Kalpa Publications in Computing

Volume 1, 2017, Pages 124–134

LPAR-21S: IWIL Workshop
and LPAR Short Presentations

Set of Support for Theory Reasoning

Giles Reger[1] and Martin Suda[2*]

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

Abstract

This paper describes initial experiments using the set of support strategy to improve how a saturation-based theorem prover performs theory reasoning with explicit theory axioms. When dealing with theories such as arithmetic, modern automated theorem provers often resort to adding explicit theory axioms, for example $x + y = y + x$. Reasoning with such axioms can be explosive. However, little has been done to explore methods that mitigate the negative impact of theory axioms on saturation-based reasoning. The set of support strategy requires that all inferences involve a premise with an ancestor in a so-called set of support, initially taken to be a subset of the input clauses, usually those corresponding to the goal. This leads to completely goal orientated reasoning but is incomplete for practical reasoning (e.g. in the presence of ordering constraints). The idea of this paper is to apply the set of support strategy to theory axioms only, and then to explore the effect of allowing some limited reasoning within this set. The suggested approach is implemented and evaluated within the Vampire theorem prover.

1 Introduction

In this paper we discuss how the set of support strategy can be utilised to improve theory reasoning with explicit theory axioms in saturation-based theorem provers. The addition of theory axioms has been shown to be surprisingly effective at proving many non-ground problems combining the theory of uninterpreted functions, arithmetic and arrays (i.e. those found in TPTP and SMTLIB). However, the use of theory axioms is limited by their explosive nature in the search space; reasoning with theory axioms can often be dominated by the generation of irrelevant theory consequences. This is where the set of support strategy, and the idea it represents, can help. By limiting the way in which theory axioms interact with each other, we hope to preserve their benefits whilst avoiding some of their disadvantages.

Reasoning with first-order quantifiers and first-order theories such as arithmetic must be necessarily incomplete for theoretical reasons and is very hard in practice. There are various successful approaches that we do not consider in this work as we are interested in improving theory axiom reasoning which has been shown to be useful both independently of, and in complement to, other approaches.

*This author was partially supported by ERC Starting Grant 2014 SYMCAR 639270 and the Austrian research projects FWF S11403-N23 and S11409-N23.

The contributions of this paper are (i) an initial study showing that removing theory axioms from the set of support can improve theory reasoning, and (ii) a new strategy inspired by set of support that performs *partial* saturation of the theory axioms in a dynamic manner to a given depth. In the remainder of this introduction we give an example-led overview of these contributions. The rest of the paper will give the details and evidence. Section 2 provides the relevant background, Section 3 describes our new idea and Section 4 gives some concluding remarks.

To reason in theories, such as arithmetic, saturation-based theorem provers, such as Vampire [12], often add *theory axioms*. For example, to prove the unsatisfiability of

$$f(1+a) < a \qquad x < f(x+1)$$

we can use the axioms

$$\neg(x < x) \qquad x + y = y + x \qquad (x < y \wedge y < z) \rightarrow x < z$$

to produce the following proof

$$\dfrac{\dfrac{\dfrac{x + y = y + x \qquad x < f(x+1)}{x < f(1+x)} \qquad \dfrac{(x < y \wedge y < z) \rightarrow x < z \qquad f(1+a) < a}{\neg(x < f(1+a)) \vee x < a}}{a < a} \qquad \neg(x < x)}{\bot}$$

We could have chosen different axioms to produce a different proof and as arithmetic cannot be captured by a finite set of axioms we need to choose which axioms we want to add. The above three seem to be reasonable base axioms in this case. Notice that these axioms were good enough in the sense that we never derive consequences of axioms themselves. However, during proof search it might be possible (with a poorly chosen selection function) to start to derive consequences such as

$$\neg(x < y) \vee \neg(y < x)$$

and (less usefully)

$$\neg(x_0 < x_1) \vee \neg(x_2 < x_0) \vee \neg(x_1 < x_3) \vee \neg(x_4 < x_5) \vee \neg(x_3 < x_4) \vee \neg(x_5 < x_2)$$

which are very unlikely to be helpful for proof search. In general, symmetry and transitivity are explosive axioms. This observation motivates our approach where we prevent inferences between theory axioms using the set of support mechanism.

However, sometimes these are needed. For example, the problem `ARI176=1` from TPTP [20] is equivalent to the clause

$$3x + 5y \neq 22$$

which can be proved unsatisfiable[1] using the theory axioms

$$x + y = y + x \quad x + (y + z) = (x + y) + z \quad x * 1 = x \quad x * (y + z) = (x * y) + (x * z)$$

by first deriving the following consequence

$$\dfrac{\dfrac{x * 1 = x \qquad x * (y + z) = (x * y) + (x * z)}{x * (1 + y) = x + (x * y)} \qquad x + (y + z) = (x + y) + z}{(x * (1 + y)) + z = x + ((x * y) + z)}$$

[1] Using Vampire in default mode.

but cannot be solved using Vampire's competition mode without performing any inferences between theory axioms. This suggests that it is necessary (in some sense independently of the theory axioms selected) to allow for a more fine-grained control over how theory axioms are allowed to interact. This motivates the idea of partial theory axiom saturation.

2 Background

2.1 First Order Logic with Theories

We consider a many-sorted first-order logic with equality. A term is either a variable, a constant, or a function symbol applied to terms. A literal is either a propositional symbol, a predicate applied to terms, an equality of two terms, or a negation of either. Function and predicate symbols are *sorted* i.e. their arguments (and the return value in the case of functions) have a unique *sort* drawn from a finite set of sorts S. We only consider well-sorted literals. There is an equality symbol per sort and equalities can only be between terms of the same sort. Formulas may use the standard notions of quantification and logical connectives, but in this work we assume all formulas are *clausified* using standard techniques. A *clause* is a disjunction of literals where all variables are universally quantified (existentially quantified variables can be replaced by Skolem functions during clausification).

An *interpretation* $\mathcal{I}$ assigns a non-finite domain D_s to every sort $s \in S$, maps every term t of sort $s \in S$ to a value in D_s, and assigns Boolean values to atoms. An interpretation is a *model* for a set of clauses if for each clause $L_1 \vee \ldots \vee L_n$ it assigns *true* to at least one literal L_i. A *theory* $\mathcal{T}$ constraints the set of viable interpretations by fixing interpretations for some part of the signature, which we refer to as *interpreted*. For example, the theory of integer arithmetic may fix the interpretation of a selected sort s_{int} to (be isomorphic to) the set of mathematical integers $\mathbb{Z}$ and analogously assign the usual meanings to $\{+, -, <, >, *\}$. An interpretation is consistent with a theory $\mathcal{T}$ if it satisfies that theory's constraints and is consistent with a *combination* of theories if it satisfies the constraints of each theory.

2.2 Saturation Based Theorem Proving

Theorem provers such as Vampire are refutational and saturation-based. The idea is that an input formula of the form $\mathit{Premises} \rightarrow \mathit{Conjecture}$ is negated, to give $\mathit{Premises} \wedge \neg \mathit{Conjecture}$, then clausified to produce a set of clauses S. This set is then *saturated* with respect to some inference system $\mathcal{I}$ meaning that for every inference from $\mathcal{I}$ with premises in S the conclusion of the inference is also in S. If the saturated set S contains a contradiction then the initial

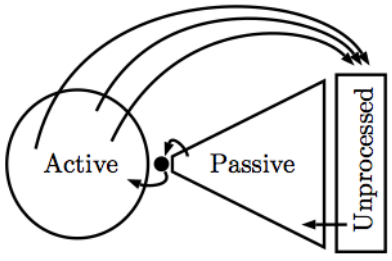

Figure 1: Illustrating the Given Clause Algorithm.

formula is necessarily valid. Otherwise, if $\mathcal{I}$ is a complete inference system, and importantly the requirements for this completeness have been preserved, then S is satisfiable and the input formula is not valid. Finite saturation may not be possible and a lot of research over the past 50 years has focussed on how to control this saturation-based proof search to make finding a contradiction more likely. This is also the focus of this paper.

The standard approach to saturation is the *given-clause* algorithm illustrated in Figure 1. The idea is to have an active set of clauses with the invariant that all inferences between active clauses have been performed and a passive set of clauses waiting to be activated. The algorithm then iteratively selects a *given clause* from passive and performs all necessary inferences to add it to active. The process of clause selection is important but not discussed here.

Vampire uses resolution and superposition as its inference system $\mathcal{I}$. A key feature of this calculus is the use of *literal selection* and *orderings* to restrict the application of inference rules, thus restricting the growth of the clause sets. There are well known conditions on literal selection that preserve completeness.

Another very important concept related to saturation is the notion of *redundancy*. It is not key to this paper so we do not discuss the details but the idea is that some clauses in S are *redundant* in the sense that they can be safely removed from S without changing the completeness result. The notion of saturation then becomes saturation-up-to-redundancy.

As a final note, from this discussion it may appear that completeness of $\mathcal{I}$ is important. But as we showed in our recent work on literal selection [10], breaking the conditions required for completeness can be helpful in proof search. Additionally, the focus of this paper is in theory reasoning, where the previous completeness arguments no longer hold.

2.3 Theory Reasoning

Here we focus on approaches that target problems combining theories and quantifiers i.e. we ignore purely ground theory reasoning as this is not the focus of this work.

There are two directions of research in the area of reasoning with problems containing quantifiers and theories. The first is the extension of SMT solvers with *instantiation* heuristics such as E-matching [8, 7]. The second is the extension of first-order reasoning approaches with support for theory reasoning. There have been a number of varied attempts in this second direction with some approaches extending various calculi [3, 9, 11, 18, 1, 6, 5] or using a SMT solver to deal with the ground part of the problem [15, 16].

In Vampire theory reasoning is currently dealt with in four ways:

1. Ground evaluation of theory terms e.g. $1 + 2$ is evaluated to 3 and $1 < 2$ evaluated to *true*,
2. Relevant theory axioms are added based on the signature of the problem i.e. if the problem uses $+$ then theory axioms such as $x + y = y + x$ and $x + 0 = x$ are added,
3. The AVATAR modulo theories approach incorporates an SMT solver to perform ground theory reasoning, see [16],
4. Simple heuristic instantiation is performed to delete literals e.g. $x < 0 \vee p(x)$ is instantiated to generate $p(1)$.

It is the second of these approaches (theory axioms) that we focus on in this paper. Notice that Vampire primarily performs non-ground theory reasoning via theory axioms as the instantiation methods implemented in Vampire are currently quite simplistic. The theory axioms

$$
\begin{array}{ccc}
x + (y + z) = (x + y) + z & x + 0 = x & x + y = y + x \\
-(x + y) = (-x + -y) & - - x = x & x + (-x) = 0 \\
x * 0 = 0 & x * (y * z) = (x * y) * z & x * 1 = x \\
x * y = y * x & (x * y) + (x * z) = x * (y + z) & \neg(x < y) \vee \neg(y < z) \vee \neg(x < z) \\
x < y \vee y < x \vee x = y & \neg(x < y) \vee \neg(y < x + 1) & \neg(x < y) \vee x + z < y + z \\
\neg(x < x) & x < y \vee y < x + 1 \ (\textit{for ints}) & x = 0 \vee (y * x)/x = y \ (\textit{for reals})
\end{array}
$$

Figure 2: Theory Axioms currently added by Vampire based on the problem signature.

Vampire might add are given in Figure 2.[2] It is important to note that Vampire normalises input formulas by rewriting all inequalities in terms of $<$ and to rewrite difference in terms of addition and unary minus.

However, theory axiom reasoning is explosive. As a simple demonstration, we take the TPTP problem `SYN000=2`, which is designed to use the full set of theory functions and predicates provided by TPTP and perform two small experiments. Firstly, proving this problem in default mode in Vampire generates of 223 clauses, 90 of which are theory consequences (40 %), of which only 1 is used in the proof. Secondly, we negate the conjecture to make it satisfiable and run Vampire in default proving mode for 10 seconds. After 10 seconds Vampire has generated 456 973 clauses, of which 449 493 are consequences of the theory axioms, over 98 %. This supports our claim that theory reasoning is explosive and establishing this claim experimentally is something we plan to do shortly.

2.4 Set of Support

The set of support strategy was introduced in [22, 14] as a method for restricting the possible inferences. The idea is to split the clauses into a *set of support* and the rest, and then restrict inferences so that they must include at least one clause that is in the set of support, with new clauses being added to the set of support. Another way of phrasing this is that all inferences must have a clause in the initial set of support as an ancestor.

If this description feels familiar then it should as the above given-clause algorithm is based on the set-of-support strategy. Here the set of support is the passive set and the rest of the clauses must be immediately added to the active set, breaking the invariant that all inferences between clauses in this set have been performed. This immediately breaks the condition required for completeness. There are arguments for how to preserve completeness using the set of support strategy, but as they generally restrict the application of techniques that we find to be essential for practical reasoning, we do not consider them. Note that our context of theory reasoning also introduces larger barriers for completeness. We are interested in how this strategy can be used to prevent the explosion of theory axioms.

In the general set of support strategy it is not prescribed which clauses should be added to the set of support, but it is reasonable to assume that it should at least include the goal, thus encouraging goal-directed reasoning. This is the approach taken in Vampire where only the conjecture is added to the set of support. We note here that this means that set of support can only be used for problems containing a conjecture which excludes many problems in TPTP and all problems in SMTLIB [4].

[2] We have ignored some less standard operators such as floor and ceiling operators. We also ignore integer division for simplicity of exposition.

Table 1: Results of an experiment looking at the usefulness of set of support strategy.

	competition mode	competition mode with `sos=off`
Solved	11 948	11 613
Uniques	422	87

In Vampire there are three set of support related options:

- `off`: do not perform set of support reasoning,
- `on`: perform set of support reasoning performing standard literal selection,
- `all`: perform set of support reasoning selecting all literals in those clauses initially added to active.

The last option helps mitigate some of the incompleteness introduced by the set of support strategy.

As a quick demonstration of the general usefulness of the set of support strategy we conducted the following experiment.[3] We ran Vampire in competition mode (CASC mode) on the full TPTP library and then ran it again forcing `sos=off`. The results of this experiment are given in Table 1. This shows that the set of support strategy was needed to solve 422 extra problems. Conversely, when it was switched off, Vampire was able to solve an additional 87 problems, most likely due to the additional time available after discarding certain strategies.

3 Set of Support for Theory Reasoning

In this section we describe our idea to use the set of support strategy to control the explosive nature of theory axioms.[4]

3.1 Special Treatment for Theory Axioms

The basic idea is to treat the given problem as the set of support in its entirety and only place the theory axioms straight into the active set, thus preventing inferences between theory axioms. This has the advantage that it applies to any theory problem, including those that do not include a specific conjecture. We call this option `sos=theory`.

This was generally straightforward to implement, with one exception. The set of support strategy in Vampire predates AVATAR [21, 17]. One element of AVATAR is that activated clauses can be deactivated i.e. placed back into passive. When AVATAR and the set of support strategy are combined this means that clauses which are initially placed in the active set might end up being reintroduced into the passive set. We have not explored the effect of this on general reasoning but in this case we enforced the condition that theory axioms should not participate in inferences with each other by directly deleting such consequences as they are generated.

Given this implementation, we carried out an experiment on all relevant problems from SMTLIB (i.e. those containing theories and quantifiers) using the default Vampire strategy

[3] All such experiments in this paper make use of StarExec [19] and version 4.1 of Vampire.

[4] We note that we do not discuss any related work that considers alternative methods for restricting the explosion of axioms. For example, Löchner's work on restricting redundant inferences with AC axioms [2, 13]. Considering whether such work is relevant to this approach remains further work.

Table 2: Results of an experiment exploring the `sos=theory` option.

	default mode	default mode + `sos=theory`
Solved	30 951	30 965
Uniques	530	227

and a 60 second time limit. Table 2 gives the results of this experiment. As expected, the total number of problems solved decreases, but there are a reasonable number of unique solutions, suggesting that this approach is helpful in general.

3.2 How Deep is Theory Reasoning?

The next obvious step is to allow some limited inferences between theory axioms. However, before we do this, we should understand how theory axioms are used in proofs. We want to know how *deep* theory reasoning is to understand how many inferences between theory axioms we actually need.

To understand this question we ran Vampire in default mode[5] across SMTLIB and analysed every proof, recording how many inferences between theory axioms were required. The results are given in the below table.

Theory axiom depth	count
none	28 409
0	1732
1	209
2	304
3	200
4	49
5	21
6	27

The table shows that theory reasoning is, in general, not very deep. Note that depth is not the same as the number of inferences requiring theory axioms (which we did not count). The depth reported in the table is the maximum derivation depth over clauses derived purely from theory axioms which occur in the proof. An important observation from the above table is that over 90% of solutions were obtained using a default strategy with no calls to an SMT solver and no theory axioms i.e. the only theory-specific feature used evaluation.

The next interesting question is what these deep theory consequences look like. Many are similar to this theory consequence

$$0 < x \vee x < 4$$

derived in a proof of `UFLIA/sledgehammer/TwoSquares/z3.637729.smt2`. Here the number 4 was generated via the application of theory axioms and the use of theory axioms to generate useful instances seems to be a common theme. A similar case is seen for the following deep theory consequences

$$\neg((x + (y + ((-x) + 2.0))) < y) \quad \text{and} \quad \neg(2.0 + x < x)$$

[5] Using the strategy `-sa discount --input_syntax smtlib2 -t 60`. We use discount saturation loop for stability as the default `lrs` option is non-deterministic.

Table 3: Results from an experiment looking at different partial saturation thresholds.

	Count when threshold =							union
Theory axiom depth	0	1	2	3	5	10	∞	
none	27 553	28 483	28 490	28 487	28 577	28 476	28 409	
0	3142	1943	1813	1757	1769	1746	1732	
1		550	237	209	217	208	209	
2			550	315	310	307	304	
3				312	254	213	200	
4					69	48	49	
5					61	21	21	
6						27	27	
total without none	3142	2493	2600	2593	2680	2570	2542	3785
uniques	866	43	27	17	90	2	4	
total with none	30 695	30 976	31 090	31 080	31 257	31 046	30 951	31 784
uniques	153	67	54	27	135	5	2	

derived in a proof of `NRA/keymaera/ETCS-essentials-live-range2.proof-node1388.smt2`. This suggests that some deep reasoning with theory axioms can be replaced by targeted instantiation methods. Integrating more intelligent instantiation techniques into Vampire is an area we are currently researching and we hope that these techniques can complement theory axiom reasoning.

3.3 Partial Saturation of Theory Axioms

Motivated by some examples and an intuition that it would be helpful, we implemented a method for partially saturating the set of theory axioms. Based on the above experiment demonstrating that theory reasoning was generally shallow, we wanted a method that gave us fine-grained control over how deep theory reasoning became. Our solution is to extend the previous idea to delete clauses with theory axioms as parents.

We attach a flag and a counter to each clause where the flag indicates whether the clause is a pure theory consequence and the counter counts the maximum number of inference steps between the clause and a theory axiom, i.e. the clause's depth in the proof. Inferences are updated to preserve and update these values as appropriate. Whenever we generate a pure theory consequence with depth larger than a given threshold we delete this clause.

Motivated by the above results we then evaluate this new approach using various thresholds. The results of this experiment, again on relevant problems drawn from SMTLIB, are given in Table 3. We initially exclude solutions that make no use of theory axioms and report these at the end separately. Here setting a threshold to 5 solved more problems than the previous two strategies of `sos=theory` and default theory axiom reasoning. This was not surprising as our previous experiment suggested that theory reasoning beyond this level was not useful. The slight variations between the 10 and ∞ cases are a demonstration of the non-deterministic nature of Vampire where techniques such as the limited resource strategy depend on the time taken to make certain proof steps. It is interesting to note that the threshold value affects the number of problems solved without theory axioms. This is due to the presence of theory axioms impacting the way in which the search space grows. However, it is not clear why a threshold of 0 is not optimal in this case. This is something we will explore further.

Table 4: Results from an experiment applying this idea to SMTLIB competition mode

	competition mode	competition mode + `sos=theory threshold=5`
Solved	37 009	36 821
Uniques	254	66

Anecdotally, we can make some further observations, which we plan to explore experimentally later:

- Frequently, problems solvable with threshold n are still solvable with threshold $m < n$; this is already clear from the results in Table 3.
- Decreasing the threshold can dramatically *decrease* the amount of time it takes to find a proof and the length of the proof found. This is what we hoped and our hypothesis is that much unnecessary theory reasoning is being avoided.
- Decreasing the threshold can dramatically *increase* the amount of time it takes to find a proof and the length of the proof found. Our hypothesis here is that reasoning that might have been done with theory axioms is now being done with non-pure consequences, which takes more effort.
- The reasoning depth reported does not account for evaluation. Some problems had very long evaluation chains leading to depths of over 300. We chose to exclude evaluation as the case of performing multiple evaluations in sequence is not interesting.

To further evaluate these ideas we forced these new options on top of the SMTCOMP competition mode and ran Vampire for the competition limit of 30 minutes on all of the relevant problems in SMTLIB. The results are given in Table 4. This shows that this new option is able to solve problems previously unsolvable by the competition mode, a significant result. Notice that the overall performance is worse when using the new option. This is because the competition mode makes use of theory reasoning techniques that are not turned on in default mode. This suggests that the threshold value should be different when using these other options. Exploring the interaction between this new option and other theory reasoning approaches (mainly the AVATAR modulo theories work [16]) remains further work.

4 Conclusion

This paper described a new approach to theory reasoning that leverages the set of support strategy to tame the explosive nature of theory axioms. The idea is to prevent inferences between theory axioms of a certain depth. Initial experiments showed that this idea can be very useful.

More work remains to full explore this idea. Directions we plan to explore include:

- Experimentally confirming the hypothesis that theory axiom reasoning is explosive.
- Understanding the relationship between these new options and other theory reasoning strategies, for example, AVATAR modulo theories, and other proof search parameters in general.

- Analysing in greater depth the kind of deep theory consequences that are generated and analysing whether it would be sensible to include directly consequences that are used often in proofs.
- Exploring whether all theory axioms should be treated equally in this approach. For example, perhaps some theory axioms are less explosive and should be allowed to partake in deeper reasoning.

The successful features described in this paper will be available in the next public release of Vampire.

References

[1] Ernst Althaus, Evgeny Kruglov, and Christoph Weidenbach. Superposition modulo linear arithmetic SUP(LA). In Silvio Ghilardi and Roberto Sebastiani, editors, *Frontiers of Combining Systems, 7th International Symposium, FroCoS 2009, Trento, Italy, September 16-18, 2009. Proceedings*, volume 5749 of *Lecture Notes in Computer Science*, pages 84–99. Springer, 2009.

[2] Jürgen Avenhaus, Thomas Hillenbrand, and Bernd Löchner. On Using Ground Joinable Equations in Equational Theorem Proving. *Journal of Symbolic Computation*, 36(1-2):217–233, 2003.

[3] Leo Bachmair, Harald Ganzinger, and Uwe Waldmann. Refutational theorem proving for hierarchic first-order theories. *Appl. Algebra Eng. Commun. Comput.*, 5:193–212, 1994.

[4] Clark Barrett, Aaron Stump, and Cesare Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). www.SMT-LIB.org, 2010.

[5] P. Baumgartner and U. Waldmann. Hierarchic Superposition With Weak Abstraction. In M.P. Bonacina, editor, *Proceedings of the 24th International Conference on Automated Deduction*, number 7898 in Lecture Notes in Artificial Intelligence, pages 39–57. Springer-Verlag, 2013.

[6] Maria Paola Bonacina, Christopher Lynch, and Leonardo Mendonça de Moura. On deciding satisfiability by theorem proving with speculative inferences. *J. Autom. Reasoning*, 47(2):161–189, 2011.

[7] Leonardo Mendonça de Moura and Nikolaj Bjørner. Efficient e-matching for SMT solvers. In *Automated Deduction - CADE-21, 21st International Conference on Automated Deduction, Bremen, Germany, July 17-20, 2007, Proceedings*, pages 183–198, 2007.

[8] David Detlefs, Greg Nelson, and James B. Saxe. Simplify: a theorem prover for program checking. *J. ACM*, 52(3):365–473, 2005.

[9] Harald Ganzinger and Konstantin Korovin. Theory instantiation. In Miki Hermann and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006, Proceedings*, volume 4246 of *Lecture Notes in Computer Science*, pages 497–511. Springer, 2006.

[10] Kryštof Hoder, Giles Reger, Martin Suda, and Andrei Voronkov. Selecting the selection. In Nicola Olivetti and Ashish Tiwari, editors, *Automated Reasoning: 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 – July 2, 2016, Proceedings*, pages 313–329, Cham, 2016. Springer International Publishing.

[11] Konstantin Korovin and Andrei Voronkov. Integrating linear arithmetic into superposition calculus. In Jacques Duparc and Thomas A. Henzinger, editors, *Computer Science Logic, 21st International Workshop, CSL 2007, 16th Annual Conference of the EACSL, Lausanne, Switzerland, September 11-15, 2007, Proceedings*, volume 4646 of *Lecture Notes in Computer Science*, pages 223–237. Springer, 2007.

[12] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *CAV 2013*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35, 2013.

[13] Bernd Löchner. *Advances in Equational Theorem Proving – Architecture, Algorithms, and Redundancy Avoidance*. PhD thesis, Universität Kaiserslautern, 2005.

[14] William McCune. OTTER 2.0. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, volume 449 of *Lecture Notes in Computer Science*, pages 663–664. Springer, 1990.

[15] V. Prevosto and U. Waldmann. SPASS+T. In G. Sutcliffe, R. Schmidt, and S. Schulz, editors, *Proceedings of the FLoC'06 Workshop on Empirically Successful Computerized Reasoning, 3rd International Joint Conference on Automated Reasoning*, number 192 in CEUR Workshop Proceedings, pages 19–33, 2006.

[16] Giles Reger, Nikolaj Bjorner, Martin Suda, and Andrei Voronkov. AVATAR modulo theories. In Christoph Benzmüller, Geoff Sutcliffe, and Raul Rojas, editors, *GCAI 2016. 2nd Global Conference on Artificial Intelligence*, volume 41 of *EPiC Series in Computing*, pages 39–52. EasyChair, 2016.

[17] Giles Reger, Martin Suda, and Andrei Voronkov. Playing with AVATAR. In P. Amy Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, pages 399–415, Cham, 2015. Springer International Publishing.

[18] P. Rümmer. A Constraint Sequent Calculus for First-Order Logic with Linear Integer Arithmetic. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proceedings of the 15th International Conference on Logic for Programming Artificial Intelligence and Reasoning*, number 5330 in Lecture Notes in Artificial Intelligence, pages 274–289. Springer-Verlag, 2008.

[19] Aaron Stump, Geoff Sutcliffe, and Cesare Tinelli. StarExec, a cross community logic solving service. https://www.starexec.org, 2012.

[20] Geoff Sutcliffe. The TPTP problem library and associated infrastructure. *J. Autom. Reasoning*, 43(4):337–362, 2009.

[21] Andrei Voronkov. AVATAR: The architecture for first-order theorem provers. In Armin Biere and Roderick Bloem, editors, *Computer Aided Verification*, volume 8559 of *Lecture Notes in Computer Science*, pages 696–710. Springer International Publishing, 2014.

[22] Lawrence Wos, George A. Robinson, and Daniel F. Carson. Efficiency and completeness of the set of support strategy in theorem proving. *J. ACM*, 12(4):536–541, October 1965.