Do Portfolio Solvers Harm?

Christoph Weidenbach

Max Planck Insitute for Informatics
Saarland Informatics Campus
66123 Saarbrücken, Germany
weidenbach@mpi-inf.mpg.de

Abstract

I discuss the question whether portfolio solvers support advances in automated reasoning. A portfolio solver is the combination of a collection of core solvers. I distinguish syntactic combinations from semantic combinations and argue that the former are useful for competitions where the latter foster progress in automated reasoning.

1 Introduction

I discuss the question whether portfolio solvers support advances in automated reasoning. In particular, I'm interested in advances in the theory of automated reasoning, e.g., the development of new calculi, rather than in solutions to specific problems, e.g., winning a systems competition. A *solver* decides (un)satisfiability of a formula of some logic. In this paper, I mainly consider propositional logic, decidable fragments of first-order logic, and full first-order logic. A *portfolio solver* is the combination of a collection of core solvers. A *Simple-Syntactic portfolio solver* or SS-portfolio solver is characterized by a combination of core solvers where the selection of the core solvers is done by purely syntactic problem properties and there is no exchange of results between different core solvers. They run independently. A *Sophisticated-Semantic portfolio solver* or SM-portfolio solver is characterized by a combination of core solvers where the selection of the core solvers is done by semantic or structural problem properties and the solvers exchange results. They run dependently.

2 SS-Portfolio Solvers

There is a long tradition in the SAT community discussing the role of SS-portfolio solvers. The SAT competition 2016 includes the rule

```
Participation of Portfolios 2016:
By a portfolio SAT solver we mean a combination of two or more
(core) SAT solvers developed by a different group of authors.
A portfolio SAT solver may participate only in the "No-Limits"
track of the competition.
```

which obviously excludes portfolio solvers of a particular kind from the main tracks of the competition. Only those portfolio solvers are excluded that are not built by the authors of the single solvers. If an author of several core SAT solvers, e.g. with different parameter configurations, combines them in a portfolio solver she/he can participate in the main tracks. An interpretation of the rule is: "The glory for building competitive core solvers solely belongs to the builder of these solvers." The SAT competition creates visibility to the outside of the SAT community. People from the outside may not be able to distinguish between the competence of combining SAT solvers via an SS-portfolio machine learning approach and the actual further development of core SAT solvers and their respective theory.

The SAT community has a long history in the discussion of SS-portfolio solvers where a prominent example is SATzilla [29] a gold medal winning SS-portfolio SAT solver at the SAT competitions 2007 and 2009. The authors of SATzilla took several core SAT solvers and by the incorporation of machine learning techniques combined them to a very powerful SS-portfolio SAT solver: SATzilla. It selects a core solver for a particular problem by purely syntactic criteria. Some of the authors of SATzilla are actually closer to the machine learning community than to the SAT community.

Proposition 2.1. A standard way of building SS-portfolio solvers from core solver instances is by core solver selection from syntactic problem properties based on machine learning techniques with training on problem libraries.

There is meanwhile a branch in automated reasoning research investigating the potential of machine learning in solver development, e.g. [12], often combined with an SS-portfolio approach. Leading competition versions of solvers for the "main" divisions of the first-order logic theorem proving competition CASC [26] namely E [23], iProver [15] and Vampire [17] are all SS-portfolio solver instances. E subsequently runs several different superposition strategies found by a machine learning approach. In addition, iProver and Vampire run implementations of different calculi such as Superposition/Ordered Resolution [3], InstGen [10], and reasoning with respect to finite models [19, 24, 9, 6] in a time-slicing approach.

Proposition 2.2. SS-portfolio solvers are particularly strong in competitions on diverse problem libraries.

Typically, SS-portfolio solvers run in a time-slicing approach. Now in order to be successful in a competition based on a problem library, the problems must not be hard in the sense that if the right core solver with the right parameters is picked, the problem can be solved fast. On the other hand the problem library must be diverse in the sense that a single solver instance cannot efficiently cope with all types of problems. So the above proposition can be further strengthened.

Claim 2.3. SS-Portfolio solvers are only useful for winning competitions on diverse domains where single problems are not hard.

If problem domains become more specific, such as in dedicated applications, or single problems become difficult, then SS-portfolio solvers are typically not a preferred option. For example, see the hardware model checking competition [14], complete reasoning in large ontologies [25], or reasoning in the context of complexity management [11].

Proposition 2.4. SS-Portfolio solvers are not particularly useful in dedicated problem domains.

Now combining all of the above, SS-portfolio solvers are mainly useful for a particular kind of competition but not from a scientific perspective.

Consequence 2.5. The development of SS-portfolio solvers does not contribute to the scientific progress in automated reasoning.

3 SM-Portfolio Solvers

The SS-portfolio solvers mentioned in the previous sections can be build at an engineering/implementation level, i.e., there is no calculus of portfolio solving but implementations of invoking different solvers based on heuristics and/or machine learning results. For an SM-Portfolio solver there is a demand for theory, e.g., what it means for the solvers to exchange results and to guarantee typical properties such as soundness and completeness for the combination of solvers.

There are a number of successful SM-portfolio solver approaches. The Nelson-Oppen combination [20] and resulting SMT solvers [21] are an example. They combine solvers for decidable theories in order to solve a problem in the more expressive logic of the union of the theories. Hierarchic superposition [4, 18] is another example, where a solver for a base theory is combined with the superposition calculus for first-order logic. Actually, the successful combination of theories via an SM-portfolio approach has had an enormous effect on the development of the field of automated reasoning in the past decade.

Consequence 3.1. The development of SM-portfolio solvers contributes to the scientific progress in automated reasoning.

However, there is currently no convincing SM-portfolio approach for a single logic. At least for logics beyond propositional logic (SAT) this could result in a break through: In SAT the size of a model for some clause set is small, at most the number of variables. Truth or falsehood of a clause can be decided in linear time with respect to a (partial) model. In first-order logic there cannot be an effective finite model representation, in general, due to undecidability. Actually the representation and use of models in first-order logic is still in its infancy. Even for decidable fragments enjoying the finite model property the problem of model representations is not solved. For example, consider the decidable Bernays-Schoenfinkel (BS) fragment of first-order logic. For this class there are finite, effective model representations, e.g., sequences of ground literals. However, a model representation based on ground literals is worst-case of exponential size. There are a number of results on calculi deciding the BS class and making use of explicit model representations [5, 22, 1] that are exponentially more compact than sequences of ground literals. The model representations differ in expressiveness and in the complexity of deciding the truth, falsehood or propagation of a clause with respect to the model representation. There is no clear "winner". Even a ground instantiation and afterwards SAT solving can be an option, if the ground instantiation does not get "too large". This is the standard technique successfully used in *Answer Set Programming*, ASP [13].

Actually, there cannot be a clear "winner" model representation. Satisfiability of BS is NEXPTIME-complete and hence not in NP. So there cannot exist a compact model representation for which the truth of a clause is polynomially decidable, in general. So a natural approach would be to combine different calculi and model representations in an SM-portfolio solver. Of course, the problem of how to separate a BS problem between the different calculi and how to combine the results needs to be solved.

The situation for full first-order logic is even more difficult. Satisfiability is undecidable. So there cannot be a finite, effective model representation, in general. Still, there are calculi that operate with respect to a model representation [5, 7] at the price that models beyond the actual concrete model representation cannot be found. A way out here could be the use of approximation techniques in combination with SM-portfolio solving. The idea of the SM-portfolio solver could be to approximate (parts of) the problem into a decidable class, use a dedicated solver for the class, combine the result with a general procedure or further approximations.

First approaches following this paradigm exist [16, 27]. But there is still the open problem of more sophisticated combinations and the incorporation of equational reasoning.

There has been work in the past on combining different solvers/calculi or different instances of the same solver/calculus in a run by exchanging results, e.g., [2, 8]. However, none of these approaches matured or survived. A key challenge is a criterion for picking the results to be exchanged. In my opinion, there is the need for a "theory" choosing the criterion dynamically with respect to the set of generated clauses. An analogous problem is the selection of an ordering in ordered resolution. CDCL clause learning can be understood as a heuristic dynamically choosing for ordering of the ordered resolution calculus. The resolution steps deriving learned clauses are actually ordered resolution steps where the ordering is dynamically given by the order of literals on the trail [28].

Proposition 3.2. SM-Portfolio solvers are a promising approach for making progress in automated reasoning in first-order logic in general.

Acknowledgments: I am grateful to Maria Paola Bonacina, Jürgen Giesel and Stephan Schulz for their detailed comments on an earlier version of this abstract.

References

[1] Gábor Alagi and Christoph Weidenbach. NRCL - A model building approach to the bernays-schönfinkel fragment. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems - 10th International Symposium, FroCoS 2015, Wroclaw, Poland, September 21-24, 2015. Proceedings*, volume 9322 of *Lecture Notes in Computer Science*, pages 69–84. Springer, 2015.

[2] Jürgen Avenhaus, Jörg Denzinger, and Matthias Fuchs. DISCOUNT: A system for distributed equational deduction. In Jieh Hsiang, editor, *Rewriting Techniques and Applications, 6th International Conference, RTA-95, Kaiserslautern, Germany, April 5-7, 1995, Proceedings*, volume 914 of *Lecture Notes in Computer Science*, pages 397–402. Springer, 1995.

[3] Leo Bachmair and Harald Ganzinger. On restrictions of ordered paramodulation with simplification. In *10th International Conference on Automated Deduction, CADE-10*, volume 449 of *LNCS*, pages 427–441. Springer, 1990.

[4] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Applicable Algebra in Engineering, Communication and Computing, AAECC*, 5(3/4):193–212, 1994.

[5] Peter Baumgartner, Alexander Fuchs, and Cesare Tinelli. Lemma learning in the model evolution calculus. In *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 572–586. Springer, 2006.

[6] Peter Baumgartner, Ulrich Furbach, and Björn Pelzer. The hyper tableaux calculus with equality and an application to finite model computation. *Journal Log. Comput.*, 20(1):77–109, 2010.

[7] Maria Paola Bonacina, Ulrich Furbach, and Viorica Sofronie-Stokkermans. On first-order model-based reasoning. In Narciso Martí-Oliet, Peter Csaba Ölveczky, and Carolyn L. Talcott, editors, *Logic, Rewriting, and Concurrency - Essays dedicated to José Meseguer on the Occasion of His 65th Birthday*, volume 9200 of *Lecture Notes in Computer Science*, pages 181–204. Springer, 2015.

[8] Maria Paola Bonacina and Jieh Hsiang. Distributed deduction by clause-diffusion: Distributed contraction and the aquarius prover. *Journal of Symbolic Computation*, 19(1-3):245–267, 1995.

[9] Koen Claessen and Ann Lillieström. Automated inference of finite unsatisfiability. In Renate A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 388–403. Springer, 2009.

[10] Harald Ganzinger and Konstantin Korovin. New directions in instatiation–based theorem proving. In Samson Abramsky, editor, *18th Annual IEEE Symposium on Logic in Computer Science, LICS'03*, LICS'03, pages 55–64. IEEE Computer Society, 2003.

[11] Christopher Junk, Robert Rößger, Georg Rock, Karsten Theis, Christoph Weidenbach, and Patrick Wischnewski. Model-based variant management with v.control. In Richard Curran, Nel Wognum, Milton Borsato, Josip Stjepandic, and Wim J. C. Verhagen, editors, *Transdisciplinary Lifecycle Analysis of Systems - Proceedings of the 22nd ISPE Inc. International Conference on Concurrent Engineering, Delft, The Netherlands, July 20-23, 2015*, volume 2 of *Advances in Transdisciplinary Engineering*, pages 194–203. IOS Press, 2015.

[12] Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.

[13] Benjamin Kaufmann, Nicola Leone, Simona Perri, and Torsten Schaub. Grounding and solving in answer set programming. *AI Magazine*, 37(3):25–32, 2016.

[14] Zurab Khasidashvili, Konstantin Korovin, and Dmitry Tsarkov. Epr-based k-induction with counterexample guided abstraction refinement. In *Global Conference on Artificial Intelligence, GCAI 2015, Tbilisi, Georgia, October 16-19, 2015*, volume 36 of *EPiC Series in Computing*, pages 137–150. EasyChair, 2015.

[15] Konstantin Korovin. iprover - an instantiation-based theorem prover for first-order logic (system description). In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, volume 5195 of *Lecture Notes in Computer Science*, pages 292–298. Springer, 2008.

[16] Konstantin Korovin. Inst-gen - A modular approach to instantiation-based automated reasoning. In Andrei Voronkov and Christoph Weidenbach, editors, *Programming Logics - Essays in Memory of Harald Ganzinger*, volume 7797 of *Lecture Notes in Computer Science*, pages 239–270. Springer, 2013.

[17] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

[18] Evgeny Kruglov and Christoph Weidenbach. Superposition decides the first-order logic fragment over ground theories. *Mathematics in Computer Science*, 6(4):427–456, 2012.

[19] William Mccune. A davis-putnam program and its application to finite first-order model search: Quasigroup existence problems. Technical report, Argonne National Laboratory, 1994.

[20] G. Nelson and D.C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, October 1979.

[21] Robert Nieuwenhuis, Albert Oliveras, and Cesare Tinelli. Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll(t). *Journal of the ACM*, 53:937–977, November 2006.

[22] Ruzica Piskac, Leonardo Mendonça de Moura, and Nikolaj Bjørner. Deciding effectively propositional logic using DPLL and substitution sets. *Journal of Automated Reasoning*, 44(4):401–424, 2010.

[23] Stephan Schulz. System Description: E 1.8. In Ken McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Proceedings of the 19th LPAR, Stellenbosch*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013.

[24] John K. Slaney and Timothy Surendonk. Combining finite model generation with theorem proving: Problems and prospects. In Franz Baader and Klaus U. Schulz, editors, *Frontiers of Combining Systems, First International Workshop FroCoS 1996, Munich, Germany, March 26-29, 1996, Proceedings*, volume 3 of *Applied Logic Series*, pages 141–155. Kluwer Academic Publishers, 1996.

[25] Martin Suda, Christoph Weidenbach, and Patrick Wischnewski. On the saturation of yago. In

Automated Reasoning, 5th International Joint Conference, IJCAR 2010, volume 6173 of *LNAI*, pages 441–456, Edinburgh, United Kingdom, 2010. Springer.

[26] Geoff Sutcliffe. The cade atp system competition - casc. *AI Magazine*, 37(2):99–101, 2016.

[27] Andreas Teucke and Christoph Weidenbach. First-order logic theorem proving and model building via approximation and instantiation. In Carsten Lutz and Silvio Ranise, editors, *Frontiers of Combining Systems, 10th International Symposium, FroCoS 2015, Wroslav, Poland, 2015. Proceedings*, volume 9322 of *LNCS*, pages 85–100. Springer, 2015.

[28] Christoph Weidenbach. Automated reasoning building blocks. In Roland Meyer, André Platzer, and Heike Wehrheim, editors, *Correct System Design - Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday, Oldenburg, Germany, September 8-9, 2015. Proceedings*, volume 9360 of *Lecture Notes in Computer Science*, pages 172–188. Springer, 2015.

[29] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Satzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intellelligence Research (JAIR)*, 32:565–606, 2008.