

There is a well known principle in automated deduction which states that a strategy that leads to a slowly growing search space will likely be more successful (at finding a proof in reasonable time) than a strategy that leads to a rapidly growing one. We propose to employ this principle and search for a strategy which, for a given problem, minimises the number of derived clauses after a certain number of iterations of the saturation loop. Focusing on the search for a good ordering as a simplifying restriction on the set of available strategies, we experimentally investigate the practical potential of this idea.

Simplification orderings in superposition-based theorem proving. The superposition calculus [7] used by modern ATPs such as E [9], SPASS [13], and Vampire [4] is parametrised by a simplification ordering on terms. The two most commonly used classes of orderings are the Lexicographic Path Ordering (LPO) [2] and the Knuth-Bendix Ordering (KBO) [3]. To get a concrete instance of an ordering from either class one needs to specify a symbol *precedence*, a total order on the predicate and function symbols occurring in the given problem.[1]

It is well known that the choice of a concrete ordering can have a huge impact on the success or failure of the subsequent proof attempt. For example, giving a high precedence to symbols introduced during clausification as names of sub-formulas [8], will effectively lead to their immediate elimination during saturation and thus give rise to an exponential clause set. Nevertheless, since there is no obvious general way to choose a good ordering in advance, ATPs typically provide only a few heuristical schemes for the automatic selection of the precedence. This leaves the majority of the $n!$ possibilities on how to choose a precedence (for a problem with a signature of size n) inaccessible to a typical theorem proving strategy.

The role of strategies in modern ATPs. Theorem provers typically have a large number of options for organising proof search and a *strategy* is obtained by fixing concrete values for these options. Since there cannot be a universally best strategy for solving all problems, an ATP may try to predict the best strategy for a given problem based on the problem's features [6, 11, 9]. An arguably more robust approach is to split the allotted time amongst a *schedule of strategies* and execute them one by one or in parallel [10, 5].[2] The success of strategy

[1] To fully determine a KBO, one also needs to specify an admissible weight function.
[2] Vampire [4] also heavily relies on strategy scheduling, but the details are currently unpublished.

scheduling can be explained by the observation pertaining to first-order theorem proving: *if a strategy solves a problem then it typically solves it within a short amount of time*. Thus there is a demand for methods for discovering new good strategies, especially strategies able solve previously unsolved problems [12].

Slow search space growth as a guiding principle. Previous work on literal selection heuristics [1] experimentally established that strategies which lead to a slowly growing search space tend to be more successful at finding proofs (within the allotted time limit). Here we propose to use this observation as a general principle for finding good strategies for solving a particular problem. The idea is to look for strategies which minimize the number of derived clauses after a certain number of iterations of the saturation loop. In contrast to random sampling of the strategy space, this approach promises to be a *directed* search for strategies that can solve previously unsolved problems!

As a proof of concept, we focus here on the space of possible orderings and, more specifically, on precedences specifying an ordering, and try to establish the extent to which the proposed idea can be useful in practice. There is a limiting factor in the fact that a precedence must be fixed in advance and cannot be changed during proof search. Thus, optimizing the precedence (exploration) and utilising a precedence shown to perform well to actually solve the problem (exploitation) must be understood as separate activities, which leads to a refinement of the question from the title: Can a good proof strategy be evolved *in time*?

References

[1] K. Hoder, G. Reger, M. Suda, and A. Voronkov. Selecting the selection. In *IJCAR 2016*, pp. 313–329. Springer International Publishing, 2016.

[2] S. Kamin and J.-J. Levy. Two generalizations of the recursive path ordering. Departement of Computer Science, University of Illinois, Urbana, IL, 1980.

[3] D. E. Knuth and P. B. Bendix. *Simple Word Problems in Universal Algebras*, pp. 342–376. Springer Berlin Heidelberg, 1983.

[4] L. Kovács and A. Voronkov. First-order theorem proving and Vampire. In *CAV 2013*, vol. 8044 of *Lecture Notes in Computer Science*, pp. 1–35, 2013.

[5] D. Kühlwein, S. Schulz, and J. Urban. E-MaLeS 1.1. In *CADE-24, 2013.*, vol. 7898 of *Lecture Notes in Computer Science*, pp. 407–413. Springer, 2013.

[6] W. McCune and L. Wos. Otter - the CADE-13 competition incarnations. *J. Autom. Reasoning*, 18(2):211–220, 1997.

[7] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In *Handbook of Automated Reasoning*, vol. I, chapter 7, pp. 371–443. Elsevier Science, 2001.

[8] G. Reger, M. Suda, and A. Voronkov. New techniques in clausal form generation. In *GCAI 2016*, vol. 41 of *EPiC Series in Computing*, pp. 11–23. EasyChair, 2016.

[9] S. Schulz. E - a brainiac theorem prover. *AI Commun.*, 15(2-3):111–126, 2002.

[10] G. Stenz and A. Wolf. E-SETHEO: an automated^3 theorem prover. In *TABLEAUX 2000*, vol. 1847 of *Lecture Notes in Computer Science*, pp. 436–440. Springer, 2000.

[11] T. Tammet. Gandalf. *J. Autom. Reasoning*, 18(2):199–204, 1997.

[12] J. Urban. BliStr: the blind strategymaker. In *GCAI 2015*, vol. 36 of *EPiC Series in Computing*, pp. 312–319. EasyChair, 2015.

[13] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski. SPASS version 3.5. In *CADE-22, 2009. Proceedings*, vol. 5663 of *LNCS*, pp. 140–145. Springer, 2009.