Journal of Automated Reasoning
https://doi.org/10.1007/s10817-022-09617-3

A Posthumous Contribution by Larry Wos: Excerpts from an Unpublished Column

Sophie Tourret[1] · **Christoph Weidenbach**[1]

Received: 5 July 2021 / Accepted: 10 January 2022

Abstract
Shortly before Larry Wos passed away, he sent a manuscript for discussion to Sophie Tourret, the editor of the AAR newsletter. We present excerpts from this final manuscript, put it in its historic context and explain its relevance for today's research in automated reasoning.

Keywords History of automated reasoning · Reasoning by instantiation · Set of support · Puzzle · First-order logic modulo arithmetic

1 Introduction

Larry Wos wrote the president's column of the AAR Newsletter from its first installment in March 1983 till #131 in May 2020. The text below was not originally planned for the AAR newsletter, but was written for a colleague of his, to show a student how such a column can be written. Larry sent this final manuscript to Sophie Tourret in 2020 asking for feedback, but passed away before they could even start a discussion about it. We do not know if the text ever made its way to the intended recipient, but, for sure, it would have made it to a later issue of the AAR newsletter, had Larry had time to finish it. Thus, we have decided to publish excerpts from this final manuscript, put them into the historic context, and explain their relevance to automated reasoning research today.

2 The Column

The typical column by Larry contained at least two elements: anecdotes on the (early) history of automated reasoning followed by an open problem. Open problems came mainly from three areas: (finite) group theory, properties of metalogic formulations from the condensed detachment or equivalential calculus, or logic puzzles. We present three excerpts from Larry's final manuscript: two anecdotes from the early history of automated reasoning—namely on the relevance of instances and the set of support strategy—and the Candy Puzzle.

✉ Christoph Weidenbach
weidenbach@mpi-inf.mpg.de

[1] Max Planck Institute for Informatics, Saarland Informatics Campus E1 4, 66123 Saarbrücken, Germany

Published online: 01 February 2022

Finding Relevant Instances: A key technique in (refutation-based) automated reasoning is to identify relevant instances of universally quantified formulas. The unpublished column contains several remarks by Larry emphasizing the role of finding such instances.

> By definition, automated reasoning refers to a field in which a computer program is written that applies logical reasoning. A person can reason in a manner that is unavailable to an automated reasoning program, a way that is called instantiation. Instantiation is a type of reasoning in which variables can be replaced with other variables or with terms.

Instantiation of first-order logic formulas was actually the basis of early attempts to automated reasoning [12, 24] able to already automatically prove simple valid first-order formulas. While Gilmore [24] resolved quantifier alternations by so called "multiplications" of ground instances of the formula, Davis and Putnam [12] already applied Skolemization to existentially quantified variables and instantiated with ground Herbrand terms. Noneveless, citing Prawitz "the main weakness of the first programs for theorem proving was the manner of generating substitution instances" [34]. Resolution was seen as—and actually still is—a major breakthrough, because it does not require guessing ground instances. However, if the set of ground instances of clauses needed for a proof is small, then explicitely generating these ground instances is to be preferred over resolving among the clauses. This fact is reflected by state-of-the-art research where, e.g., InstGen [22] is currently the best single reasoning approach to first-order problems where a rather small set of ground instances from the input clause set suffices. InstGen is based on the instantiation of clauses guided by an abstraction of the first-order clauses to propositional clauses. However, building resolvents on input clauses can lead to exponentially shorter proofs compared to solely reasoning on instances, even for decidable fragments and small clause sets with only a few constants as the only function symbols [20, 33]. For an overview of recent reasoning methods incorporating instantiation, see [6].

> For example, if you are studying groups in which the square of every element is the identity e,and if you are employing the function $f$to mean product, then, with instantiation, you can obtain $f(f(u,v),f(u,v)) = e$from $f(x,x) = e$.Since instantiation is unavailable for an automated reasoning program, the obtaining of the preceding equation, $f(f(u,v),f(u,v)) = e$,will not occur.

The statement "instantiation is unavailable for an automated reasoning program" by Larry means that finding the "relevant" instances towards a proof is often as hard as finding the proof itself. Since validity is not decidable in equational logic, finding the right instances cannot be decidable as well, in general, hence instantiation by relevant instances is unavailable for an automated reasoning program. Technically, almost all automated reasoning calculi enable the addition of instances. The work of [22], see above, was actually an important step towards finding relevant instances through an abstraction into a decidable fragment, in this case propositional logic. For an overview on possibilities towards finding relevant instances, consider again [6]. Reasoning by SMT [32] (Satisfiability Modulo Theories) is built on these ideas, see below.

Nevertheless, group theory, and equational reasoning in general is a nice example for the power of instantiation. Ground equations can always be oriented, e.g., by a Knuth–Bendix ordering [26], and Knuth–Bendix completion terminates on a set of ground equalities and disequalities. Oriented equations are rewrite systems that are fundamental for many concepts in computer science such as the semantics of programming languages and have therefore obtained a lot of attention [11, 13, 27]. An alternative approach, typically implemented in SMT [3, 29, 32] solvers, is to decide the satisfiability of a set of ground equations by congruence closure [31]. Therefore, if a finite set of needed (ground) instances for a refutation is known in advance, then the problem of whether a conjectured equation is a consequence of other equations turns from being undecidable, in general, into a decidable problem.

```
1979 was the year that Veroff began to express an interest
in automated reasoning, and he has remained interested
ever since. In 1996, Veroff introduced his hints strategy
for directing a program's reasoning in a manner that
proved to be more effective than is weighting.
He introduced his "sketches" in 2001, an approach that
enabled him to answer various open questions.
```

Veroff introduced *hints* [42] to direct the proof search, integrated in Otter [28], the role model of all modern theorem provers, designed by Bill McCune. Otter has a sophisticated *weight system* for clauses. In its simplest form, the weight of the clause is the sum of the weights of its symbols, and clauses with a small weight are preferred for performing inferences during proof search. A clause weighting mechanism is an inherent ingredient of all of today's implementations of saturation-based theorem provers [5, 18, 36, 38, 41, 45]. The hint technique introduced by Veroff adds to the clause weighting mechanism: clauses subsumed by the hint clause or clauses subsuming the hint clause receive special treatment with respect to their weight. For example, if certain clauses sharing some abstract structure are thought to be essential in finding a proof, then a clause representing this abstract structure can be used as a hint clause. Then any clause that is subsumed, i.e., a superset of an instance of the hint clause, can be preferred for performing further inferences. In an extreme setting, hints can simulate the restriction of inferences to certain ground instances. Veroff successfully used the clause hint technique to find and analyze proofs [43] and combined it with another technique called *proof sketches*. A proof sketch is an incomplete, possibly incorrect proof that is then used to guide the search for a correct proof. A proof sketch may be obtained by an abstraction of the actual problem, for example by replacing non-variable terms with fresh variables.

This idea of restricting inferences and instantiation to certain patterns plays also an important role in SMT solving. Based on ideas from the Nelson-Oppen combination procedure [7, 9, 10, 16, 30], modern SMT solvers are highly efficient decision procedures for the ground combination of several theories, e.g., ground first-order logic with equality combined with ground arithmetic—on this topic, see also The Candy puzzle, page 5. If this approach needs to consider problems that also contain universally quantified formulas, it does so by consecutive additions of ground instances of these formulas. The research on this topic started with the Simplify system [14], where so-called *triggers* are used to generate ground instances out of universally quantified formulas. A trigger is a term and only instances of this term are used for proof search, similar to the concept of hints. Simplify makes additional use of the decidability of equational reasoning on the ground level. It builds an explicit model of the generated congruence and then prefers instances of the trigger that are related to the congruence. This idea was generalized to consider ground models not only for an equational theory, but also

for other theories for restricting the generation of ground instances [15, 23]. More recently, it has turned out that, as a last resort and if done with care, even an enumeration of possible ground instances can result in an efficient system [35]. In summary, the ideas suggested by Larry and his collaborators have influenced future research and are still further developed nowadays, in particular in the areas of first-order theorem proving and SMT solving.
The Set Of Support Strategy: Another anecdote by Larry is on the origin of the set of support strategy [47].

> In a summer visit, Willam F. Miller, Director of the Applied Mathematics Division at Argonne National Laboratory, invited John Alan Robinson. Miller introduced Robinson to Larry Wos and to Dan Carson, an introduction that had unbelievable consequences for the field that would eventually be called automated reasoning. While Robinson was visiting Argonne, he introduced his new inference rule that he called binary resolution. The introduction of the inference rule binary resolution changed the course of history for automated reasoning forever.
> [...]
> When Carson, who was a brilliant programmer in IBM assembly, learned of the new inference rule, he wrote a mechanical theorem-proving program encoding the rule. Carson and Wos used his program in an attempt to prove a trivial theorem in group theory. Carson's program was unable to find the sought-after proof. The theorem asserts that, if the square of every element $x$in the group is the identity e, then the group is a commutative group, $f(x, y) = f(y, x)$for every element in the group. Carson called me by phone and said that, if I had nothing to suggest, we were finished, and he gave me forty-five minutes to come up with something. After thirty-two minutes had elapsed, he called again, and I told him about the set of support strategy. Carson eagerly asked for permission to extend his program so that the program could apply the new strategy. The rest is history: with the extended program, the sought-after proof was found and found in less than 3 CPU-seconds. If Carson had not been so impatient, the use of some type of strategy would never have occurred. He is a hero for, without strategy, proofs would almost never be found with a program that reasons logically.
> [...]
> Learning about the set of support strategy, Robinson set about to prove that the strategy is refutation complete.

The encoding of the group theory used at that time by Wos and his collaborators was the following [46, 47]: the group operation is modeled by a ternary predicate P, where $P(x, y, z)$ stands for the group multiplication of x and y resulting in z. Using this predicate, basic properties of a group can be axiomatized. For example, associativity was represented by

clauses like $\neg P(x, y, u) \vee \neg P(y, z, v) \vee \neg P(u, z, w) \vee P(x, v, w)$ by introducing explicit result variables for the sub-products. However, a group theory axiomatization solely built on this predicate P is not complete, in general, because P is a relation and does not represent that the group operation is a function nor that it is total. For cases where this is needed, e.g., proving that a group where the square of each element is the identity is commutative, an additional binary predicate R for equality was added together with the equivalence relation axioms for R and the congruence axioms for P and used functions. For example, for the function f expressing totality by the unit $P(x, y, f(x, y))$ the congruence axiom $\neg R(x, x') \vee \neg R(y, y') \vee R(f(x, y), f(x', y'))$ was added, for the group multiplication the congruence axiom $\neg R(x, x') \vee \neg R(y, y') \vee \neg R(z, z') \vee \neg P(x, y, z) \vee P(x', y', z')$. The motivation for this encoding was to not expose the group multiplication to full equational reasoning which turned out to be far more successfull, because Knuth–Bendix completion and equational reasoning [26, 37] were not yet developed at this time. The problem was attacked by resolution through a partial axiomatization of equality, depending on what was needed for the actual problem at hand. A number of different axiomatizations following this approach can be found in [46]. Using such a formulation without specific equational reasoning, such problems are still a challenge to today's automated reasoning systems. Examples of such encodings are contained in the TPTP [40], e.g., the above-mentioned problem is called GRP001-1 in the TPTP.

The set of support strategy consists of dividing an unsatisfiable clause set into two disjoint sets N and S. Then, any resolution inference is restricted to use at least one clause from S, the set of support. Newly derived clauses are added to S. Wos together with his collaborators showed [47] that this restriction of the resolution calculus is complete if N is satisfiable. In a setting where a conjecture is to be shown from a set of satisfiable axioms, such a setup can easily be obtained by putting the axiom formulas into N and the (negated) conjecture into S. Recently, it has been shown that for completeness of the set of support strategy, it is necessary and sufficient that there exists a resolution refutation with at least one clause from S [25]. The set of support strategy has become an integral part of many approaches to automated reasoning. It is implemented in all first-order logic resolution-based theorem provers [5, 18, 36, 38, 41, 45]. In case of additional ordering restrictions, it is still complete if the set N is closed under non-redundant inferences. In the purely equational case, this boils down to the generation of all critical pairs modulo rewriting and elimination of trivial equations, and in the case of first-order logic with equality to the generation of all superposition inferences modulo redundancy [1].

The Candy Puzzle: Puzzles have a long history in motivating automated reasoning research by showing deficiencies of state-of-the-art reasoning calculi or systems. An early example is Schubert's Steamroller problem [39], a puzzle that was not easy to solve without a concept of typing and, therefore, motivated research in this direction. Larry's manuscript contained the following puzzle.

```
                    Candy Puzzle
Jane's Confections is an old-fashioned sweet shop next to
the old post office in Chicago, and sweet-toothed Illinois
residents traveled many a mile to buy chocolate and caram-
els, as they did when they were kids.
This afternoon, four locals, all on their way to collect
their children from a nearby junior high school, have pop-
ped in to buy a bag of something scrumptious. From the clu-
```

es, can you say at what time each customer called, and what weight of what sugary treat Jane sold them?

1. Tobias nipped in later than Sarah and bought one and a quarter pound more of his chosen sweet than the purchaser of toffee bought of toffee.
2. The customer who bought 3 pounds of creams was not the customer who came in as the clock struck 3.
3. The second customer of these four bought one and a half pounds more humbugs than Ursula bought of her favorite confection.
4. Virgil bought one and a half pounds more of his selected sweet than did the person who walked into the shop at 3:10 but who did not buy lemondrops.
5. The customers called at 3, 3:05, 3:10, and 3:15.
6. They bought one and three quarter pounds, three pounds, three pounds and a quarter pound, and four and a half pounds.

The first exercise is to solve the puzzle with the help of an automated reasoning system:
Challenge:

(1) Formulate the puzzle in logic and let an automated reasoning system find the solution.

Implicitly, the assumption is that all mentioned constants occur exactly once in the solution. For example, every customer does exactly one purchase, each different sweet is bought exactly once, etc. Then the puzzle boils down to a finite domain problem with 4^4 different possibilities of a single purchase. It can then be formalized in pure first-order logic—without theories—or even translated into propositional logic and solved that way. Solving the puzzle amounts to finding a model for the formalization, a task that is more difficult than finding a refutation (proof). We did a formalization in first-order logic and both SPASS [45] and Vampire [36] find a solution in less than a second thanks to splitting [44]. Also the grounding of the first-order formulization can be immediately solved by any SAT solver starting at the performance of MiniSat [19]. However, these formalizations require the explicit axiomatization of the needed arithmetic and linear-order concepts and properties involved in the puzzle. A more natural formulization is possible in a first-order logic over linear arithmetic. We could use a four-place predicate P where $P(u, x, y, z)$ means that u bought the amount x of y at time z. For the timing we use the values 300, 305, 310, and 315 and for the amount the values in quarter pounds, i.e., 7, 12, 13, and 18. Then a complete formalization is as follows. Firstly, we declare the existence of the four purchases

$$P(\text{Tobias}, at, bt, ct) \wedge P(\text{Sarah}, as, bs, cs) \quad \wedge$$
$$P(\text{Ursula}, au, bu, cu) \wedge P(\text{Virgil}, av, bv, cv)$$

where all symbols starting with a, b, or c are constants (existentially quantified variables). Then, we get from the first clue

$$1.\text{a}\ P(\text{Tobias}, x, y, z) \wedge P(\text{Sarah}, x', y', z') \rightarrow z > z'$$
$$1.\text{b}\ P(\text{Tobias}, x, y, z) \wedge P(u, x', \text{toffee}, z') \rightarrow x = x' + 5$$

where all variables x, y, z, u (with primes) are universally quantified and $\wedge$ binds stronger than $\rightarrow$. The second clue becomes

$$2\ P(c_{21}, 12, \text{creams}, c_{22}) \wedge c_{22} \neq 300$$

where the constants c_{21} and c_{22} are finally mapped by the finite domain axioms to the available timings, and persons. The clues three and four become

$$3\ P(c_{31}, au + 6, \text{humbugs}, 305) \wedge c_{31} \neq \text{Ursula}$$
$$4\ P(c_{41}, av - 6, c_{42}, 310) \wedge c_{41} \neq \text{Virgil} \wedge c_{42} \neq \text{lemondrops}$$

and the remaining clues are contained in the finite domain axioms.

$$5\ P(u, x, y, z) \rightarrow (z = 300 \vee z = 305 \vee z = 310 \vee z = 315)$$
$$6\ P(u, x, y, z) \rightarrow (x = 7 \vee x = 12 \vee x = 13 \vee x = 18)$$
$$7\ P(u, x, y, z) \rightarrow (y = \text{toffee} \vee y = \text{creams} \vee y = \text{humbugs} \vee y = \text{lemondrops})$$
$$8\ P(u, x, y, z) \rightarrow (u = \text{Tobias} \vee u = \text{Sarah} \vee u = \text{Ursula} \vee u = \text{Virgil})$$

Solving the puzzle formulated this way is a challenge. To the best of our knowledge, there is no automated reasoning tool that can, without further massage, derive a solution from this formalization. We tried the SMT solvers CVC4 [4] and Z3 [17] without success. Also, approaches based on resolution-style reasoning over constraint clauses will not succeed as long as the finite domain axioms do not get special treatment [2, 8]. One reason is the combination of constants and universally quantified variables over numbers which, together with first-order predicates, leads to a logic that is no longer compact, in general. Recall that compactness here means that for every infinite unsatisfiable set of clauses there exists always finite unsatisfiable subset. For example, the combination of first-order predicates and linear rational arithmetic already enables the definition of the natural numbers and the introduction of a single constant can then cause non-compactness [21]. The formulas

$$\begin{aligned}
&\text{Nat}(0)\\
&\text{Nat}(x) \rightarrow \text{Nat}(x + 1)\\
&x < 0 \rightarrow \neg\text{Nat}(x)\\
&0 < x < 1 \rightarrow \neg\text{Nat}(x)\\
&x > 0 \wedge \text{Nat}(x + 1) \rightarrow \text{Nat}(x)
\end{aligned}$$

define the natural numbers with respect to a universally quantified variable x ranging over the rationals using the predicate Nat. Non-compactness arises when this definition is combined with the four formulas

$$\begin{aligned}
&\text{Nat}(a)\\
&P(0)\\
&P(x) \rightarrow P(x + 1)\\
&\neg P(a)
\end{aligned}$$

for some constant a. The combination of all the formulas is unsatisfiable; however, every finite grounding of the formulas has a model. Still the above formulation of the puzzle is solvable, and even decidable because of the finite domain axioms. Further challenges are the following:

Challenge:

(2) Is the solution from the clues unique?
(3) If clue six is removed, is it possible to derive automatically the amounts for the purchases?

Conclusion: The scientific contributions by Larry Wos have shaped automated reasoning. Even in his later years, his columns continued to offer relevant insight into the past of automated reasoning and presented challenges that are still relevant to automated reasoners nowadays. Larry Wos was a founder of automated reasoning. He died on 20 August 2020.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Bachmair, L., Ganzinger, H.: Rewrite-based equational theorem proving with selection and simplification. J. Log. Comput. **4**(3), 217–247 (1994)
2. Bachmair, L., Ganzinger, H., Waldmann, U.: Refutational theorem proving for hierarchic first-order theories. Appl. Algebra Eng. Commun. Comput. **5**, 193–212 (1994)
3. Barrett, C.W., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185, pp. 825–885. IOS Press (2009)
4. Barrett, C.W., Conway, C.L., Deters, M., Hadarean, L., Jovanovic, D., King, T., Reynolds, A., Tinelli, C.: CVC4. In: Gopalakrishnan, G., Qadeer, S. (eds.) Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14–20, 2011. Proceedings, Springer, Lecture Notes in Computer Science, vol. 6806, pp. 171–177 (2011)
5. Bentkamp, A., Blanchette, J., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. Log. Methods Comput. Sci. **17**(2), 1–38 (2021)
6. Bonacina, M.P., Furbach, U., Sofronie-Stokkermans, V.: On first-order model-based reasoning. In: Martí-Oliet, N., Ölveczky, P.C., Talcott, C.L. (eds.) Logic, Rewriting, and Concurrency—Essays dedicated to José Meseguer on the Occasion of His 65th Birthday, Springer, Lecture Notes in Computer Science, vol. 9200, pp. 181–204 (2015)
7. Bonacina, M.P., Fontaine, P., Ringeissen, C., Tinelli, C.: Theory combination: beyond equality sharing. In: Lutz, C., Sattler, U., Tinelli, C., Turhan, A., Wolter, F. (eds.) Description Logic, Theory Combination, and All That—Essays Dedicated to Franz Baader on the Occasion of His 60th Birthday, Springer, Lecture Notes in Computer Science, vol. 11560, pp. 57–89 (2019)
8. Bromberger, M., Fiori, A., Weidenbach, C.: Deciding the Bernays–Schoenfinkel fragment over bounded difference constraints by simple clause learning over theories. In: Henglein, F., Shoham, S., Vizel, Y. (eds.) Verification, Model Checking, and Abstract Interpretation - 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17–19, 2021, Proceedings, Springer, Lecture Notes in Computer Science, vol. 12597, pp. 511–533 (2021)
9. Bruttomesso, R., Cimatti, A., Franzén, A., Griggio, A., Sebastiani, R.: Delayed theory combination vs. Nelson–Oppen for satisfiability modulo theories: a comparative analysis. Ann. Math. Artif. Intell. **55**(1–2), 63–99 (2009)
10. Chocron, P.D., Fontaine, P., Ringeissen, C.: Politeness and combination methods for theories with bridging functions. J. Autom. Reason. **64**(1), 97–134 (2020)
11. Comon, H., Godoy, G., Nieuwenhuis, R.: The confluence of ground term rewrite systems is decidable in polynomial time. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14–17 October 2001, pp. 298–307. Nevada, USA, IEEE Computer Society, Las Vegas (2001)
12. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3), 201–215 (1960)
13. Dershowitz, N., Plaisted, D.A.: Rewriting. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 535–610. Elsevier and MIT Press (2001)
14. Detlefs, D., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM **52**(3), 365–473 (2005)

15. de Moura, L.M., Bjørner, N.: Efficient e-matching for SMT solvers. In: Pfenning, F. (ed.) Automated Deduction—CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17–20, 2007, Proceedings, Springer, Lecture Notes in Computer Science, vol. 4603, pp. 183–198 (2007)
16. de Moura, L.M., Bjørner, N.: Model-based theory combination. Electron. Notes Theor. Comput. Sci. **198**(2), 37–49 (2008)
17. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29–April 6, 2008. Proceedings, Springer, Lecture Notes in Computer Science, vol. 4963, pp. 337–340 (2008)
18. Duarte, A., Korovin, K.: Implementing superposition in iProver (system description). In: Peltier, N., Sofronie-Stokkermans, V. (eds.) Automated Reasoning—10th International Joint Conference, IJCAR 2020, Paris, France, July 1–4, 2020, Proceedings, Part II, Springer, Lecture Notes in Computer Science, vol. 12167, pp. 388–397 (2020)
19. Eén, N., Sörensson, N.: An extensible SAT-solver. In: Giunchiglia, E., Tacchella, A. (eds.) Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5–8, 2003 Selected Revised Papers, Springer, Lecture Notes in Computer Science, vol. 2919, pp. 502–518 (2003)
20. Fiori, A., Weidenbach, C.: SCL clause learning from simple models. In: Fontaine, P. (ed.) Automated Deduction—CADE 27—27th International Conference on Automated Deduction, Natal, Brazil, August 27–30, 2019, Proceedings, Springer, Lecture Notes in Computer Science, vol. 11716, pp. 233–249 (2019)
21. Fiori, A., Weidenbach, C.: SCL with Theory Constraints. CoRR arXiv:2003.04627 (2020)
22. Ganzinger, H., Korovin, K.: New directions in instantiation-based theorem proving. In: 18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22–25 June 2003, Ottawa, Canada, Proceedings, IEEE Computer Society, pp. 55–64 (2003)
23. Ge, Y., de Moura, L.M.: Complete instantiation for quantified formulas in satisfiabiliby modulo theories. In: Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26–July 2, 2009. Proceedings, Springer, Lecture Notes in Computer Science, vol. 5643, pp. 306–320 (2009)
24. Gilmore, P.C.: A proof method for quantification theory: its justification and realization. IBM J. Res. Dev. **4**(1), 28–35 (1960)
25. Haifani, F., Tourret, S., Weidenbach, C.: Generalized completeness for SOS resolution and its application to a new notion of relevance. In: Platzer, A., Sutcliffe, G. (eds.) Automated Deduction—CADE 28—28th International Conference on Automated Deduction, Virtual Event, July 12–15, 2021, Proceedings, Springer, Lecture Notes in Computer Science, vol. 12699, pp. 327–343 (2021)
26. Knuth, D.E., Bendix, P.B.: Simple word problems in universal algebras. In: Leech, I. (ed.) Computational Problems in Abstract Algebra, pp. 263–297. Pergamon Press (1970)
27. Lankford Dallas, S.: Canonical inference. Technical Report ATP-32, Southwestern University, Georgetown, Texas (1975)
28. McCune, W.: OTTER 2.0. In: Stickel, M.E. (ed.) 10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24–27, 1990, Proceedings, Springer, Lecture Notes in Computer Science, vol. 449, pp. 663–664 (1990)
29. Monniaux, D.: A survey of satisfiability modulo theory. In: Gerdt, V.P., Koepf, W., Seiler, W.M., Vorozhtsov, E.V. (eds.) Computer Algebra in Scientific Computing—18th International Workshop, CASC 2016, Bucharest, Romania, September 19–23, 2016, Proceedings, Springer, Lecture Notes in Computer Science, vol. 9890, pp. 401–425 (2016)
30. Nelson, G., Oppen, D.C.: Simplification by cooperating decision procedures. ACM Trans. Program. Lang. Syst. **1**(2), 245–257 (1979)
31. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. J. ACM **27**(2), 356–364 (1980)
32. Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: from an abstract Davis–Putnam–Logemann–Loveland procedure to dpll(T). J. ACM **53**(6), 937–977 (2006)
33. Pérez, J.A.N., Voronkov, A.: Proof systems for effectively propositional logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12–15, 2008, Proceedings, Springer, Lecture Notes in Computer Science, vol. 5195, pp. 426–440 (2008)
34. Prawitz, D.: Commentary by the author: an improved proof procedure. In: Siekmann, J., Wrightson, G. (eds.) Automation of Reasoning: Classical Papers on Computational Logic, vol. 1, pp. 159–161. Springer (1983)
35. Reynolds, A., Barbosa, H., Fontaine, P.: Revisiting enumerative instantiation. In: Beyer, D., Huisman, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems—24th International Confer-

ence, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14–20, 2018, Proceedings, Part II, Springer, Lecture Notes in Computer Science, vol. 10806, pp. 112–131 (2018)
36. Riazanov, A., Voronkov, A.: The design and implementation of VAMPIRE. AI Commun. **15**(2–3), 91–110 (2002)
37. Robinson, J.A., Voronkov, A. (eds.): Handbook of Automated Reasoning, vol. 2. Elsevier and MIT Press (2001)
38. Schulz, S., Cruanes, S., Vukmirovic, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) Automated Deduction—CADE 27—27th International Conference on Automated Deduction, Natal, Brazil, August 27–30, 2019, Proceedings, Springer, Lecture Notes in Computer Science, vol. 11716, pp. 495–507 (2019)
39. Stickel, M.E.: Schubert's steamroller problem: formulation and solutions. J. Autom. Reason. **2**(1), 89–101 (1986)
40. Sutcliffe, G.: The TPTP problem library and associated infrastructure—from CNF to TH0, TPTP v.6.4.0. J. Autom. Reason. **59**(4), 483–502 (2017)
41. Tammet, T.: GKC: a reasoning system for large knowledge bases. In: Fontaine, P. (ed.) Automated Deduction—CADE 27—27th International Conference on Automated Deduction, Natal, Brazil, August 27–30, 2019, Proceedings, Springer, Lecture Notes in Computer Science, vol. 11716, pp. 538–549 (2019)
42. Veroff, R.: Using hints to increase the effectiveness of an automated reasoning program: case studies. J. Autom. Reason. **16**(3), 223–239 (1996)
43. Veroff, R.: Solving open questions and other challenge problems using proof sketches. J. Autom. Reason. **27**(2), 157–174 (2001)
44. Weidenbach, C.: Combining superposition, sorts and splitting. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 2, pp. 1965–2013. Elsevier and MIT Press (2001)
45. Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS version 3.5. In: Schmidt, R.A. (ed.) Automated Deduction—CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2–7, 2009. Proceedings, Springer, Lecture Notes in Computer Science, vol. 5663, pp. 140–145 (2009)
46. Wos, L., Carson, D.F., Robinson, G.A.: The unit preference strategy in theorem proving. In: Proceedings of the 1964 fall Joint Computer Conference, part I, AFIPS 1964 (Fall, part I), San Francisco, California, USA, October 27–29, 1964, ACM, pp. 615–621 (1964)
47. Wos, L., Robinson, G.A., Carson, D.F.: Efficiency and completeness of the set of support strategy in theorem proving. J. ACM **12**(4), 536–541 (1965)