Vampire 4.7-SMT System Description

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

[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 4.7. Since version 4.6 there have been many minor improvements throughout the codebase that should make small, but impactful improvements in proof search performance. The main scientific contribution in 4.7 are extensions to automated induction for integers [6] and recursive functions [4].

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 [18]. Splitting in resolution-based proof search is controlled by the AVATAR architecture [17, 23]. 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 [20].

A number of standard redundancy criteria and simplification techniques are used for pruning the search space. The reduction ordering is the Knuth-Bendix Ordering. Problems are clausified during preprocessing [19]. Vampire implements many useful preprocessing transformations including the Sine axiom selection algorithm [5]. 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 [16]. Secondly, recent work [2] 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 [13] which incorporates Z3 [1] (version 4.8.12[1]) into AVATAR (in this sense Vampire is a wrapper solver). In this setup the ground part of the problem is

[1] To be precise, commit `f03d756e086f81f2596157241e0decfb1c982299`, with thanks to Nikolaj Bjorner.

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 [21, 20], 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 [14] 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 [22, 3] that leverages AVATAR to explore multiple inductions concurrently.

Additionally, Vampire incorporates a MACE-style finite-model finding method that operates on multi-sorted problems [18] (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.

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 is entering this track for the second time. There are no specific improvements since last 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. Extracting this core from the proof is trivial but it may not be minimal.

Parallel and Cloud Tracks. In the parallel track Vampire will run using its parallel portfolio mode. In the cloud track each node will run Vampire with randomized portfolio on a randomized version of the problem.

Proof Exhibition. Vampire produces fine-grained proofs where a step describes deriving a new formula from premises [12]. Our proof checking utilizes other solvers (CVC5 and Z3) to validate these steps [15]. Preprocessing and AVATAR require special treatment.

References

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

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

[3] Marton Hajdu, Petra Hozzova, Laura Kovacs, Johannes Schoisswohl, and Andrei Voronkov. Induction with generalization in superposition reasoning. In *CICM*, 2020.

[4] Márton Hajdú, Petra Hozzová, Laura Kovács, and Andrei Voronkov. Induction with recursive definitions in superposition. In *FMCAD 21*, pages 1–10. IEEE, 2021.

[5] 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.

[6] Petra Hozzová, Laura Kovács, and Andrei Voronkov. Integer induction in saturation. In André Platzer and Geoff Sutcliffe, editors, *CADE 28*, volume 12699, pages 361–377. Springer, 2021.

[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, *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. Better proof output for vampire. In Laura Kovacs and Andrei Voronkov, editors, *Vampire 2016*, volume 44 of *EPiC*, pages 46–60. EasyChair, 2017.

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

[14] Giles Reger, Johannes Schoisswohl, and Andrei Voronkov. Making theory reasoning simpler. In *TACAS21*, 2021.

[15] Giles Reger and Martin Suda. Checkable proofs for first-order theorem proving. In Giles Reger and Dmitriy Traytel, editors, *ARCADE 2017*, volume 51 of *EPiC*, pages 55–63. EasyChair, 2017.

[16] Giles Reger and Martin Suda. Set of support for theory reasoning. In *IWIL + LPAR Short Presentations*, volume 1 of *Kalpa*, pages 124–134. EasyChair, 2017.

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

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

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

[20] 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.

[21] Giles Reger, Martin Suda, and Andrei Voronkov. Unification with abstraction and theory instantiation in saturation-based reasoning. In *TACAS 2018*, 2018.

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

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