Can a Failed Strategy be Useful?
(A Research Proposal)

Giles Reger[1] and Martin Suda[2]

[1] University of Manchester, Manchester, UK
[2] Czech Technical University in Prague, Czech Republic

Modern automatic theorem provers for first-order logic (ATPs), such as E [7][1] or Vampire [2][2], provide many *options* for setting up and tuning the deduction algorithm, thus giving rise to a huge number of possible *strategies* that can be used to attack a problem. To achieve their best performance, these ATPs employ *strategy scheduling*, a technique in which they execute a *sequence* of strategies of complementary strengths, each for a certain fraction of the allotted time. There are usually about a dozen of such pre-selected sequences and the prover picks one based on a number of simple problem features such as the size of the signature, presence of equality, the shape of clauses, etc. Apart from this initial choice, however, the strategies are executed in a fixed sequential order (or, optionally, in parallel) until one of them finds a solution or an overall time limit is reached. In particular, there is no sharing of information between the strategies, which can thus be viewed as independent attempts at solving the problem. In this work, we want to investigate to what degree could this simple setup be improved by extracting information from the failed runs of earlier strategies and making use of this information during the execution of the later strategies.

There seem to be at least two conceptually distinct ways of elaborating this idea. In the first one, we ask whether the *proof search characteristics* of a failed run could be used as additional features of the given problem, helping to choose the next strategy to try on the problem in a more informed way. In this view, running a strategy is not just an attempt at solving the problem, but also a probe aimed at learning more about it. In the second, we ask to what degree it would be possible to retain certain parts of *logical information* derived from the problem and successfully reuse them in the subsequent strategy runs in the form of lemmas. These lemmas should ideally be non-obvious (i.e. hard to derive) consequences of the problem formulation to represent enough invested work to be worth transferring. At the same time, they should be of sufficiently simple form not to overload or distract the freshly starting strategy. Ultimately, these considerations lead to an architecture with a large room for experimental tuning (subject to a particular target distribution of problems). We eventually want to investigate how to make this tuning fully automatic by posing the task as an instance of *reinforcement learning*.

Focusing on failures The second aspect of our proposed improvement, i.e. that of sharing logical information between individual strategies, has already been studied in the past in various incarnations, see e.g. [1, 6, 9, 12]. Although encouraging results were always reported, the fact that none of the current "mainstream"[3] ATPs is employing the idea suggests that the ensuing architectures might be hard to maintain or, perhaps, that further theoretical understanding is needed before the idea can be applied in a sufficiently principled way.

That is why we would first like to further our general understanding of the behaviour of strategies by focusing on the first aspect mentioned above, i.e. the question of to what extent can the information about a failed strategy be useful to eventually solving a given problem.

[1] http://www.eprover.org/
[2] https://vprover.github.io/
[3] Such as the ones participating in the annual CASC competition [10].

Proof search characteristics that could be readily used here include the number of generated and selected clauses, the number of performed inferences and reductions of each kind or the amount of time spent by the prover in the individual proving sub-routines such as parsing, preprocessing, reductions, or the maintenance of individual data structures. An encouragement that such runtime statistics can be informative comes from our previous work on the evolution of simplification orderings [5] as well as from the work on predicting the success of strategies at runtime [4]. We even believe that these characteristics could completely replace the more classical static problem features used for problem clustering in the past [3].

A data-mining experiment The strategy scheduling mode (a.k.a. CASC mode) of Vampire 4.3.0, consists of approximately 1200 strategies in 39 scheduled sequences dispatched according to simple syntactic properties of the given problem. As our initial experiment, we intend to evaluate these strategies on a feasible subset of the TPTP library [11], collecting the information about successful runs as well as proof search characteristics of the failed runs. This will provide a data-set for the evaluation of our hypothesis that the characteristics of the failed runs, when used as features of a problem, can be used to improve the selection process of a strategy that will succeed on the problem.

We would also like to use the collected data to further explore and statistically quantify previously established phenomena such as 1) the inherent fragility of the space of strategies: Are there certain options parametrising a strategy the change of which tends to have a continuous effect on the strategy's behaviour or is the behaviour almost always chaotic? 2) the effect of sub-linearity [8], usually expressed as the observation that: If a problem can be solved by a strategy then it typically can be solved within a short amount of time. Ultimately, we would like to shed more light on the question 3) to what degree is predicting a good strategy for a particular problem actually feasible or practical and to what degree is "just quickly trying sufficiently orthogonal strategies in succession" enough to solve all the solvable problems.

References

[1] J. Denzinger and M. Kronenburg. Planning for distributed theorem proving: The teamwork approach. In *KI-96: Advances in Artificial Intelligence, 20th Annual German Conference on Artificial Intelligence, Dresden, Germany, September 17-19, 1996, Proceedings*, vol. 1137 of *Lecture Notes in Computer Science*, pp. 43–56. Springer, 1996.

[2] L. Kovács and A. Voronkov. First-order theorem proving and Vampire. In *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, vol. 8044 of *Lecture Notes in Computer Science*, pp. 1–35. Springer, 2013.

[3] D. Kühlwein and J. Urban. MaLeS: A framework for automatic tuning of automated theorem provers. *J. Autom. Reasoning*, 55(2):91–116, 2015.

[4] M. Rawson and G. Reger. Dynamic strategy priority: Empower the strong and abandon the weak. In *PAAR 2018, Proceedings of the 6th Workshop on Practical Aspects of Automated Reasoning co-located with Federated Logic Conference 2018*, 2018.

[5] G. Reger and M. Suda. Measuring progress to predict success: Can a good proof strategy be evolved? (extended abstract). In *AITP 2017, The Second Conference on Artificial Intelligence and Theorem Proving, Abstracts of the talks*, pp. 20–21, 2017.

[6] G. Reger, D. Tishkovsky, and A. Voronkov. Cooperating proof attempts. In *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, vol. 9195 of *Lecture Notes in Computer Science*, pp. 339–355. Springer, 2015.

[7] S. Schulz. System Description: E 1.8. In *Proc. of the 19th LPAR, Stellenbosch*, vol. 8312 of *LNCS*. Springer, 2013.

[8] G. Stenz and A. Wolf. E-SETHEO: design, configuration and use of a parallel automated theorem prover. In *Advanced Topics in Artificial Intelligence, 12th Australian Joint Conference on Artificial Intelligence, AI '99, Sydney, Australia, December 6-10, 1999, Proceedings*, vol. 1747 of *Lecture Notes in Computer Science*, pp. 231–243. Springer, 1999.

[9] G. Sutcliffe. The design and implementation of a compositional competition-cooperation parallel ATP system. In *Proceedings of the 2nd International Workshop on the Implementation of Logics*, pp. 92–102, 2001.

[10] G. Sutcliffe. The CADE ATP System Competition - CASC. *AI Magazine*, 37(2):99–101, 2016.

[11] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. From CNF to TH0, TPTP v6.4.0. *Journal of Automated Reasoning*, 59(4):483–502, 2017.

[12] A. Wolf and M. Fuchs. Cooperative parallel automated theorem proving. Technical report, SFB Bereicht 342/21/97, Technische Universität München, 1997.