EPiC Series in Computing

Volume 46, 2017, Pages 49–64

LPAR-21. 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning

EPiC Computing

First-Order Interpolation and Interpolating Proofs Systems*

Laura Kovács[1,3] and Andrei Voronkov[2,3,4]

[1] Vienna University of Technology
[2] The University of Manchester
[3] Chalmers University of Technology
[4] EasyChair

Abstract

It is known that one can extract Craig interpolants from so-called local derivations. An interpolant extracted from such a derivation is a boolean combination of formulas occurring in the derivation. However, standard complete proof systems, such as superposition, for theories having the interpolation property are not necessarily complete for local proofs.

In this paper we investigate interpolant extraction from non-local refutations (proofs of contradiction) in the superposition calculus and prove a number of general results about interpolant extraction and complexity of extracted interpolants. In particular, we prove that *the number of quantifier alternations in first-order interpolants of formulas without quantifier alternations is unbounded.* The proof of this result relies on a small number of assumptions about the theory and the proof system, so it also applies to many first-order theories beyond predicate logic. This result has far-reaching consequences for using local proofs as a foundation for interpolating proof systems - any such proof system should deal with formulas of arbitrary quantifier complexity.

To search for alternatives for interpolating proof systems, we consider several variations on interpolation and local proofs. Namely, we give an algorithm for building interpolants from resolution refutations in logic without equality and discuss additional constraints when this approach can be also used for logic with equality. We finally propose a new direction related to interpolation via local proofs in first-order theories.

1 Introduction

Craig interpolation [6] has found many application in formal verification, including bounded model checking [23], invariant generation [17], testing [20] and concurrency [11]. Some of the interpolation-based verification methods compute interpolants from proofs by restricting proofs to so-called local (or split) proofs [14, 15]. In particular, [16, 7, 22, 4] apply theory-specific transformations and generate quantifier-free interpolants from local SMT proofs in various theories, whereas [17, 15] build quantified interpolants from local proofs in full first-order logic.

*We acknowledge funding from the ERC Starting Grant 2014 SYMCAR 639270, the Wallenberg Academy Fellowship 2014 TheProSE, the Swedish VR grant GenPro D0497701, the Austrian FWF research project RiSE S11409-N23, and the EPSRC grant ReVeS: Reasoning for Verification and Security.

While local proofs admit efficient interpolation algorithms, see e.g. [14, 15], local proofs do not always exist.

In this paper we address limitations of local proof generation and discuss interpolation in non-local proofs using the resolution and superposition calculus. We start with preliminaries in Section 2 and introduce notions relevant to inference systems in Section 3.

McMillan's result [17] on extracting interpolants from local proofs gives an idea of designing local proof systems for various logics. Due to the generality of this result, it can be used for essentially arbitrary first-order theories. This results poses a problem of finding powerful proof systems producing local proofs for such theories, ideally *complete* proof systems.

Completeness of proof systems for theories having the interpolation property (such as first-order predicate logic) can be defined as existence of local refutations for every unsatisfiable well-colored formula (the precise definitions will be given below). For theories having no interpolation property in general, we can still define completeness as existence of local refutations for all pairs of formulas having a reverse interpolant.

It would be ideal to have such complete proof systems which can be used by existing efficient theorem provers for these theories, or their simple modifications, for example, a suitable modification of the superposition calculus for first-order predicate logic. In Section 4 we show that there is, in a way, no such modification of the superposition calculus. To this end, we prove that every complete proof system must deal with formulas having an unbounded number of quantifier alternations (remember that the superposition calculus only deals with universal formulas having no quantifier alternation at all).

We notice that this result relies on a small number of assumptions about the theory and the proof system, therefore it applies to many first-order theories beyond predicate logic.

In Section 5 we consider so-called *relational interpolants* introduced in [12]. We give an algorithm for building relational interpolants from arbitrary proofs in the superposition calculus and show that the obtained relational interpolants are boolean combinations of formulas from the proof. This result implies that relational interpolants exist for arbitrary axiomatizable theories. Our proof and construction are simpler than that of [12].

We then note in Section 6 that for logic without equality one can turn relational interpolants into interpolants by quantifying away colored terms occurring in relational interpolants. This result is not original and was stated and proved in a number of other papers. It implies that, although the size of an interpolant is, in general, non-elementary in the size of the input formulas, it is polynomial in the size of the proof, if the interpolant is represented as a dag. Our results thus yield an interpolation procedure for *arbitrary proofs in the resolution calculus* for logic without equality. We also give an example showing that this way of building interpolants *fails for logic with equality.*

Finally, in Section 7 we discuss new directions in interpolating proof systems. One interesting avenue to exploit is to weaken the soundness property of proof systems. However, in this case the result of [17] also does not apply. Therefore, we pose an open problem of extending results on local proofs and interpolation to proof systems having this weaker property and also finding a modification of the superposition calculus resulting in an interpolating proof system. We outline one possible approach to efficiently build interpolants from superposition proofs by using definitions (naming) and skolemization on the fly. It is especially interesting to have a proof system that is a simple modification of the superposition calculus, since such a system can potentially be implemented at low cost by modifying existing efficient first-order theorem provers.

We believe that our results raise fundamental questions in first-order interpolation and pose further questions for proof-based interpolant extraction in the presence of full first-order theories.

Contributions. We summarize our contributions below.

1. We prove that there is no bound on the number of quantifier alternations in interpolants of universal formulas (Theorem 4.1) in first-order predicate logic.

2. As a corollary, we prove that any complete local proof system for first-order predicate logic must have inferences dealing with formulas of an arbitrary quantifier complexity, even if the input formulas have no quantifier alternations (Theorem 4.3). In a way, this implies that there is no simple modification of the superposition calculus for logic with/without equality in which every unsatisfiable formula has a local refutation.

3. We consider so-called relational interpolants, which are weaker than interpolants. We give an algorithm for constructing relational interpolants from arbitrary superposition proofs (Lemma 5.2).

4. We show that relational interpolants can be turned into interpolants for logic without equality (Lemma 6.1). Under additional constraints on the signature of the input formulas, this results also holds for first-order logic with equality (Theorem 6.2).

2 Preliminaries

We deal with the standard first-order predicate logic with equality. The equality symbol will be denoted by $\simeq$; instead of writing $\neg(s \simeq t)$ we will simply write $s \not\simeq t$. We allow all standard boolean connectives and quantifiers in the language and, in addition, assume that it contains the logical constants $\top$ for always true and $\bot$ for always false formulas.

We will denote formulas by A, B, C, D, terms by r, s, t, variables by x, y, z, constants by a, b, c, function symbols by f, g and predicates with p, q, possibly with indices. As usual, a *literal* is an atomic formula or its negation and a *clause* is a disjunction (or a multiset) of literals. Since clauses are multisets, we do not distinguish clauses obtained by a permutation of literals. Let A be a formula with free variables $\bar{x}$, then $\forall A$ (respectively, $\exists A$) denotes the formula $(\forall \bar{x})A$ (respectively, $(\exists \bar{x})A$). A formula is called *closed*, or a *sentence*, if it has no free variables. We call a *symbol* a predicate symbol, a function symbol or a constant. Thus, variables are not symbols. We consider equality $\simeq$ part of the language, that is, equality is not a symbol. A formula or a term is called *ground* if it has no occurrences of variables. A formula is called *universal* if it has the form $(\forall \bar{x})A$, where A is quantifier-free. We write $C_1, \ldots, C_n \vdash C$ to denote that the formula $C_1 \wedge \ldots \wedge C_n \rightarrow C$ is a tautology. Note that $C_1, \ldots, C_n, C$ may contain free variables.

A *signature* is any finite set of symbols. If E is a set of expressions, for example, formulas, and constants $c_1, \ldots, c_k$ do not occur in E, then we say that $c_1, \ldots, c_k$ are *fresh* for E. We will less formally simply say *fresh constants* when E is the set of all expressions considered in the current context.

To simplify the presentation, we will use *colored* signatures, in which every symbol is colored in exactly one of the following three colors: *red*, *blue* and *grey*. We call a formula or a term *grey* if it uses only grey symbols. A formula or a term is called *red* (respectively, *blue*) if it uses only red and grey symbols and has at least one red (respectively, *blue*) symbol. Thus, every formula is either grey, or red, or blue, or both red and blue.

The symbol $\square$ will denote the end of lemmas, theorems, proofs, definitions and examples.

Theorem 2.1 (Craig's Interpolation Theorem [6])**.** Let R be a red formula, B be a blue formula and $R \vdash B$. Then there exists a closed grey formula I such that $R \vdash I$, $I \vdash B$ and every free variable of I is also a free variable in both R and B. □

Every formula I satisfying Theorem 2.1 will be called an *interpolant* of R and B.

In the sequel, R will always denote a red formula and B a blue formula. For example, if we write "let R, B are formulas", we assume that R is read and B is blue.

A *theory* is any set of grey sentences. If T is a theory, we write $C_1, \ldots, C_n \vdash_T C$ to denote that the formula $C_1 \wedge \ldots \wedge C_n \to C$ holds in all models of T. When we work with a theory T, we call symbols occurring in T *interpreted* while all other symbols *uninterpreted*. Note that interpreted symbols are always grey.

In [15] we proved that the following result holds for interpolation in theories:

Theorem 2.2. Let R be a red sentence, B be a blue sentences, T be a theory and $R \vdash_T B$. Then there exists a grey sentence I such that

1. $R \vdash_T I$ and $I \vdash B$;
2. every interpreted symbol of I occurs in B.

Likewise, there exists a grey sentence I such that

1. $R \vdash I$ and $I \vdash_T B$;
2. every interpreted symbol of I occurs in R. □

In the sequel we will be interested in the interpolation property with respect to a given theory T. For this reason, we will use $\vdash_T$ instead of $\vdash$ and relativize all definitions to T. To be precise, we call an *interpolant* of R and B any grey formula I with the properties $R \vdash_T I$ and $I \vdash_T B$.

We call a *reverse interpolant of* R *and* B any interpolant of R and $\neg B$. Interpolation-based verification papers call reverse interpolants simply interpolants, see e.g. [13, 14, 17], and change the condition $R \vdash B$ by $R, B \vdash \bot$. This makes sense when we deal with proof systems based on refutation, or satisfiability-checking, since $R, B \vdash \bot$ means that the set $\{R, B\}$ is unsatisfiable. We use both notions to avoid ambiguous terminology. Note that any reverse interpolant I of R and B has the properties $R \vdash I$ and $B \vdash \neg I$.

3 Inference Systems and Proofs

We recall some terminology from [15], related to inference systems and adapted to our setting. It is commonly used in the theory of resolution and superposition [2, 19].

Definition 3.1. An *inference rule* is an n-ary relation on formulas, where $n \geq 0$. The elements of such a relation are called *inferences* and usually written as

$$\frac{A_1 \quad \ldots \quad A_n}{A}\,. \tag{1}$$

The formulas $A_1, \ldots, A_n$ are called the *premises*, and the formula A the *conclusion*, of this inference. An *inference system* is a set of inference rules. An *axiom* of an inference system is any conclusion of an inference with 0 premises. Any inferences with 0 premises and a conclusion A will be written without the bar, simply as A.

A *derivation* in an inference system is a tree built from inferences in this inference system. If the root of this derivation is A, then we say it is a *derivation of* A. A derivation of A is called a *proof* of A if it is finite and all leaves in the derivation are axioms. A formula A is called *provable* in I if it has a proof. We say that a derivation of A is *from assumptions* $A_1, \ldots, A_m$ if the derivation is finite and every leaf in it is either an axiom or one of the formulas $A_1, \ldots, A_m$. A formula A is said to be *derivable from* assumptions $A_1, \ldots, A_m$ if there exists a derivation of A from $A_1, \ldots, A_m$. A *refutation* is a derivation of $\perp$. □

Note that a proof is a derivation from the empty set of assumptions. Any derivation from a set of assumptions S can be considered as a derivation from any larger set of assumptions $S' \supseteq S$.

An inference (1) is called *sound* with respect to a theory T if $T \vdash A_1 \wedge \ldots \wedge A_n \rightarrow A$. In other words, the conclusion of the inference is a logical consequence of its premises with respect to T. When T is empty, that is, the conclusion of the inference is a logical consequence of its premises in first-order predicate logic, we will simply say that the inference is sound. A derivation or an inference system is called *sound (with respect to a theory T)*, if so is every inference in this derivation or this system. Evidently, if an inference system is sound, then every derivation in this inference system is sound too.

In the sequel we use the standard notion of *substitutions* as mappings from variables to terms, and denote substitutions by σ, possibly with indices. We write the application of a substitution σ to an expression E (e.g., a term or a formula) as $E\sigma$. A *unifier* of two expressions E_1 and E_2 is a substitution σ such that $E_1\sigma = E_2\sigma$. It is known that if two expressions have a unifier, then they have a so-called *most general unifier (mgu)* [21].

4 Interpolation and Local Proofs

In this section we are especially interested in a special kind of derivation introduced in [14] and called *local* (they are also sometimes called *split-proofs*). We call an inference *local* if does not contain both a red and a blue formula. A derivation is called *local* if every inference in this derivation is local too.

It is known that from a local sound refutation of R, B (with respect to a theory) one can extract a reverse interpolant of R and B [14, 15] (with respect to this theory). Moreover, algorithms of [14, 15] give interpolants that are boolean combinations of certain grey formulas occurring in the refutation.

In this paper we restrict our attention to the resolution and superposition system, simply called the superposition calculus, which is both sound and refutationally complete. As we show later, by restricting the calculus to local proofs we lose completeness. Even more, we will prove that there exist no simple modification of the superposition calculus that is complete for local proofs.

The superposition calculus is a family of inference systems parametrized by a selection function on literals and a simplification ordering $\succ$ on terms, literals and clauses, see [19] for details.

For a literal L and term s, we write $L[s]$ to mean that s is a subterm of L. Further, $L[s]_p$ denotes that the subterm of C at position p is s. With these notations at hand, the inference rules of the superposition calculus for logic with equality are given in Figure 1. All inference rules in this calculus work with clauses. Note that superposition calculi used in theorem provers have extra conditions on inferences rules not shown here, however these extra conditions are irrelevant for our purpose.

Binary Resolution (BR):

$$\frac{A \vee C \quad \neg A' \vee D}{(C \vee D)\sigma}$$

(i) σ is an mgu of A and A'.

Factoring (F):

$$\frac{A \vee A' \vee C}{(A \vee C)\sigma}$$

(i) σ is an mgu of A and A'.

Superposition (Sup):

$$\frac{s \simeq t \vee C \quad L[s']_p \vee D}{(C \vee D \vee L[t]_p)\sigma}$$

(i) σ is an mgu of s and s',
(ii) s' is not a variable,
(iii) $t\sigma \not\succ s\sigma$.

Equality Factoring (EF):

$$\frac{s \simeq t \vee s' \simeq t' \vee C}{(s \simeq t \vee t \not\simeq t' \vee C)\sigma}$$

(i) σ is an mgu of s and s',
(ii) $t\sigma \not\succ s\sigma$,
(iii) $t'\sigma \not\succ s'\sigma$,
(iv) $t'\sigma \not\succ t\sigma$.

Equality Resolution (ER):

$$\frac{s \not\simeq s' \vee C}{C\sigma}$$

(i) σ is an mgu of s and s'.

Figure 1: The Superposition Calculus

We consider all formulas used in the superposition calculus implicitly universally quantified. If we use this convention, the conclusion of every rule is a logical consequence of its premises, so the calculus is a sound inference system.

We implemented search for local proofs and the algorithm of [15] in our superposition-based theorem prover Vampire [9]. In some cases interpolation in Vampire however fails to find a local derivation, even in the case when a non-local derivation exists. Let us give an example.

Example 4.1. Let R be the universal atomic formula $(\forall x)p(r, x)$, and B be $(\forall y)\neg p(y, b)$. The constant r is red, b is blue, and the predicate p is grey. A reverse interpolant I of R and B is $(\exists y)(\forall x)p(y, x)$. Note that, while R and B are universal formulas, so they contain no quantifier alternations, yet their reverse interpolant I contains quantifier alternations. One can prove that every interpolant of this formula must have at least one quantifier alternation.

The reverse interpolant I cannot be computed using existing methods and provers, including Vampire, as there is no local proof of $R, B \vdash \bot$ in the resolution/superposition calculus. No inference rule of the superposition calculus can be applied on R and B without mixing colored symbols from R and B. While a local proof does not exist, note that the following proof is a non-local derivation in the substitution superposition (resolution) calculus:

$$\frac{p(r, x) \quad \neg p(y, b)}{\bot}\ (\mathsf{BR})$$

□

Example 4.1 shows that local proofs do not necessarily exist for complete proof systems, such as superposition. It also shows that the reverse interpolant of a universally quantified formula is not always a universal interpolant, and the use of quantifier alternations in reverse interpolants is unavoidable. Example 4.1 thus poses the following two questions:

(Q1) Is there a (simple, natural) modification of the superposition calculus which is complete also for local proof generation?

(Q2) Is there a bound on the number quantifier alternations in reverse interpolants of formulas without quantifier alternations?

In what follows we address these questions and study interpolation in the superposition calculus. To this end, we prove some results showing that *there is no complete "simple" modification of the superposition calculus*. We start by addressing question (Q2) and show that there is no lower bound on the number of quantifier alternations in an interpolant of universal sentences.

Theorem 4.1. There is no lower bound on the number of quantifier alternations in interpolants of universal sentences. That is, for every positive integer n there exist universal sentences R, B such that $\{R, B\}$ is unsatisfiable and every reverse interpolant of R and B has at least n quantifier alternations.

Proof. For this proof we need a skolemization algorithm with the following property: given any sentence F it converts F into a universal sentence F', possibly using some fresh symbols with the following properties:

(a) $F' \vdash F$;

(b) for every sentence G not using the fresh symbols of F', the set $\{F, G\}$ is unsatisfiable if and only if $\{F', G\}$ is unsatisfiable.

To obtain F' one can use the standard skolemization algorithm: first, convert F to the negation normal form, then replace every subformula of F of the form $\exists x(G[x])$ by the formula $G[f(\bar{y})]$, where f is a fresh function symbol and $\bar{y}$ are all free variables of $G[x]$.

Let n be a positive integer. Take any grey sentence A with n quantifier alternations which is not equivalent to any sentence with $n-1$ quantifier alternations. Apply skolemization to A and color every fresh symbol of the skolemized formula in the red color. Denote the resulting formula by R. Likewise, apply skolemization to $\neg A$ and color every fresh symbol of the skolemized formula in the blue color. Denote the resulting formula by B. We know that R and B are universal sentences, $R \vdash A$ and $B \vdash \neg A$. It follows that $\{R, B\}$ is unsatisfiable, so R and B has a reverse interpolant. We claim that every reverse interpolant I of R and B has at least n quantifier alternations.

Take any such reverse interpolant I. We know that $R \vdash I$, so $\{R, \neg I\}$ is unsatisfiable. By the properties of the skolemization algorithm, $\{A, \neg I\}$ is also unsatisfiable, and so $\vdash A \rightarrow I$. In the same way we can prove that $\{\neg A, I\}$ is unsatisfiable, so $\vdash I \rightarrow A$. But then A is equivalent to I, so I must have at least n quantifier alternations. □

Example 4.2. Note that one can use formulas of the form $\forall x_1 \exists y_1 \forall x_1 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots)$ as A in the proof. In this case we have

$$\begin{array}{rcl} R & = & \forall x_1 \forall x_2 \ldots p(x_1, r_1(x_1), x_2, r_2(x_1, x_2), \ldots) \\ B & = & \forall y_1 \forall y_2 \ldots \neg p(b_1, y_1, b_2(y_1), y_2, \ldots) \\ I & = & \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots) \end{array}$$

Interestingly, the resolution refutation consists of a single step deriving the empty clause from R and B. The formula I is a reverse interpolant of R and B with the smallest number of quantifier alternations among all such reverse interpolants. □

Evidently, one can easily modify the proof of Theorem 4.1 to work with ordinary interpolants instead of reverse ones, so we have the following result.

Theorem 4.2. There is no lower bound on the number of quantifier alternations in interpolants of formulas without quantifier alternations. That is, for every positive integer n there exist a universal sentence R and an existential sentence B such that $\vdash R \rightarrow B$ and every interpolant of R and B has at least n quantifier alternations.

Our original motivation for investigating interpolation for local and non-local derivations was similar to that of [17]: find a modification of the superposition calculus that is complete for local proofs. Theorem 4.1 shows that there is, in fact, no such modification, answering also question (Q1) above. Even more, we have the following result.

Theorem 4.3. Let S be an inference system with the following property: for every red formula R and blue formula B, if $\{R, B\}$ is unsatisfiable, then there is a local refutation of R, B in S. Then the number of quantifier alternations in refutations of universal formulas of S is not bound by any positive integer.

Proof. Take a positive integer n and formulas R, B satisfying the conditions of Theorem 4.1. Take any local refutation of R, B in S. Then the algorithm of [15] extracts from this refutation a reverse interpolant I of R and B such that I is a boolean combination of formulas occurring in this refutation. This implies that the refutation contains at least one formula with n or more quantifier alternations. □

Theorems 4.1 and 4.3 imply that the number of alternations in an interpolant is not bound by any integer and that local refutations in simple modifications of the superposition calculus cannot be used for interpolant extraction. These results answer our two questions above on interpolation.

Theorem 4.3 is a serious obstacle to creating complete inference systems producing local proofs: indeed, modern methods of first-order theorem proving rely upon inference systems dealing only with universal formulas, such as superposition.

Let us analyze what assumptions and properties of first-order predicate logic we used for proving the theorems above.

1. The proof of Theorem 4.1 used the existence of a skolemization procedure converting F to F'. This procedure can be used for an arbitrary theory T.

2. We used is *soundness* on the underlying inference system: in all inferences of our proof system the conclusion is a logical consequence of the premises. This property is required for extracting interpolants from local proofs.

Let T be a first-order theory. We say that T *has quantifier complexity* n, where $n \geq 0$, if n is the smallest number such that every formula of the language of T with $n + 1$ quantifier alternations is equivalent in T to a formula with n quantifier alternations. If there is no such number n, then we say that the quantifier complexity of T is ω. First-order predicate logic has quantifier complexity ω. Evidently, any complete theory T has quantifier complexity 0, since every formula is equivalent either to $\top$ or to $\bot$. The last property we used in Theorem 4.1 is

3. First-order predicate logic has quantifier complexity ω.

By inspecting the proof of Theorem 4.1 we can conclude the following:

Theorem 4.4. Let T be a theory with the quantifier complexity ω. There is no lower bound on the number of quantifier alternations in interpolants of universal formulas with respect to T in the following sense. For every positive integer n there exist universal formulas R, B, using symbols in T and fresh function symbols such that

1. $\{R, B\}$ is T-unsatisfiable;
2. there exists a reverse interpolant of R and B with respect to T;
3. every reverse interpolant of R and B with respect to T has at least n quantifier alternations. □

While quantifier complexity is an intrinsic property of a theory and therefore cannot be changed, we next consider variations of local proofs and interpolation where we weaken some of the notions used in this paper, namely soundness and the notion of interpolant.

5 Interpolants and Non-Local Proofs

In this section we use a notion of *relational interpolant* and show later that it can be used to extract interpolants from arbitrary (and not just local) resolution proofs for *logic without equality*. Relational interpolants use a weaker condition on colors, allowing mixing red and blue colors in function symbols but not in predicate symbols.

The notion of relational interpolant can already be found in [12] and was used in a number of publications, including [1] and [3]. In this section we will use an arbitrary first-order theory T, whenever possible.

A formula F is said to be *greyish* if all predicate symbols occurring in F are grey. Note that a greyish formula can contain red and blue function symbols.

Definition 5.1 (Relational Interpolant)**.** Let R be a red formula, B be a blue formula and $R \vdash_T B$. Then we call a *relational interpolant* of R and B (with respect to T) any formula I such that:

1. $R \vdash_T I$ and $I \vdash_T B$;
2. all free variable of I are also free variables in both R and B;
3. I is greyish.

We call a *relational reverse interpolant of* R *and* B (with respect to T) any relational interpolant of R and $\neg B$ (with respect to T).

Our next aim is to show how to extract reverse interpolants from arbitrary, possibly non-local superposition refutations in predicate logic with or without equality. To this end we will first show how one can extract relational reverse interpolants from such refutations. After that, in Section 6 we turn our attention to logic without equality and show that relational reverse interpolants can be turned into reverse interpolants. Interestingly, for relational interpolants there is a bound on the number of quantifier alternations. Our technique is based on a generalization of the notion of C-interpolant from [15].

Let C be a clause. By $C^{(r)}$ (respectively, $C^{(b)}$) we denote the multiset of all literals in C whose predicate symbols are red (respectively blue). By $C^{(g)}$ we denote the multiset of literals in C which belong to neither $C^{(r)}$ not $C^{(b)}$, that is all literals with a grey predicate symbol and equalities. Evidently, C is equal to $C^{(r)} \vee C^{(b)} \vee C^{(g)}$.

Definition 5.2 (C-interpolant)**.** Let R be a red sentence, B be a blue sentence and $R, B \vdash \bot$. Let C be a clause. A formula I is called a *C-interpolant* of R and B if it has the following properties.

Resolution (BR):

$$\frac{A \vee C \quad \neg A \vee D}{C \vee D}$$

Superposition (Sup):

$$\frac{s \simeq t \vee C \quad L[s]_p \vee D}{C \vee D \vee L[t]_p}$$

(ii) s' is not a variable,
(iii) $t \not\succ s$.

Factoring (F):

$$\frac{A \vee A \vee C}{A \vee C}$$

Equality Factoring (EF):

$$\frac{s \simeq t \vee s \simeq t' \vee C}{s \simeq t \vee t \not\simeq t' \vee C}$$

(ii) $t \not\succ s$,
(iii) $t' \not\succ s$,
(iv) $t' \not\succ t$.

Substitution (Subst):

$$\frac{C}{C\sigma}$$

Equality Resolution (ER):

$$\frac{s \not\simeq s \vee C}{C}$$

Figure 2: The Substitution Superposition Calculus

(C1) I is greyish.

(C2) $R, \neg I \vdash_T C^{(r)} \vee C^{(g)}$;

(C3) $B, I \vdash_T C^{(b)} \vee C^{(g)}$.

Note that the notion of relational reverse interpolant is a special case of the notion of C-interpolant. Indeed, take C to be $\bot$: then we have $R, \neg I \vdash_T \bot$ and $B, I \vdash_T \bot$.

The following obvious lemma demonstrates why dealing with relational interpolants and C-interpolants is easier than dealing with interpolants.

Lemma 5.1. Let I be a C-interpolant of sentences R and B and σ be an arbitrary substitution. Then $I\sigma$ is a $C\sigma$-interpolant of R and B. □

This lemma does not hold for standard interpolants, since applying a substitution to a grey formula I may introduce red and/or blue colors in I. However, the lemma holds for relational interpolants since applying a substitution to a formula cannot change the colors of predicate symbols in it.

In the sequel we will simply say "C-interpolant" instead of "C-interpolant of R and B" when R and B are clear from the context. We will now consider an alternative formulation of the superposition calculus, where substitution application is separated from other inferences. It is called the *substitution superposition calculus* and shown in Figure 2.

Evidently, this calculus is sound and any derivation in the superposition calculus can be changed into a derivation in the substitution superposition calculus by replacing applications of an mgu to inferences using (Subst).

Example 5.1. The non-local derivation of Example 4.1 can be changed into

$$\dfrac{\dfrac{p(r,x)}{p(r,b)}\ (\mathsf{Subst}) \qquad \dfrac{\neg p(y,b)}{\neg p(r,b)}\ (\mathsf{Subst})}{\bot}\ (\mathsf{BR})$$

Note that the formulas $p(r,b)$ and $\neg p(r,b)$ used in this refutation are greyish but mix red and blue colors, so the derivation is not local.

We are interested in resolution and superposition refutations from a set of clauses derived from R and B. We call a clause C an R-*clause* if C is implied by R and C contains no blue predicate symbols. Likewise, a clause C is said to be a B-*clause* if C is implied by B and C contains no red predicate symbols. Consider again the above non-local derivation. The formula $p(r,b)$ is an R-clause, whereas $\neg p(r,b)$ is a B-clause.

Based on Figure 2, the next lemma shows how to build C-interpolants from proofs in the substitution superposition calculus.

Lemma 5.2. Given a derivation of a clause C in the substitution superposition calculus from a set of R-clauses and B-clauses, one can build a C-interpolant I of R and B.

Proof. The proof is by induction on the number of inferences in the derivation, by case distinction on the inference rules.
Base case. Let us start with the base case where C is an R-clause. We show that $\bot$ is a required C-interpolant. The conditions $(C1)$ and $(C3)$ are obvious. The condition $(C2)$ becomes $R \vdash_T C^{(r)} \vee C^{(g)}$ and follows from the fact that C is an R-clause, so C is equal to $C^{(r)} \vee C^{(g)}$. Similarly, we conclude that $\top$ is a C-interpolant for every B-clause C.
Substitution rule. Consider the substitution rule

$$\frac{C}{C\sigma}\ .$$

and let I be a C-interpolant. Then by Lemma 5.1, $I\sigma$ is a $C\sigma$-interpolant.
Resolution rule. Consider a propositional resolution inference

$$\frac{A \vee C_1 \quad \neg A \vee C_2}{C_1 \vee C_2}\ .$$

and suppose that I_1 is a $(A \vee C_1)$-interpolant and I_2 is a $(\neg A \vee C_2)$-interpolant. It is easy to see that I_1 is a also a $(A \vee C_1 \vee C_2)$-interpolant and I_2 is a $(\neg A \vee C_1 \vee C_2)$-interpolant. Denote by C the clause $C_1 \vee C_2$. It follows that I_1 is a $(A \vee C)$-interpolant and I_2 is a $(\neg A \vee C)$-interpolant, and we should find a C-interpolant. Consider the three possible cases, depending on the color of the predicate symbol of A.
(a) A is greyish. In this case we claim that $(A \vee I_1) \wedge (\neg A \vee I_2)$ (that is, *if* A *then* I_2 *else* I_1) is a C-interpolant.
(b) A has a red predicate symbol. In this case we claim that $I_1 \vee I_2$ is a C-interpolant.
(c) A has a blue predicate symbol. In this case we claim that $I_1 \wedge I_2$ is a C-interpolant.
Superposition rule. Consider a propositional superposition inference. Similar to the resolution rule, we can assume that it has the form

$$\frac{s \simeq t \vee C \quad L[s] \vee C}{L[t] \vee C}\ ,$$

where L is an arbitrary literal. Consider, again, the three possible cases.
(a) L is greyish. In this case we claim that $(s \simeq t \vee I_1) \wedge (s \not\simeq t \vee I_2)$ (that is, *if* $s \simeq t$ *then* I_2 *else* I_1) is a C-interpolant.
(b) L has a red predicate symbol. In this case we claim that $(s \not\simeq t \wedge I_1) \vee I_2$ is a C-interpolant.
(c) L has a blue predicate symbol. In this case we claim that $(s \simeq t \vee I_1) \wedge I_2$ is a C-interpolant.
Factoring rule, equality factoring rule, and equality resolution rule. For all of these rules the conclusion C' is a logical consequence of its premise C, so every C-interpolant is also a C'-interpolant. □

Example 5.2. Consider the non-local proof of Example 5.1. Each clause in this proof is either an R-clause or a B-clauses. Using the construction of Lemma 5.2, we obtain $p(r, b)$ as the relational reverse interpolant of R and B.

Lemma 5.2 yields an algorithm to build a C-interpolant from an arbitrary, possibly non-local derivation of C in the substitution superposition calculus. Thus, we can also build a relational reverse interpolant I of R and B from a refutation. In the next section we show that *this result in logic without equality* can be extended to building a reverse interpolant by applying so-called *greying* to I.

The result of extracting relational reverse interpolants from superposition refutations was also proved in [12] and [3]. Our proof is considerably shorter and simpler than either of these proofs.

6 Interpolation for Logic Without Equality Using Relational Interpolants

In this section we investigate extraction of interpolants from arbitrary resolution refutations in *logic without equality*. Using Lemma 5.2, we already know how to build relational interpolants. In what follows, we discuss the result of [1] showing how to turn relational interpolants into interpolants in logic without equality by quantifying away colored terms occurring in relational interpolants.

This problem is addressed also in [12] and [3], however in both papers for logic with equality. We will comment on the relation of our result to the results proved in these related papers after Theorem 6.1 below.

Lemma 6.1. Let I, R, B be sentences without equality and I be a reverse relational interpolant of R and B. Let t be a non-variable term occurring in I with the following properties: (i) the top-level function symbol of t is red (respectively blue); (ii) t is not a proper subterm of any term whose top-level symbol is either red or blue. Consider the formula I' obtained by replacing all occurrences of t by a fresh variable x. Then $\exists x I'$ (respectively $\forall x I'$) is also a reverse relational interpolant of R and B. □

We do not prove this lemma since it is a special case of Theorem 6.1 below.

Let I, R, B be sentences without equality and I be a reverse relational interpolant of R and B. Let us call *greying* the process of quantifying away colored subterms in I as in Lemma 6.1 in an arbitrary order until it is not more applicable. For example, by applying greying to the greyish formula $p(r(b), r(r(b)), b)$ we first obtain $\exists x(p(r(b), x, b))$, then $\exists y \exists x(p(y, x, b))$ and finally the formula $\forall z \exists y \exists x(p(y, x, z))$. It is not hard to argue that greying always terminates and that the final formula of greying is grey. Therefore, we obtain the following result.

Theorem 6.1. Let I, R, B be sentences without equality and I be a reverse relational interpolant of R and B. Let I' be obtained by applying greying to I. Then I' is a reverse interpolant of R and B. $\square$

A proof of this theorem can be found in the book by Baaz and Leitsch [1], the theorem is Lemma 8.2.2 in that book.

Paper [3] proves the result of Theorem 6.1 for logic with equality but only for the case when colored symbols are constants. That is, it is shown that reverse interpolants can be extracted from arbitrary superposition refutations provided that all colored symbols are constants. In this case, the correctness of greying is obvious (for example, because each greying step is, effectively, a skolemization steps, where colored symbols are considered as Skolem functions) and hence Theorem 6.1 holds also in the setting of [3]. We will comment on this result below in Theorem 6.2. Note however that our Theorem 6.1 addresses arbitrary resolution proofs in logic without equality, without restricting colored symbols to only constants; that is, we work with arbitrary colored function symbols.

On the other hand, [12] claims that Theorem 6.1 holds for arbitrary proofs in logic with equality without any restriction on the input sentences R and B and concludes that using it one can extract interpolants from superposition proofs for logic with equality. Let us show that this result of [12] for logic with equality is incorrect.

Example 6.1. Consider the formula $a \simeq b \wedge r(a) \not\simeq r(b)$. The formula is unsatisfiable. Since $r(a)$ is a term satisfying Lemma 6.1, the lemma claims that $\exists x(a \simeq b \wedge x \not\simeq r(b))$ is unsatisfiable too. However, this formula is trivially satisfiable (it even has a model of size 2). The problem here is that the red inequality on larger terms $r(a) \not\simeq r(b)$ implies an inequality on smaller terms, moreover not containing the red color. $\square$

We conclude this section with a note related to the case when only constants are colored.

Theorem 6.2. Let R, B be formulas (with or without equality) in a language in which all colored function symbols are constants (but which can contains arbitrary colored predicate symbols) and $R \wedge B$ is unsatisfiable. Then there exists a reverse interpolant of R and B having at most one quantifier alternation.

Proof. Extract a reverse relational interpolant I from a superposition refutation of $R \wedge B$ as in the proof of Lemma 5.2. Then apply greying to I so that all red constants are quantified away first, followed by all blue constants. The resulting reverse interpolant has the form $\forall x_1 \ldots \forall x_n \exists \forall y_1 \ldots \exists y_m I'$, where I' is quantifier-free. $\square$

Note that Example 4.1 shows that at least one quantifier alternation in this theorem is necessary, even if there is no equality and no colored predicate symbols.

7 New Directions

The main results of this paper on the unbounded quantifier complexity of interpolants and non-existence of local proofs with a bounded quantifier complexity seem to demonstrate that the approach to building interpolants using local proofs fails for first-order logic.

However, we have already seen that for logic without equality one has an alternative way of extracting interpolants from refutations by building a relational interpolant and then applying greying. The interesting open question is what can be done for logic with equality. Paper [18] actually gives such a construction by using several techniques, including translation of logic with equality into logic without equality. The disadvantage of this technique is the large size of the interpolant as compared to the size of the refutation, which probably makes it unrealistic for applications of interpolation in program analysis and verification. On the contrary, interpolants extracted from local proofs are linear in the size of proofs[1] if they are represented as dags and if the number of premises of inference rules is bounded [15].

Interpolants can have an arbitrary number of quantifier alternations, which seems to exclude building a complete simple modification of the superposition calculus. Dealing efficiently with formulas with arbitrary quantifier alternations in a first-order theorem prover seems to be out of the question.

However, our results on quantifier complexity of interpolants rely on the soundness of the inference system. What we conjecture here is that one can define a relatively simple interpolating modification of the superposition calculus by relaxing the soundness property. One possible approach would be to introduce definitions or skolemization on the fly. Such calculi are routinely used in modern theorem proving. For example, Vampire uses the AVATAR architecture [24], where one can introduce propositional names for clauses during proof-search, though AVATAR does not let one handle arbitrary quantifier alternations. The geometric resolution calculus of [8] actually allows one to skolemize while applying resolution. If we introduce skolem functions (or even just constants) on the fly and implicit definitions corresponding to these functions or constants, we can still have an inference system using only clauses, but these clauses will denote formulas with quantifier alternations. If the resulting inference system is a simple modification of the superposition calculus, it should also be possible to adapt modern theorem provers to give practically efficient proof procedures for such calculi.

Note that the result of skolemization is a formula that can be stronger than the premise, so we lose the soundness property for the inference system, but not the soundness property for refutations: any set of formulas having a refutation is still unsatisfiable. In order to implement the approach described above, one will have to modify the notion of local proofs and related results to deal with fresh symbols and a weaker soundness property.

8 Related Work

Most of the interpolation-based verification methods, for example [16, 5, 7, 22, 4], require explicit construction of local proofs in the combined quantifier-free theory of linear arithmetic with uninterpreted function symbols. While local proofs in such proof systems can always be found, the construction of local proofs is theory specific, and thus cannot be easily extended to other theories, especially those with quantifiers. Unlike these methods, we address full first-order interpolation with theories and do not restrict ourselves to theory-specific reasoning. We show that there is no simple modification of the superposition calculus for local proof generation in first-order predicate logic with equality, and discuss limitations of a complete local proof generation in logic with equality.

First-order interpolation using local proofs in the superposition calculus is also studied in [3]. The interpolant extraction algorithm of [3] for non-ground refutations is proved to be complete if the only colored function symbols are constants. A related result is given in the interpolation algorithm of [10]: it is shown that non-local proofs can always be changed into local ones if all colored symbols are uninterpreted constants. Our results from Section 3 show that one cannot do better than [3, 10], that is there exist no complete local proof system using any calculus that is sound and deals with clauses. The works [1, 3] also relational interpolants (called

[1]The formulation in [15] does not mention that the number of rules must be bounded, if we do not use this condition, then it can become quadratic, as pointed out by Ken McMillan (private communications).

weak interpolants). We believe that our algorithm for a construction of relational interpolants and related proofs are simpler than those of [1, 3]. We show that interpolants can always be computed from relational interpolants in logic without equality, which is not discussed in [3]. As pointed out in [3], our result also holds for logic with equality under the restriction that all colored symbols are constants.

The notion or relational interpolants was originally introduced in [12]. The work of [12] claims that relational interpolants can always be turned into interpolants for superposition proofs in first-order logic with equality by greying. Our Example 6.1 shows however that this result is incorrect.

9 Conclusions

We addressed interpolation in first-order logic with equality and proved a number of new results on the (non-)existence of complete interpolation algorithms based on local proofs. We also proved a new result that for every positive integer n, there exist formulas with no quantifier alternations all whose interpolants have at least n quantifier alternations.

Our results imply that there is no simple modification of the superposition calculus that is complete for local proof generation. We noted however that by weakening the notion of interpolants, interpolants can always be computed for logic without equality and, under additional constraints, also for logic with equality. We also conjecture that by weakening the soundness requirement for inference systems one can obtain an efficient interpolating theorem prover for first-order logic with equality using a modification of the superposition calculus.

Acknowledgments. We thank Stefan Hetzl and Matthias Baaz for pointing out relevant works on relational interpolants and discussing proof-theoretic approaches on constructing such interpolants. We also thank Bernhard Gleiss for stimulating discussions.

References

[1] M. Baaz and A. Leitsch. *Methods of Cut-Elimination*, volume 34 of *Trends in Logic*. Springer Verlag, 2011.

[2] 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.

[3] M. P. Bonacina and M. Johansson. On Interpolation in Automated Theorem Proving. *J. Autom. Reasoning*, 54(1):69–97, 2015.

[4] J. Christ and J. Hoenicke. Proof Tree Preserving Tree Interpolation. *J. Autom. Reasoning*, 57(1):67–95, 2016.

[5] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Interpolant Generation in Satisfiability Modulo Theories. In *TACAS*, volume 4963 of *LNCS*, pages 397–412, 2008.

[6] W. Craig. Three uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957.

[7] A. Gupta, C. Popeea, and A. Rybalchenko. Solving Recursion-Free Horn Clauses over LI+UIF. In *APLAS*, pages 188–203, 2011.

[8] Jia Meng Hans de Nivelle. Geometric Resolution: A Proof Procedure Based on Finite Model Search. In *IJCAR*, pages 303–317, 2006.

[9] K. Hoder, L. Kovács, and A. Voronkov. Interpolation and Symbol Elimination in Vampire. In *IJCAR*, volume 6173 of *LNCS*, pages 188–195, 2010.

[10] K. Hoder, L. Kovács, and A. Voronkov. Playing in the Grey Area of Proofs. In *POPL*, pages 259–272, 2012.

[11] A. Holzer, D. Schwartz-Narbonne, M. Tabaei Befrouei, G. Weissenbacher, and T. Wies. Error Invariants for Concurrent Traces. In *FM*, volume 9995 of *LNCS*, pages 370–387, 2016.

[12] G. Huang. Constructing Craig Interpolation Formulas. In *COCOON*, volume 959 of *LNCS*, pages 181–190, 1995.

[13] R. Jhala and K. L. McMillan. Interpolant-based Transition Relation Approximation. In *CAV*, pages 39–51, 2005.

[14] R. Jhala and K. L. McMillan. A Practical and Complete Approach to Predicate Refinement. In *TACAS*, pages 459–473, 2006.

[15] L. Kovács and A. Voronkov. Interpolation and Symbol Elimination. In *CADE*, volume 5663 of *LNCS*, pages 199–213, 2009.

[16] K. L. McMillan. An Interpolating Theorem Prover. *J. of TCS*, 345(1):101–121, 2005.

[17] K. L. McMillan. Quantified Invariant Generation Using an Interpolating Saturation Prover. In *TACAS*, pages 413–427, 2008.

[18] N. Motohashi. Equality and Lyndon's Interpolation Theorem. *Journal of Symbolic Logic*, 49(1):123–128, 1984.

[19] 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.

[20] A. Podelski, M. Schäf, and T. Wies. Classifying Bugs with Interpolants. In *TAP*, volume 9762 of *LNCS*, pages 151–168, 2016.

[21] J. A. Robinson. A Machine-Oriented Logic Based on the Resolution Principle. *J. ACM*, 12(1):23–41, 1965.

[22] N. Totla and T. Wies. Complete Instantiation-Based Interpolation. *J. Autom. Reasoning*, 57(1):37–65, 2016.

[23] Y. Vizel, A. Gurfinkel, and S. Malik. Fast Interpolating BMC. In *CAV*, volume 9206 of *LNCS*, pages 641–657, 2015.

[24] 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 Verlag, 2014.