EPiC Series in Computing

Volume 53, 2018, Pages 33–41

Vampire 2017. Proceedings of the 4th Vampire Workshop

Local proofs and AVATAR*

Giles Reger[1] and Martin Suda[2]

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

Abstract

With first-order interpolation as the application in mind, we study the problem of generating local proofs in theorem provers employing the AVATAR architecture. The theory is complemented by experimental results based on our implementation of the techniques in theorem prover Vampire.

1 Introduction

In the context of computational logic, interpolation is an important operation with applications in formal verification, ranging from bounded model checking [16], invariant generation [10], and testing [12] to concurrency [6]. One of the state-of-the-art approaches to generating first-order interpolants is based on processing so called local proofs [7] and combining conclusions of color-eliminating inferences [8, 5]. Such an approach to interpolation can be stated in terms of assigning colors to the symbols of the signature and local proofs are those that do not mix colors in inferences. Local proofs can be obtained by instructing a first-order prover to block inferences that would violate this rule. Such a restriction in general does not preserve completeness, but local proofs are discovered in many cases.

One of the most influential improvements of first-order theorem provers of recent years is the AVATAR architecture for clause splitting [17]. AVATAR employs a SAT solver to pick splitting branches, thus delegating the propositional essence of the given problem to the dedicated solver. This leads to an architecture which has been shown to be highly successful in practice [17, 13]. This paper deals with the question of how to adapt AVATAR to producing local proofs.

After summarising the necessary preliminaries concerning interpolation (Section 2) and AVATAR (Section 3), we observe that the most straightforward adaptation of AVATAR to produce local proofs may be quite restrictive as it does not allow the important color-eliminating inferences to happen within the first-order part of the prover (subsection 3.1). We then explore how severely the mentioned restriction impairs the strength of AVATAR for finding local proofs in practice by evaluating the current approach on a set of interpolation benchmarks based on the TPTP library (Section 4). Finally, we sketch a new approach for obtaining local proofs with AVATAR which compromises on the side of clause splitting and the use of a SAT solver to allow

*The second author was supported by the ERC Starting Grant 2014 SYMCAR 639270 and the Austrian research project FWF RiSE S11409-N23.

first-order color-eliminating inferences in AVATAR (Section 5). As we summarise in the last section (Section 6), it is not clear whether the expected gains justify the effort of completing the theoretical investigations and implementing the new approach in Vampire [9].

2 Interpolation and Local Proofs

We assume standard syntax and semantics of first-order logic, and the proof-theoretical notions of inference, derivation, and refutation [4]. We recall that *symbols* are either function symbols (including constants, which are function symbols of arity 0) or predicate symbols (excluding the equality predicate, which is assumed to be part of the logic). Given a formula F, the language $\mathcal{L}(F)$ is the set of all the formulas G whose symbols are among the symbols of F.

Definition 1 (Craig's Interpolant)**.** *Given formulas A and B such that $\models A \to B$, there is a formula I, called the* interpolant *of A and B, such that*

1. $\models A \to I$ *and* $\models I \to B$,
2. $I \in \mathcal{L}(A) \cap \mathcal{L}(B)$.

An interpolant of A and B is usually extracted from a proof of $A \to B$ of a certain form. One kind of proofs particularly relevant for practical interpolation are local proofs (also called split proofs) introduced by Jhala and McMillan [7].

Let us from now on fix the formulas A and B. We will refer to formulas in $\mathcal{L}(A) \setminus \mathcal{L}(B)$ as red, those in $\mathcal{L}(B) \setminus \mathcal{L}(A)$ as blue, and those in $\mathcal{L}(A) \cap \mathcal{L}(B)$ as grey.

Definition 2. *A derivation D in an inference system $\mathcal{I}$ is called* local *if for every inference*

$$\frac{F_1 \quad \ldots \quad F_k}{G}$$

in D either $\{F_1, \ldots, F_k, G\} \subseteq \mathcal{L}(A)$ or $\{F_1, \ldots, F_k, G\} \subseteq \mathcal{L}(B)$.

Efficient methods for extracting interpolants from local refutations have been proposed in the literature [7, 8, 5, 4]. In particular, in the line of work started by Kovács and Voronkov, the interpolant arises as a boolean combination of *color-eliminating* inferences. Such an inference has a red or blue premise, but a grey conclusion.

In practice, one can obtain local refutations by simply instructing a theorem prover to block, i.e. not perform, inferences which would violate the locality condition. For a typical calculus employed by a theorem prover, such as the resolution and superposition calculus [1, 11], this leads to the loss of completeness guarantees. In Section 4, we ask how much does the locality restriction actually influence prover's performance.

3 AVATAR

The highly successful AVATAR architecture for first-order theorem provers [17] is a modern realisation of the clause splitting techninue [18, 14] which employs a SAT solver to make splitting decisions and thus delegates to SAT solver the "propositional essence" of the given problem. In this section, we recall the basic notions behind AVATAR and explain the steps needed to adapt it for the generation of local refutations.

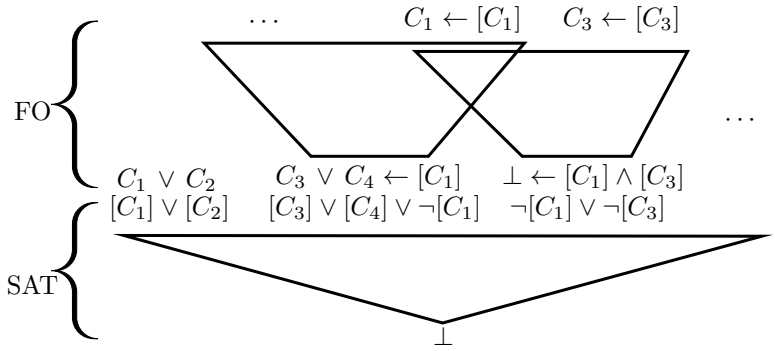

Figure 1: The general shape of an AVATAR refutation. The refutation consist of a first-order (FO) part (upper half) and a propositional (SAT) part (lower half).

Given a first-order clause C, understood as a multi-set of its literals, one considers the binary relation on the literals "to have a variable in common", denoted $\sim_C$, and defines the *components* of C as the equivalence classes of the transitive closure of $\sim_C$. We commonly write

$$C = C_1 \vee \ldots \vee C_n$$

whenever $C_1, \ldots, C_n$ are the components of C and call the clause C *splittable* if $n > 1$.

During the interaction between the first-order (FO) part of the prover and the SAT solver, an abstraction mapping $[\_]$ is used to transform a splittable first-order clause $C = C_1 \vee \ldots \vee C_n$ into a propositional clause $[C_1] \vee \ldots \vee [C_n]$. The abstraction is designed to be injective up to variable renaming, which means that in practice, it introduces a new propositional variable for every new component C_i unless C_i is a variant of a previously seen component C_j.

When accommodating AVATAR, the FO part performs *proving under assumptions*. This means each considered clause is annotated by a finite set of (propositional) assumptions and these assumptions are propagated during inferences such that the conclusion of an inference gets annotated by the union of the assumptions of the premises. This is exemplified on the following resolution inference

$$\frac{(l \vee C_1) \leftarrow A_1 \qquad (\neg l \vee C_2) \leftarrow A_2}{(C_1 \vee C_2) \leftarrow A_1 \wedge A_2},$$

where we write $C \leftarrow A$ for a clause C with assumptions A and (ab)use the symbol $\wedge$ to denote the set union operation.

The exchange between the SAT solver and the FO part consists of the following operations:

1. when the SAT solver decides that a certain component C should be active in the FO part, the *component clause* with assumptions $C \leftarrow [C]$ is inserted into the FO part,

2. when a splittable clause with assumptions $C_1 \vee \ldots \vee C_n \leftarrow [D_1] \wedge \ldots \wedge [D_m]$ is derived by the FO part, an abstracted clause $[C_1] \vee \ldots \vee [C_n] \vee \neg[D_1] \vee \ldots \vee \neg[D_m]$, which we usually call the *split clause*, is inserted into the SAT solver,

3. finally, when an empty clause with assumptions $\bot \leftarrow [D_1] \wedge \ldots \wedge [D_m]$ is derived by the FO part, a propositional clause $\neg[D_1] \vee \ldots \vee \neg[D_m]$ is inserted into the SAT solver.

When a refutation is found by AVATAR, it has the form depicted in Figure 1. We can see that the FO part is responsible for deriving the splittable first-order clauses and the (conditional) empty clauses while the SAT solver collects the corresponding propositional abstractions and derives the final (unconditional) empty clause.

To understand such a refutation as a monolithic object in a single (first-order) calculus, we just "forget" the abstraction operation and treat the symbols $\leftarrow$ and $\wedge$ as logical connectives with their standard semantics.[1] Notice that this dismisses the boundary between the FO part and SAT part of the refutation (the corresponding translation step becomes a no-op). It is also interesting to note that in this light component clauses become tautologies $C_i \vee \neg C_i$ and the originally SAT part of the refutation, which usually consists of propositional resolution steps, now possibly performs (non-standard) resolutions with general first-order formulas as pivots.

3.1 Ensuring locality of AVATAR refutations

To ensure that AVATAR produces only local refutations, we proceed as before and in the FO part of the prover block inferences which would mix colors. Note that this needs to take into account the colors of symbols in the assumptions understood as first-order formulas. For example, the last step of the following derivation will be blocked, but not because colors would be mixed in the first-order parts of the clauses, but because of mixing of colored assumptions:[2]

$$\dfrac{\dfrac{C_r \leftarrow [C_r] \qquad D \leftarrow [C_1] \wedge [C_2]}{D_1 \leftarrow [C_r] \wedge [C_1] \wedge [C_2]} \qquad \dfrac{\ldots}{D_2 \leftarrow [C_b]}}{D_3 \leftarrow [C_r] \wedge [C_1] \wedge [C_2] \wedge [C_b].}$$

Since every colored formula is introduced into the FO part annotated by itself as an assumption (e.g., $C_r \leftarrow [C_r]$) and since the FO part of any AVATAR refutation simply propagates assumptions from premises to conclusions, one can see that under the locality restriction the FO part can never contain a color-eliminating inference. As we know, it is only the conclusions of such inferences which in the end form the interpolant. In this sense, only the SAT part of such a local AVATAR refutation can be considered "interesting for interpolation".[3] In the next section, we explore to what extent is this way of imposing locality of AVATAR refutations also a problem for actually discovering them in practice.

Remark. *In order to obtain a local AVATAR refutation, also the SAT part of the refutation needs to be made local. Although local propositional refutations always exist, SAT solvers do not produce them by default. One option to overcome this problem is to localise general SAT refutation in a post processing proof transformation phase [3]. The solution currently adopted in* Vampire *also relies on post processing, but instead of transforming a SAT refutation (which is not even available in general),* Vampire *uses the technique of interpolation without proofs [2] applied to the unsatisfiable core from the SAT solver. This way, denoting the red part of the unsatisfiable core R, the blue part B, and the grey part G, the final SAT part becomes*

$$\dfrac{\dfrac{R \qquad G}{I} \qquad B}{\bot,}$$

where I is the obtained (propositional) interpolant corresponding to $A = R \wedge G$ and B.[4]

[1] We also universally close each component, i.e., treat C_i as $\forall \mathbf{X}.C_i$ where $\mathbf{X}$ are the variables of C_i.

[2] In the example derivation, formula C_r is red and formula C_b is blue.

[3] For instance, the optimisation techniques focusing on the "grey area" of the proof [5, 4] will be only relevant for the SAT part.

[4] The choice of assigning the grey propositional input formulas G to the A side is arbitrary.

Table 1: The effect of using AVATAR on finding local derivations by Vampire.

	plain		AVATAR	union
plain	7262	$\frac{+754}{-218}$	7798	8016
	$\frac{+338}{-758}$		$\frac{+371}{-887}$	$\frac{+315}{-785}$
local	6842	$\frac{+704}{-264}$	7282	7546

4 An Experiment

To evaluate the impact of AVATAR on the generation of local derivations we proceeded as follows. We took the 14 827 first-order problems from the TPTP library [15] version 6.4.0, clausified each using Vampire and split the obtained set of clauses into halves, treating the first half as A and the second as B (the same methodology for obtaining benchmarks for interpolation was used by Gleiss et al. [4]). We attempted to refute each of the obtained problems using Vampire with a single fixed strategy and time limit of 10 s.[5] On top of this base strategy, we varied whether AVATAR should be turned on or not and whether Vampire should pay heed to symbol colors and thus attempt to produce only local refutations.

The results of the experiment are summarized in Table 1. The whole numbers correspond to the number of problems solved by each strategy variation while the fractions represent the sizes of set difference between the respective results for the neighbours (explained in more detail below). The columns of the table capture the effect of turning AVATAR on (and "union" shows how much Vampire solves when the results of both strategies are combined). The rows capture the transition from a proof search without restriction ("plain") to a color-aware one ("local"). In the first row, we can thus see the strength of AVATAR in the standard unrestricted setting: here AVATAR allows Vampire to solve 754 new problems (while 218 are not solved anymore). Analogously, the first column demonstrates that focusing only on local refutations leads to a drop in performance, namely from 7262 to 6842 solved problems. Interestingly, there are 338 problems that Vampire only solves using the (incomplete) strategy which blocks inferences mixing colors.[6] Our main focus in this experiment, however, is on comparing the first row to the second. We observe that also when restricted to local derivations AVATAR helps Vampire solving many problems, but there are slightly fewer newly solved problems (704 compared to 754) and slightly more problems which cannot be solved with AVATAR anymore (264 compared to 218). This suggests that AVATAR is not as powerful when restricted to local derivations (in the way described in the previous section) as it is in an unrestricted form.

5 An Alternative Approach

In this section, we discuss the possibility of altering the AVATAR architecture such that it produces local refutations while allowing for color-eliminating inferences in the FO part. The

[5] The strategy employs the non-default discount saturation loop and age-weight ration 10 and is identified by the option string `-t 10s -sa discount -awr 10`.

[6] This suggests that (regardless of the interest in interpolation) a viable strategy for solving previously unsolved problems in Vampire is to arbitrarily split a problem in two halves, accordingly assign colors to symbols, and instruct the prover to search only for local refutations. Such a strategy could then, along with many others, be a useful part of a strategy portfolio.

core of the idea is 1) to allow at most one colored component in a splittable clause and 2) instead of annotating it with its own abstraction as usual to use (the negations of) the abstractions of the remaining components for the annotation. This way, colored symbols only appear in the first-order parts of the clauses and thus first-order color-eliminating inferences are possible.

Formally, we change the definition of a component and define it as an equivalence class of the transitive closure of the relation on clause's literals "to share a variable *or be of the same non-grey color*". For instance, instead of splitting the following clause as shown on the left, the two colored components get merged into one:

$$\underbrace{C_r^1 \vee C_r^2}_{\text{same non-grey color}} \vee C_g^3 \vee C_g^4 \quad \rightsquigarrow \quad C_r^{12} \vee C_g^3 \vee C_g^4.$$

Under this definition, we perform less splitting in general, but ensure that there is at most one colored component in each clause.[7]

Now, when the SAT solver decides that a colored component such as C_r^{12} should be active in the FO part, instead of inserting the usual component clause $C_r^{12} \leftarrow [C_r^{12}]$, we insert the following *greyified* clause

$$C_r^{12} \leftarrow \neg[C_g^3] \wedge \neg[C_g^4]. \tag{1}$$

To see that this is sound, notice that (1), can be derived from the component clause $C_r^{12} \leftarrow [C_r^{12}]$ and the corresponding split clause $[C_r^{12}] \vee [C_g^3] \vee [C_g^4]$ by resolution over the pivot $[C_r^{12}]$. The net effect of this modification is that we only deal in the FO part of the prover with clauses with grey assumptions and thus allow color-eliminating inferences in the FO part of the refutation.

5.1 Complications

The general picture is more complicated for at least two reasons. To explain the first reason, let us elaborate what it means for the SAT solver to decide that a component C should be active in the FO part. Under normal conditions, the SAT solver repeatedly looks for a model M of the propositional clauses inserted so far and a component C should be active in the FO part if and only if $M \vDash [C]$. As a consequence, AVATAR maintains an invariant that all assumptions of clauses active in the FO part are true in the current model M so that when a conditional empty clause $\bot \leftarrow [D_1] \wedge \ldots \wedge [D_m]$ is derived, the corresponding propositional clause $\neg[D_1] \vee \ldots \vee \neg[D_m]$ will be false in the current model M and force the SAT solver to look for a different model, thus ensuring progress. One way of maintaining this important invariant under the modification with colored components described above could be to refrain from inserting the usual split clause (i.e. the clause $[C_r^{12}] \vee [C_g^3] \vee [C_g^4]$ in our example) to the SAT solver and instead to have an extra rule saying that the corresponding greyified clause (i.e. clause 1) should be active in the FO part whenever the current model M makes all the remaining grey components' abstractions false (i.e. $M \vDash \neg[C_g^3] \wedge \neg[C_g^4]$ here). Notice that we would never need to register the colored component (i.e. the variable $[C_r^{12}]$) in the SAT solver.

The second reason is related to component sharing. Recall that the abstraction mapping $[\_]$ is injective up to variable renaming. This means that when we split a clause (a variant of) one of its components could already have been derived before, but under a different context. If this component is colored, we have more than one way of constructing a greyified clause for it. For example, after splitting the clauses

$$C_r \vee C_g^1 \vee C_g^2 \quad \text{and} \quad C_r \vee C_g^3 \vee C_g^4 \vee C_g^5,$$

[7] Since we never deal with clauses that would mix colors.

Table 2: Approximating the performance of the alternative approach.

	plain	AVATAR	SingleColComp	IgnAssumpCol
plain	7262	7798		
local	6842	7282	7089 (+252 / +40)	7406 (+158 / +34)

there will be two greyified clauses

$$C_r \leftarrow \neg[C_g^1] \wedge \neg[C_g^2] \qquad \text{and} \qquad C_r \leftarrow \neg[C_g^3] \wedge \neg[C_g^4] \wedge \neg[C_g^5]$$

corresponding to the component C_r and we might be forced to insert both to the FO part (in particular, if the current model makes both $\neg[C_g^1] \wedge \neg[C_g^2]$ and $\neg[C_g^3] \wedge \neg[C_g^4] \wedge \neg[C_g^5]$ true). But this seems to defeat the purpose of component sharing anyway, because originally, we were always only inserting one component clause per component.

5.2 Preliminary Experiment – Would it Pay Off?

Since overcoming the just described complications seems to require a non-trivial effort on the programming side and most likely a non-negligible overhead during execution, a question arises whether allowing color-eliminating inferences in the FO part of AVATAR refutations is worth the effort. To establish an upper bound on the potential gains of the proposed idea, we modified Vampire to always merge colored components as described above but to ignore colors in assumptions for blocking inferences (as if the assumption were guaranteed to be always grey). This modification should perform similarly with respect to the number of refutations found to the proposed idea as if implemented with zero overhead.[8]

The results of an experiment with the modified Vampire are presented in Table 2. The setup of the experiment was the same as for the one from Section 4 and the values in the left part of the table are copied for comparison from Table 1. The middle column (SingleColComp) presents the number of problems solved by the just described modified Vampire in AVATAR mode. We can see that the modified Vampire solves fewer problems than the original localising AVATAR, namely 7089 problems versus 7282. This could be explained by the fact that the modified Vampire performs fewer splits. At the same time, modified Vampire solves 252 problems not solved by the original localising AVATAR out of which 40 problems are not even solved by the original Vampire with AVATAR turned off. These problems could be accounted to the ability of the modified Vampire to perform color-eliminating inferences in the FO part.

For comparison, the last column (IgnAssumpCol) shows the performance of even less restrictive modification which keeps the original definition of a component (does not merge colored components), but still ignores colors in assumptions. This combines the advantages of more splitting with the ability to perform color-eliminating inferences in the FO part. However, we currently do not have any theory on how to obtain local refutations (perhaps in a post-processing phase) from the ones produced. In any case, although IgnAssumpCol performs better than SingleColComp in the total number of solved problems, its gains in terms of newly solved problems are (surprisingly) not so dramatic. (See the numbers in the brackets.)

[8] Of course, the produced refutations would not be local in general.

6 Summary and Discussion

In this paper we studied how to adapt the AVATAR architecture for theorem proving to produce local proofs for interpolation. We first presented the obvious solution and evaluated it experimentally. Since this solutions appears to be relatively restrictive, we proposed a direction in which to look for an alternative solution and identified obstacles which would need to be overcome to implement it. We then performed an experiment with a mock implementation in order to establish, whether investing into working out the alternative fully could pay off.

To zoom in on the details, there seem to be two ways of overcoming the complications identified in Section 5.1. We can either give up on the sharing of colored components, introducing a fresh propositional abstraction whenever a new clause containing a particular colored component is derived. This, however, may lead to unnecessary duplication of the FO prover work. Alternatively, we could simply ignore colors in assumptions and try to localise the potentially non-local refutation in a post-processing phase. This seems to be straightforward under the assumption that only one colored component per clause is present (i.e., for SingleColComp), but could also work in general (i.e., for IgnAssumpCol). The corresponding proof transformation [3], however, also conceals a blowup. We conjecture that a certain blowup is inherent.

Concerning the results of the experiments, we do not have a clear support that working out the alternative solution would dramatically increase the performance of the prover. This, however, only reflects the behaviour on our benchmark set which is arguably quite artificial from the perspective of interpolation. A final verdict should therefore be postponed and only made in relation to concrete applications of interpolation in practice.

References

[1] L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.

[2] Hana Chockler, Alexander Ivrii, and Arie Matsliah. Computing interpolants without proofs. In Armin Biere, Amir Nahir, and Tanja E. J. Vos, editors, *Hardware and Software: Verification and Testing - 8th International Haifa Verification Conference, HVC 2012, Haifa, Israel, November 6-8, 2012. Revised Selected Papers*, volume 7857 of *Lecture Notes in Computer Science*, pages 72–85. Springer, 2012.

[3] Bernhard Gleiss. Interpolation and local proofs. Master's thesis, TU Wien, 2016.

[4] Bernhard Gleiss, Laura Kovács, and Martin Suda. Splitting proofs for interpolation. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 291–309. Springer, 2017.

[5] Krystof Hoder, Laura Kovács, and Andrei Voronkov. Playing in the grey area of proofs. In John Field and Michael Hicks, editors, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 259–272. ACM, 2012.

[6] Andreas Holzer, Daniel Schwartz-Narbonne, Mitra Tabaei Befrouei, Georg Weissenbacher, and Thomas Wies. Error invariants for concurrent traces. In John S. Fitzgerald, Constance L. Heitmeyer, Stefania Gnesi, and Anna Philippou, editors, *FM 2016: Formal Methods - 21st International Symposium, Limassol, Cyprus, November 9-11, 2016, Proceedings*, volume 9995 of *Lecture Notes in Computer Science*, pages 370–387, 2016.

[7] Ranjit Jhala and Kenneth L. McMillan. A practical and complete approach to predicate refinement. In *TACAS*, volume 3920 of *LNCS*, pages 459–473. Springer, 2006.

[8] Laura Kovács and Andrei Voronkov. Interpolation and symbol elimination. 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 199–213. Springer, 2009.

[9] 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.

[10] Kenneth L. McMillan. Quantified invariant generation using an interpolating saturation prover. In C. R. Ramakrishnan and Jakob Rehof, editors, *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*, volume 4963 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2008.

[11] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier Science, 2001.

[12] Andreas Podelski, Martin Schäf, and Thomas Wies. Classifying bugs with interpolants. In Bernhard K. Aichernig and Carlo A. Furia, editors, *Tests and Proofs - 10th International Conference, TAP 2016, Held as Part of STAF 2016, Vienna, Austria, July 5-7, 2016, Proceedings*, volume 9762 of *Lecture Notes in Computer Science*, pages 151–168. Springer, 2016.

[13] Giles Reger, Martin Suda, and Andrei Voronkov. Playing with AVATAR. In Amy P. Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 399–415. Springer, 2015.

[14] A. Riazanov and A. Voronkov. Splitting without backtracking. In B. Nebel, editor, *17th International Joint Conference on Artificial Intelligence, IJCAI'01*, volume 1, pages 611–617, 2001.

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

[16] Yakir Vizel, Arie Gurfinkel, and Sharad Malik. Fast interpolating BMC. In Daniel Kroening and Corina S. Pasareanu, editors, *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I*, volume 9206 of *Lecture Notes in Computer Science*, pages 641–657. Springer, 2015.

[17] 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.

[18] C. Weidenbach. Combining superposition, sorts and splitting. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 27, pages 1965–2013. Elsevier Science, 2001.