Vampire 4.6-SMT System Description

Giles Reger[1], Martin Suda[3], Andrei Voronkov[1,4], Laura Kovács[2],
Evgeny Kotelnikov, Simon Robillard, Martin Riener[2], Michael Rawson[1],
Bernhard Gleiss, Jakob Rath[2], Petra Hozzova[2], Johannes Schoisswohl[1,2],
Ahmed Bhayat[1], Petra Hozzova[2], and Marton Hajdu[2]

[1] University of Manchester, Manchester, UK
[2] Institute for Information Systems, Vienna University of Technology, Austria
[3] Czech Technical University in Prague, Czech Republic
[4] EasyChair

Changes in 4.6. Since version 4.5 there have been routine improvements and two main new additions: a new set of inference rules for reasoning with arithmetic [13] and a new simplification rule called subsumption demodulation [3]. Inspired by recent experiments with randomisation [1] our portfolio mode also includes a 'fallback' mode where it randomises previously attempted strategies whilst time allows.

General Approach. Vampire [11] is an automatic theorem prover for first-order logic and implements the calculi of ordered binary resolution and superposition for handling equality as well as the Inst-gen calculus [7] and a MACE-style finite model builder [16]. Splitting in resolution-based proof search is controlled by the AVATAR architecture [15, 21]. Both resolution and instantiation based proof search make use of global subsumption [7]. It should be noted, to avoid confusion, that unlike the standard SMT approach of instantiation, Vampire deals directly with non-ground clauses via the first-order resolution and superposition calculi [18].

A number of standard redundancy criteria and simplification techniques are used for pruning the search space. The reduction ordering is the Knuth-Bendix Ordering. Internally, Vampire works only with clausal normal form. Problems are clausified during preprocessing [17]. Vampire implements many useful preprocessing transformations including the Sine axiom selection algorithm [6]. Vampire is a parallel portfolio solver, executing a schedule of complementary strategies in parallel.

Theory Reasoning. Vampire supports all logics apart from bit vectors, floating point, and strings. This is thanks to recent support for a first-class boolean sort [9], arrays [8], and datatypes [10], which are supported by special inference rules and/or preprocessing steps.However, Vampire has no special support for ground problems (see Z3 point below) and is therefore not entered into any *quantifier-free* divisions. The main techniques Vampire uses for theory reasoning are:

1. The addition of *theory axioms.* The main technique Vampire uses for non-ground theory reasoning is to add axioms of the theory. This is clearly incomplete but can be effective for a large number of problems. However, such axioms can be explosive in proof search. Vampire uses two techniques to control the use of theory axioms. Firstly, the standard *set-of-support* mechanism is employed to limit inferences between theory axioms [14]. Secondly, recent work [4] introduces the notion of *layered* clause queues that allow the clause selection process central to the saturation algorithm to concentrate on inferences that balance the use of theory axioms with input axioms.

2. AVATAR modulo theories [12] which incorporates Z3 [2] (version 4.8.7[1]) into AVATAR (in this sense Vampire is a wrapper solver). In this setup the ground part of the problem is passed to Z3 along with a propositional naming of the non-ground part (with no indication of what this names) and the produced model is used to select a sub-problem for Vampire to solve. The result is that Vampire only deals with problems that have theory-consistent ground parts. In the extreme case where the initial problem is ground, Z3 will be passed the whole problem. To reiterate, we never pass Z3 anything which is non-ground.

3. As described in [19, 18], Vampire combines new approaches to unification and instantiation with the aim of leveraging an SMT solver (Z3) for reasoning within a clause. The first idea is to lazily introduce constraints in cases where syntactic unification fails but unification modulo a theory may be possible e.g. adding $2x \neq 10$ when unifying $p(2x)$ and $\neg p(10)$. These constraints can then be dealt with by the second idea, to utilise an SMT solver to find instances of a clause where some theory constraints are satisfied e.g. $p(7)$ is such an instance of $p(x) \vee 14x \neq x^2 + 49$.

4. Recent work [13] introduces a set of new simplification rules designed inspired by limitations in the previous approach. The first rule, called Gaussian Variable Elimination eliminates variables that can be described completely in terms of other variables e.g. replacing $7x \neq 6 \vee p(y)$ by $p(7x - 6)$. The other rules handle subterm generalisation, evaluation, and cancellation.

5. For datatypes, we extend the superposition calculus with inference rules capturing distinctness, injectivity, and acyclicity of datatypes [10]. Recent work adds rules for structural induction [20, 5] that leverages AVATAR to explore multiple inductions concurrently.

Additionally, Vampire incorporates a MACE-style finite-model finding method that operates on multi-sorted problems [16] (applicable to UF only). There are only two cases where Vampire can return sat: Firstly in UF and secondly, if Vampire produces a ground problem after preprocessing it may pass this problem to Z3 and report its result (possibly sat) directly. However, this second case is not utilised in SMT-COMP.

Availability and Licensing. Please see `https://vprover.github.io/` for instructions on how to obtain Vampire and information about its licence. In the first instance, please direct any queries to the first author.

Expected Performance. Generally, Vampire should perform best in quantifier-heavy problems; if a problem is mostly-ground there is less that Vampire can achieve compared to a traditional SMT solver. We expect performance to be similar to last year.

Unsat Core Track. Vampire has entered the Unsat Core track for the first time this year. Under normal operation, Vampire will always produce a proof of unsatisfiability. The unsat core is defined as the subset of input clauses in the proof that were labelled in the input.

Parallel Track. In the parallel track Vampire will run using its parallel portfolio mode.

[1] To be precise, commit `5a1003f6ed10fc65a1cbcd2554f183714c413c7c`, with thanks to Nikolaj Bjorner for small changes to help with our integration.

References

[1] Filip Bártek and Martin Suda. Learning precedences from simple symbol features. In *7th Workshop on Practical Aspects of Automated Reasoning*, 2020.

[2] Leonardo Mendonça de Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In *Proc. of TACAS*, volume 4963 of *LNCS*, pages 337–340, 2008.

[3] Bernhard Gleiss, Laura Kovács, and Jakob Rath. Subsumption demodulation in first-order theorem proving. In *Automated Reasoning - 10th International Joint Conference, IJCAR*, 2020.

[4] Bernhard Gleiss and Martin Suda. Layered clause selection for theory reasoning (short paper). In *The 10th International Joint Conference on Automated Reasoning*, 2020.

[5] Marton Hajdu, Petra Hozzova, Laura Kovacs, Johannes Schoisswohl, and Andrei Voronkov. Induction with generalization in superpositionreasoning. In *The 13th Conference on Intelligent Computer Mathematics*, 2020.

[6] Krystof Hoder and Andrei Voronkov. Sine qua non for large theory reasoning. In *CADE-23 2011. Proceedings*, volume 6803 of *Lecture Notes in Computer Science*, pages 299–314. Springer, 2011.

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

[8] Evgenii Kotelnikov, Laura Kovács, Giles Reger, and Andrei Voronkov. The vampire and the FOOL. In *ACM SIGPLAN 2016 proceedings*, pages 37–48, 2016.

[9] Evgenii Kotelnikov, Laura Kovács, and Andrei Voronkov. A first class boolean sort in first-order theorem proving and TPTP. In *CICM 2015, Proceedings*, pages 71–86, 2015.

[10] Laura Kovács, Simon Robillard, and Andrei Voronkov. Coming to terms with quantified reasoning. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017*, pages 260–270. ACM, 2017.

[11] Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In *CAV 2013. Proceedings*, pages 1–35, 2013.

[12] Giles Reger, Nikolaj Bjorner, Martin Suda, and Andrei Voronkov. AVATAR modulo theories. In *GCAI 2016*, volume 41 of *EPiC Series in Computing*, pages 39–52. EasyChair, 2016.

[13] Giles Reger, Johannes Schoisswohl, and Andrei Voronkov. Making theory reasoning simpler. In *The 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2021.

[14] Giles Reger and Martin Suda. Set of support for theory reasoning. In Thomas Eiter, David Sands, Geoff Sutcliffe, and Andrei Voronkov, editors, *IWIL Workshop and LPAR Short Presentations*, volume 1 of *Kalpa Publications in Computing*, pages 124–134. EasyChair, 2017.

[15] Giles Reger, Martin Suda, and Andrei Voronkov. Playing with AVATAR. In *CADE-25 2015, Proceedings*, pages 399–415, 2015.

[16] Giles Reger, Martin Suda, and Andrei Voronkov. Finding finite models in multi-sorted first order logic. In *SAT 2016, Proceedings*, 2016.

[17] Giles Reger, Martin Suda, and Andrei Voronkov. New techniques in clausal form generation. In *GCAI 2016*, volume 41 of *EPiC Series in Computing*, pages 11–23. EasyChair, 2016.

[18] Giles Reger, Martin Suda, and Andrei Voronkov. Instantiation and pretending to be an SMT solver with vampire. In *SMT Workshop 2017*, volume 1889, pages 63–75, 2017.

[19] Giles Reger, Martin Suda, and Andrei Voronkov. Unification with abstraction and theory instantiation in saturation-based reasoning. In *IWIL Workshop and LPAR Short Presentations*, volume 1 of *Kalpa Publications in Computing*. EasyChair, 2017.

[20] Giles Reger and Andrei Voronkov. Induction in saturation-based proof search. In *The 27th International Conference on Automated Deduction*, 2019.

[21] Andrei Voronkov. AVATAR: the architecture for first-order theorem provers. In *CAV 2014. Proceedings*, pages 696–710, 2014.