Interpolation and Symbol Elimination*

Laura Kovács[1] and Andrei Voronkov[2]

[1] EPFL
[2] University of Manchester

Abstract. We prove several results related to local proofs, interpolation and superposition calculus and discuss their use in predicate abstraction and invariant generation. Our proofs and results suggest that symbol-eliminating inferences may be an interesting alternative to interpolation.

1 Introduction

The study of interpolation in connection to verification has been pioneered by McMillan in connection with model checking [16], and by McMillan [17] and Henzinger et.al. [7] in connection with predicate abstraction. A number of papers appeared later discussing generation of interpolants for various theories and their use in verification, for example invariant generation [25, 8, 10, 23, 22, 18, 2].

In this paper we discuss interpolation and its use in verification. We start with preliminaries in Section 2. In Section 3 we define so-called local derivations and prove a general form of a result announced in [8], namely that interpolants can be extracted from proofs of special form (so-called local proofs) in *arbitrary theories and inference systems* sound for these theories. We also show that interpolants extracted from such proofs are boolean combinations of conclusions of so-called *symbol-eliminating* inferences. By observing that a similar form of symbol elimination turned out to be useful for generating complex quantified invariants in [13] and that interpolants obtained from proofs seem to be better for predicate abstraction and invariant generation than those obtained by quantifier elimination, we conclude that symbol elimination can be a key concept for applications in verification.

Further, in Section 4 we consider interpolation for inference systems dealing with universal formulas. We point out that a result announced in [18] is incorrect by giving a counterexample. We also further study the superposition inference system and show that, when we use a certain family of orderings, all ground proofs are local. This gives us a way of extracting interpolants from ground superposition proofs. We extend this result to the LASCA calculus of [12], which gives us a new procedure for generating interpolants for the quantifier-free theory of uninterpreted functions and linear rational arithmetic. Finally, in Section 5 we investigate the use of interpolants in invariant generation.

* This research was partly done in the frame of the Transnational Access Programme at RISC, Johannes Kepler University Linz, supported by the European Commission Framework 6 Programme for Integrated Infrastructures Initiatives under the project SCIEnce (contract No 026133). The first author was supported by the Swiss NSF.

2 Preliminaries

We will 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 and function symbols by f, g, possibly with indices. 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 $C_1 \wedge \ldots \wedge C_1 \rightarrow C$ is a tautology. Note that $C_1, \ldots, C_n, C$ may contain free variables.

A *signature* is any finite set of symbols. The *signature of a formula* A is the set of all symbols occurring in this formula. For example, the signature of $f(x) \simeq a$ is $\{f, a\}$. The *language of a formula* A, denoted by $\mathcal{L}_A$, is the set of all formulas built from the symbols occurring in A, that is formulas whose signatures are subsets of the signature of A.

THEOREM 1 (CRAIG'S INTERPOLATION THEOREM [3]). Let A, B be closed formulas and let $A \vdash B$. Then there exists a closed formula $I \in \mathcal{L}_A \cap \mathcal{L}_B$ such that $A \vdash I$ and $I \vdash B$.

In other words, every symbol occurring in I also occurs in both A and B. Every formula I satisfying this theorem will be called an *interpolant* of A and B.

Let us emphasise that Craig's Interpolation Theorem 1 makes no restriction on the signatures of A and B. There is a stronger version of the interpolation property proved in [15] (see also [19]) and formulated below.

THEOREM 2 (LYNDON'S INTERPOLATION THEOREM). Let A, B be closed formulas and let $A \vdash B$. Then there exists a closed formula $I \in \mathcal{L}_A \cap \mathcal{L}_B$ such that $A \vdash I$ and $I \vdash B$. Moreover, every predicate symbol occurring positively (respectively, negatively) in I, occurs positively (respectively, negatively), in both A and B.

By inspecting proofs of the two mentioned interpolation theorems one can also conclude that in the case when A and B are ground, they also have a ground interpolant; we will use this property later.

We call a *theory* any set of closed formulas. If T is a theory, we write $C_1, \ldots, C_n \vdash_T C$ to denote that the formula $C_1 \wedge \ldots \wedge C_1 \rightarrow C$ holds in all models of T. In fact, our notion of theory corresponds to the notion of *axiomatisable theory* in logic. When we work with a theory T, we call symbols occurring in T *interpreted* while all other symbols *uninterpreted*.

Note that Craig's interpolation also holds for theories in the following sense.

THEOREM 3. Let A, B be formulas and let $A \vdash_T B$. Then there exists a formula I such that

1. $A \vdash_T I$ and $I \vdash B$;
2. every uninterpreted symbol of I occurs both in A and B;
3. every interpreted symbol of I occurs in B.

Likewise, there exists a formula I such that

1. $A \vdash I$ and $I \vdash_T B$;
2. every uninterpreted symbol of I occurs both in A and B;
3. every interpreted symbol of I occurs in A.

Proof. We start with proving the first part. By $A \vdash_T B$ and compactness there exists a finite number of formulas $T' \subseteq T$ such that $A, T' \vdash B$. Denote by C the conjunction of formulas in T', then we have $A \wedge C \vdash B$. By Craig's interpolation theorem 1 there exists a formula I whose symbols occur both in $A \wedge C$ and B such that $A \wedge C \vdash I$ and $I \vdash B$. Let us prove that I satisfies all conditions of the theorem. Note that $A \wedge C \vdash I$ implies $A \vdash_T I$, so the first condition is satisfied. Now take any uninterpreted symbol of I. Note that it cannot occur in C, since all symbols in C are interpreted, so it occurs in A, and so in both A and B. The condition on interpreted symbols is obvious since all symbols occurring in I also occur in B.

The second part is proved similarly, in this case take Craig's interpolant of A and $C \rightarrow B$. $\square$

The proof of Theorem 3 is similar to a proof in [10]. It is interesting that this formulation of interpolation is not symmetric with respect to A and B since it does not state $A \vdash_T I$ and $I \vdash_T B$: only one of the implications $A \rightarrow I$ and $I \rightarrow B$ should be a theorem of T while the other one is a tautology in first-order logic. Thus, theory reasoning is required only to show one of these implications.

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 relativise all definitions to T. To be precise, we call an *interpolant* of A and B any formula I with the properties $A \vdash_T I$ and $I \vdash_T B$.

If E is a set of expressions, for example, formulas, and constants $c_1, \ldots, c_n$ do not occur in E, then we say that $c_1, \ldots, c_n$ 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.

There is a series of papers on using interpolants in model checking and verification starting with [16]. Unfortunately, in some of these papers the notion of interpolant has been changed. Although the change seems to be minor, it affects Lyndon's interpolation property and also does not let one use interpolation for formulas with free variables. Namely [17, 18] call an interpolant of A and B any formula I such that $A \vdash I$ and $B \wedge I$ is unsatisfiable. To avoid any confusion between the two notions of interpolant we introduce the following notion. We call a *reverse interpolant* of A and B any formula I such that $A \vdash_T I$ and $I, B \vdash_T \perp$. It is not hard to argue that reverse interpolants for A and B are exactly interpolants of A and $\neg B$ and that, when B is closed, reverse interpolants are exactly interpolants in the sense of [17, 18].

3 Inference systems and local derivation

In this section we will recall some terminology related to inference systems. It is commonly used in the theory of resolution and superposition [1, 20]; we do not restrict ourselves to the superposition calculus.

DEFINITION 4. 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

$$\frac{A_1 \quad \dots \quad A_n}{A} \ .$$

The formulas $A_1, \dots, A_n$ 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* $A_1, \dots, A_m$ if the derivation is finite and every leaf in it is either an axiom or one of the formulas $A_1, \dots, A_m$. A formula A is said to be *derivable from* assumptions $A_1, \dots, A_m$ if there exists a derivation of A from $A_1, \dots, A_m$. A *refutation* is a derivation of $\perp$. $\square$

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 $S' \supseteq S$.

Let us now fix two sentences A and B. In the sequel we assume A and B to be fixed and give all definitions relative to A and B. Denote by $\mathcal{L}$ the intersection of the languages of A and B, that is, $\mathcal{L}_A \cap \mathcal{L}_B$. We call signature symbols occurring both in A and B *clean* and all other signature symbols *dirty*. For a formula C, we say that C is *clean* if $C \in \mathcal{L}$, otherwise we say that C is *dirty*. In other words, clean formulas contain only clean symbols and every dirty formula contains at least one dirty symbol.

DEFINITION 5 (AB-DERIVATION). Let us call an *AB-derivation* any derivation Π satisfying the following conditions.

(AB1) For every leaf C of Π one of following conditions holds:
1. $A \vdash_T \forall C$ and $C \in \mathcal{L}_A$ or
2. $B \vdash_T \forall C$ and $C \in \mathcal{L}_B$.

(AB2) For every inference

$$\frac{C_1 \quad \dots \quad C_n}{C}$$

of Π we have $\forall C_1, \dots, \forall C_n \vdash_T \forall C$.

We will refer to property (AB2) as *soundness*. □

We will be interested in finding reverse interpolants of A and B. The case $\mathcal{L}_A \subseteq \mathcal{L}_B$ is obvious, since in this case A is a reverse interpolant of A and B. Likewise, if $\mathcal{L}_B \subseteq \mathcal{L}_A$, then $\neg B$ is a reverse interpolant of A and B. For this reason, in the sequel we assume that $\mathcal{L}_A \not\subseteq \mathcal{L}_B$ and $\mathcal{L}_B \not\subseteq \mathcal{L}_A$, that is, *both A and B contain dirty symbols*.

We are especially interested in a special kind of derivation introduced in [8] and called *local* (or sometimes called *split-proofs*). The definition of a local derivation is relative to formulas A and B.

DEFINITION 6 (LOCAL AB-DERIVATION). An inference

$$\frac{C_1 \quad \ldots \quad C_n}{C}$$

in an AB-derivation is called *local* if the following two conditions hold.

(L1) Either $\{C_1, \ldots, C_n, C\} \subseteq \mathcal{L}_A$ or $\{C_1, \ldots, C_n, C\} \subseteq \mathcal{L}_B$.
(L2) If all of the formulas $C_1, \ldots, C_n$ are clean, then C is clean, too.

A derivation is called *local* if so is every inference of this derivation. □

In other words, (L1) says that either all premises and the conclusion are in the language of A or all of them are in the language of B. Condition (L2) is natural (inferences should not introduce irrelevant symbols) but it is absent in other work. This condition is essential for us since without it the proof of our key Lemma 10 does not go through.

Papers [8, 18] claim that from a local AB-refutation one can extract a reverse interpolant for A and B. Moreover, the proofs of these papers imply that the interpolant is universal when both A and B are universal and ground when both A and B are ground. However, the proofs of these properties use unsound arguments. First, the proof for the ground case from [8] uses an argument that for a sound inference

$$\frac{C_1 \quad \ldots \quad C_n}{C}$$

the set of formulas $C_1, \ldots, C_n, \neg C$ is propositionally unsatisfiable and then refers to a result on interpolation for propositional derivations from [16]. Unfortunately, this argument cannot be used: for example, the ground formula $a \not\simeq a$ is unsatisfiable in the theory of equality but not propositionally unsatisfiable. Second, the proof for the universal case in [18] refers to the ground case, but one cannot use this reduction since substituting terms for variables in non-ground local derivations may give a non-local derivation. Let us give an example showing that the result for the universal case is incorrect.

EXAMPLE 7. Let A be the formula $a \not\simeq b$ and B the formula $\forall x(x \simeq c)$. Then a, b, c are dirty symbols and there is no clean symbol. The following is a local refutation in the superposition calculus:

$$\cfrac{\cfrac{\cfrac{x \simeq c \quad y \simeq c}{x \simeq y} \quad a \not\simeq b}{y \not\simeq b}}{\bot}$$

One possible reverse interpolant of A and B is $\exists x \exists y(x \not\simeq y)$, however, this reverse interpolant is not universal. Let us show that there exist no universal reverse interpolant of A and B. Suppose, by contradiction, that such a reverse interpolant exists. Then it has the form $\forall x_1 \ldots \forall x_n I(x_1, \ldots, x_n)$, where $I(x_1, \ldots, x_n)$ is a quantifier-free formula, $x_1, \ldots, x_n$ are the only variables of $I(x_1, \ldots, x_n)$, and $I(x_1, \ldots, x_n)$ does not contain a, b. Take fresh constants $c_1, \ldots, c_n$, then we have $a \not\simeq b \vdash I(c_1, \ldots, c_n)$. By Craig's interpolation applied to ground formulas there exists a ground reverse interpolant J of $a \not\simeq b$ and $I(c_1, \ldots, c_n)$. By the conditions on the signature of J, J can contain no symbols. Therefore J is either equivalent to $\bot$ or equivalent to $\top$. The former is impossible since we do not have $a \not\simeq b \vdash \bot$, hence J is equivalent to $\top$. But then $I(c_1, \ldots, c_n)$ is a tautology. Since the constants c_i's are fresh, $\forall x_1 \ldots \forall x_n I(x_1, \ldots, x_n)$ is also a tautology. But we have $\forall x_1 \ldots \forall x_n I(x_1, \ldots, x_n), \forall x(x \simeq y) \vdash \bot$, so $\forall x(x \simeq y) \vdash \bot$ too. This contradicts the obvious observation that $\forall x(x \simeq y)$ has models. $\square$

Below we will prove a general result on extracting interpolants from local refutations from which the ground case will follow. Moreover, in Section 4 we note that in the ground case, if we use the superposition calculus and a certain family of orderings, all superposition proofs are local, so an arbitrary superposition prover can be used for finding interpolants for ground formulas.

The proofs of our results will also show the structure of interpolants extracted from refutations. Any such interpolant is a boolean combination of some key inferences of the refutation, called *symbol-eliminating* inferences. This suggests that in addition to studying interpolants one can study symbol elimination in proofs.

Consider any AB-derivation Π. Note that by the soundness condition (AB2) we can replace every formula C occurring in this derivation by its universal closure $\forall C$ and obtain an AB-derivation Π' where inferences are only done on closed formulas. We will call such derivations Π' *closed*.

We want to show how one can extract interpolants from local proofs and also investigate the structure of such interpolants. To this end, we will prove key Lemma 10 about local proofs and introduce a notion that will be used to characterise interpolants. Let Π be a local AB-derivation and C a formula occurring in Π. We say that C is *justified by A (respectively by B) in Π* if C is clean and one of the following conditions hold:

(J1) C is a leaf of Π and $A \vdash_T C$.
(J2) C is a conclusion of an inference in Π of the form

$$\frac{C_1 \quad \cdots \quad C_n}{C} \ ,$$

such that for some $k \in \{1, \ldots, n\}$ the formula C_k is dirty and $C_k \in \mathcal{L}_A$ (respectively, $C_k \in \mathcal{L}_B$).

Note that the fact that C is justified by A does not necessarily imply that $A \vdash_T C$. Yet, the derivation of any such formula C uses at least one formula derived from A (see the proof of the following lemma).

Let us introduce a key notion of symbol-eliminating inference. We call a *symbol-eliminating inference* any inference of the form described in (J1) or (J2). That is, a symbol-eliminating inference is an inference having a clean conclusion and either no premises at all, as in (J1), or at least one dirty premise, as in (J2). The name is due to the fact that the inference of (J2) has at least one dirty symbol occurring in premises and this symbol is "eliminated" in the conclusion. In the case of (J1) one can use the following explanation. Suppose, for example, that C in (J1) is justified by A. The we can consider C as derived from A by an inference

$$\frac{A}{C} \ ,$$

which also "eliminates" a dirty symbol occurring in A.

LEMMA 8. Let Π be a local AB-derivation. Further, let C be a clean formula such that C is justified by A in Π and C is not a leaf of Π. Take the largest sub-derivation Π' of Π with the following properties:

1. The last inference of Π' is an inference of C satisfying (J2).
2. All formulas in Π' are in the language $\mathcal{L}_A$.

Then for every leaf C' of Π' one of the following conditions hold:

1. C' is clean and justified by B.
2. $C' \in \mathcal{L}_A$ and $A \vdash_T C'$.

The same holds if we swap A and B in the conditions.

Proof. First, consider the case when C' is dirty. Since every formula in Π' is in the language $\mathcal{L}_A$, then C' is in the language $\mathcal{L}_A$ too, but not in $\mathcal{L}_B$. Consider two cases. If C' is also a leaf of Π, then by (L2) and (AB1) we have $A \vdash_T C'$ and we are done. If C' is not a leaf of Π, consider the inference of C' in Π. Since Π is local, all premises of this inference are in the language $\mathcal{L}_A$, so the inference must be in Π', which contradicts the assumptions that C' is a leaf of Π' and that Π' is the largest sub-derivation satisfying (1) and (2).

It remains to consider the case when C' is clean. If $A \vdash_T C'$, then (since $C' \in \mathcal{L}_A$) we are done. If $A \not\vdash C'$, then there exists an inference of C' in Π from some premises $C_1, \ldots, C_n$. If all these premises are in the language $\mathcal{L}_A$, then the inference itself belongs to Π', which contradicts the assumptions that C' is a leaf of Π' and that Π' is the largest sub-derivation satisfying (1) and (2). Therefore, at least one of the premises is dirty and belongs to $\mathcal{L}_B$, then C' is justified by B and we are done. □

Note that in the sub-derivation Π' of this lemma *every leaf is a conclusion of a symbol-eliminating inference*.

Our aim now is to show how one can extract an interpolant from a local derivation. To this end we will first generalise the notion of interpolant by relativising it to a quantifier-free formula C.

DEFINITION 9 (C-INTERPOLANT). Let C be a quantifier-free formula. A formula I is called a C-*interpolant* of A and B if it has the following properties.

(C1) Every free variable of I is also a variable of C.
(C2) I is clean;
(C3) $\neg C, A \vdash_T I$;
(C4) $I, B \vdash_T C$. □

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

LEMMA 10. Let Π be a local closed AB-derivation of a clean formula C. Then there exists a C-interpolant of A and B. Moreover, this C-interpolant is a boolean combination of conclusions of symbol-eliminating inferences of Π.

Proof. The proof is by induction on the number of inferences in Π. If Π consists of a single formula C, then by property (AB1) of AB-derivation we should consider the following cases: $A \vdash_T C$ (so C is justified by A) and $B \vdash_T C$ (so C is justified by B). Consider the first case. We claim that C is a C-interpolant. Indeed, (C1) and (C2) are obvious since I coincides with C. (C3) becomes $\neg C, A \vdash_T C$ and is implied by $A \vdash_T C$. Finally, (C4) becomes $C, B \vdash_T C$ and holds trivially.

For the second case we claim that $\neg C$ is a C-interpolant. Indeed, (C1) and (C2) are obvious as in the previous case. (C3) becomes $\neg C, A \vdash_T \neg C$ and is trivial. Finally, (C4) becomes $\neg C, B \vdash_T C$ and is implied by $B \vdash_T C$.

Now suppose that Π consists of more than one formula. Consider the last inference of the derivation

$$\frac{C_1 \quad \cdots \quad C_n}{C} .$$

Let S be the set of formulas $\{C_1, \ldots, C_n\}$. Let us consider the following three cases.

1. S contains a dirty formula and $S \subseteq \mathcal{L}_A$ (note that in this case C is justified by A);
2. S contains a dirty formula and $S \subseteq \mathcal{L}_B$ (in this case C is justified by B).
3. $S \subseteq \mathcal{L}$ (that is, all premises of the inference are clean);

By the property (L1) of local derivations, these three cases cover all possibilities. We will show how to build a C-interpolant in each of the cases.
CASE 1 *(C is justified by A.)* Consider the largest sub-derivation Π' of Π deriving C and satisfying Lemma 8. This sub-derivation has zero or more clean leaves $C_1, \ldots, C_n$ justified by B and one or more leaves in the language $\mathcal{L}_A$ implied by A. Without loss of generality we assume that there is exactly one leaf D of the latter kind (if there is more than one, we can take their conjunction as D). By the soundness property of derivations we have $C_1, \ldots, C_n, D \vdash_T C$, which implies $C_1, \ldots, C_n, A \vdash_T C$. By the induction hypothesis, for all $j = 1, \ldots, n$ one can build a C_j-interpolant I_j of A and B satisfying the conditions of the lemma. We claim that

$$I \stackrel{\text{def}}{=} (C_1 \vee I_1) \wedge \ldots \wedge (C_n \vee I_n) \wedge \neg(C_1 \wedge \ldots \wedge C_n)$$

is a C-interpolant of A and B. We have to prove $\neg C, A \vdash_T I$ and $I, B \vdash_T C$. Since each I_j is a C_j-interpolant of A and B, we have $A \vdash_T C_j \vee I_j$, for all $j = 1, \ldots, n$. In addition, we have $C_1, \ldots, C_n, A \vdash_T C$, and so $\neg C, A \vdash_T \neg(C_1 \wedge \ldots \wedge C_n)$. This proves $\neg C, A \vdash_T I$.

It remains to prove $I, B \vdash_T C$. We will prove a stronger property $I, B \vdash_T \bot$. By the induction hypothesis we have $I_j, B \vdash_T C_j$, for all $j = 1, \ldots, n$. But $I \vdash I_j \vee C_j$, hence $I, B \vdash_T C_j$ for all $j = 1, \ldots, n$. Finally, we have $I \vdash \neg(C_1 \wedge \ldots \wedge C_n)$, which yields $I, B \vdash_T \bot$.

Note that I in this proof is a boolean combination of $C_1, \ldots, C_n, I_1, \ldots, I_n$, which implies that it is a boolean combination of conclusions of symbol-eliminating inferences.

CASE 2 *(C is justified by B.)* Consider the largest sub-derivation Π' of Π deriving C and satisfying Lemma 8. This sub-derivation has zero or more clean leaves $C_1, \ldots, C_n$ justified by A and one or more leaves in the language $\mathcal{L}_B$ implied by B. As in the previous case, we assume that there is exactly one leaf D of the latter kind. By the induction hypothesis, for all $j = 1, \ldots, n$ one can build a C_j-interpolant I_j of A and B satisfying the conditions of the lemma. We claim that

$$I \overset{\text{def}}{=} (C_1 \vee I_1) \wedge \ldots \wedge (C_n \vee I_n)$$

is a C-interpolant of A and B. The proof is similar to case 1.

CASE 3 *(all premises are clean).* In this case one can build a C-interpolant as in the previous cases by replacing D by $\top$. $\square$

This lemma implies the following key result.

THEOREM 11. Let Π be a closed local AB-refutation. Then one can extract from Π in linear time a reverse interpolant I of A and B. This reverse interpolant is a boolean combination of conclusions of symbol-eliminating inferences of Π. $\square$

When we speak about "linear time" in this theorem we mean that we build a dag representation of the interpolant. As a corollary of this theorem we obtain the following one.

THEOREM 12. Let Π be a closed local AB-refutation. Then one can extract from Π in linear time a reverse interpolant I of A and B. This interpolant is ground if all formulas in Π are ground. $\square$

If all formulas in the derivation are universal, the reverse interpolant is a boolean combination of universal formulas but not necessarily a universal formula. Example 7 shows that this result cannot be improved, since there may be no universal reverse interpolant even when all formulas in the local derivation are universal.

It is interesting to consider the use of interpolants in verification in view of this theorem. Most papers on the use of interpolation outside of propositional logic do not use the interpolant per se, but use the set of atoms occurring in some interpolant extracted from a proof [8]. Theorem 11 says that this set of atoms is exactly the set of atoms occurring in the conclusions of symbol-eliminating inferences. There is also a strong evidence that symbol elimination is a key to finding loop invariants [13]. This poses an interesting problem of studying symbol elimination in various theories. More precisely, given formulas A and B, we are interested in formulas C such that C is a conclusion of a symbol-eliminating inference in a derivation from A and B.

4 Superposition and interpolation

In this section we investigate extraction of interpolants from superposition refutations. This is motivated by potential applications of such interpolants to verification. We consider only ground formulas and two kinds of superposition calculus: the standard one and its extension LASCA from [12].

We have already pointed out using Example 7 that the result on extracting interpolants from superposition proofs in [18] is incorrect. The flaw in the proofs of this paper is as follows. It cites (without proof) the following property of the superposition calculus: if a clause C is implied by a saturated set of clauses S, then C is implied by a subset of S consisting of clauses strictly smaller than C. This property does not hold even if we use the subset consisting of clauses smaller than or equal to C. For example, the set consisting of a single clause $f(a) \not\simeq f(b)$ is saturated and $a \not\simeq b$ follows from it but $a \not\simeq b$ is strictly smaller than $f(a) \not\simeq f(b)$ in all simplification orderings.

In the rest of this section, unless stated otherwise, we assume to deal with ground formulas only. A detailed description of the superposition calculus can be found in [20], see [20, 12] for more details. The calculus LASCA [12] for ground linear rational arithmetic and uninterpreted functions is given in Figure 1. It is a two-sorted theory and uses the symbol $=$ for equality on the sort of rationals and the symbol $\simeq$ for equality on the second sort. In all rules we have the condition $l \succ r$. The standard superposition calculus for ground formulas can be obtained from LASCA by

- removing all arithmetical rules;
- replacing equality modulo AC $=_{AC}$ by the syntactic equality.
- Replacing the $\perp$-elimination rule with the equality resolution rule

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

Let us call a simplification ordering $\succ$ on ground terms *separating* if each dirty ground term is greater in this ordering than any clean ground term. It is not hard to argue that such orderings exists. For example, both the Knuth-Bendix [11] and the LPO [9] families of orderings can be made into separating orderings using the following ideas. In the case of KBO one should use ordinal-based KBO of [14], make every dirty symbol of weight w have weight $w\omega$ and preserve the weights of clean symbols. Then the weight of every dirty ground term will be at least ω and the weight of every clean ground term will be finite. Likewise, for LPO we should make all dirty symbols have higher precedence than any clean symbol. Note that in practice KBO (with finite weights) and LPO and the only orderings used in theorem provers.[3]

[3] Vampire [21] uses KBO where predicates may have weights greater than ω.

Fig. 1. Linear Arithmetic Superposition Calculus (LASCA) for ground clauses

Ordered Paramodulation:

$$\frac{C \vee l \simeq r \quad L[l']_p \vee D}{C \vee D \vee L[r]_p} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (l \simeq r) \succ C.\end{array}$$

Equality Factoring:

$$\frac{C \vee t' \simeq s' \vee t \simeq s}{C \vee s \not\simeq s' \vee t \simeq s'} \qquad \begin{array}{l}\text{(i) } t =_{AC} t',\\ \text{(ii) } (t \simeq s) \succeq C \vee t' \simeq s'.\end{array}$$

Gaussian Elimination:

$$\frac{C \vee l = r \quad L[l']_p \vee D}{C \vee D \vee L[r]_p} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (l = r) \succ C.\end{array}$$

Theory Equality Factoring:

$$\frac{C \vee l' = r' \vee l = r}{C \vee r > r' \vee r' > r \vee l = r'} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (l = r) \succeq C \vee l' = r'.\end{array}$$

Fourier-Motzkin Elimination:

$$\frac{C \vee l > r \quad -l' > r' \vee D}{C \vee D \vee -r' > r} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (l > r) \succ C,\\ \text{(iii) there is no } l'' > r'' \in C \text{ such that } l'' =_{AC} l\\ \text{(iv) } (-l' > r') \succ D\\ \text{(v) there is no } -l'' > r'' \in D \text{ such that } l'' =_{AC} l.\end{array}$$

Inequality Factoring (InF1):

$$\frac{C \vee \pm l' > r' \vee \pm l > r}{C \vee r > r' \vee \pm l > r} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (\pm l > r) \succeq C \vee \pm l' > r'.\end{array}$$

Inequality Factoring (InF2):

$$\frac{C \vee \pm l' > r' \vee \pm l > r}{C \vee r' > r \vee \pm l > r'} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (\pm l > r) \succeq C \vee \pm l' > r'.\end{array}$$

$\perp$-Elimination:

$$\frac{C \vee \perp}{C} \qquad \text{(i) } C \text{ contains only } \top, \perp \text{ literals.}$$

THEOREM 13. If $\succ$ is separating, then every AB-derivation in LASCA is local.

Proof. We will prove by induction that every inference in an AB-derivation is local. Note that this implies that every clause in this inference has either only symbols in $\mathcal{L}_A$ or only symbols in $\mathcal{L}_B$.

We will only consider the ordered paramodulation rule:

Ordered Paramodulation:

$$\frac{C \vee l \simeq r \quad L[l']_p \vee D}{C \vee D \vee L[r]_p} \qquad \begin{array}{l}\text{(i) } l =_{AC} l',\\ \text{(ii) } (l \simeq r) \succ C.\end{array}$$

The proof for all other rules is similar.

By the induction hypothesis both premises have either only symbols in $\mathcal{L}_A$ or only symbols in $\mathcal{L}_B$. Note that if the left premise is clean, then the inference is local: indeed, all symbols occurring in the conclusion also occur in premises. Suppose that the left premise is dirty. Without loss of generality we assume that it belongs to $\mathcal{L}_A$. Note that the conditions $l \succ r$ and $(l \simeq r) \succ C$ guarantee that l is the greatest term in the left premise: this implies that l contains a dirty symbol. But the right premise contains a term l' that is AC-equal to l, so l' contains this symbol too. Hence, the right premise cannot contain dirty symbols occurring in B and so the inference is local. $\square$

Theorems 11 and 13 yield a new algorithm for generating interpolants in the combination of linear rational arithmetic and superposition calculus. Namely, one should search for proofs in LASCA using a separating ordering and then extract interpolants from them.

5 Interpolation and invariant generation

In this section we discuss the use of interpolants in invariant generation for proving loop properties. For proving an assertion for a program containing loops one needs to find loop invariants. Jhala and McMillan [8] propose the following technique for extracting predicates used in loop invariants. Suppose that $P(\bar{s})$ is a post-condition to be proved, where $\bar{s}$ is the set of state variables of the program; the program variables are also considered as constants in a first-order language. One can generate one or more loop unrollings and consider the corresponding set of paths, each path leading from an initial state to a state where $P(\bar{s})$ must hold. Take any such path π, which uses n transitions represented by quantifier-free formulas

$$T_1(\bar{s}, \bar{s}'), \ldots, T_n(\bar{s}, \bar{s'}).$$

Let $Q(\bar{s})$ be a formula representing the set of initial states. Then one can write the (ground) formula

$$Q(\bar{s}_0) \wedge T_1(\bar{s}_0, \bar{s}_1) \wedge \ldots \wedge T_n(\bar{s}_{n-1}, \bar{s}_n) \wedge \neg P(\bar{s}_n), \tag{1}$$

expressing that we can follow the path π from an initial state to a state satisfying the negation of $P(\bar{s}_n)$. If (1) is satisfiable, then the path gives a counterexample for the post-condition $P(\bar{s}_n)$. Otherwise (1) is unsatisfiable, so it should have some kind of refutation. For simplicity consider the case when $n = 1$, then (1) becomes

$$Q(\bar{s}_0) \wedge T_1(\bar{s}_0, \bar{s}_1) \wedge \neg P(\bar{s}_1).$$

This is a formula containing constants referring to two states. Note that the reverse interpolant of $Q(\bar{s}_0) \wedge T_1(\bar{s}_0, \bar{s}_1)$ and $\neg P(\bar{s}_1)$ is a state formula using only the constants $\bar{s}_1$. It is proposed to generate such a reverse interpolant from the proof and try to build an invariant state formula from the collection of atoms occurring in interpolants for all paths. It turns out that this approach works well in practice, however notice that what is used in the invariant is not the interpolant but only atoms occurring in it. We know that, for local proofs, these atoms are exactly those occurring in conclusions of symbol-eliminating inferences. Jhala and McMillan [8] define a more general notion of interpolant referring to a sequence of formulas; we can reformulate all results of this section for this more general notion too.

Let us make a few observations suggesting that symbol elimination is another interesting property in this context. Note that [8] defines signatures in such a way that all symbols apart from the state constants are clean. Therefore a reverse interpolant is always sought for a pair of formulas $A(\bar{s}_0, \bar{s}_1)$ and $B(\bar{s}_1)$, where $\bar{s}_0$ are the only dirty symbols.

THEOREM 14. The formula I defined as $\exists \bar{x} A(\bar{x}, \bar{s}_1)$ is a reverse interpolant of $A(\bar{s}_0, \bar{s}_1)$ and $B(\bar{s}_1)$. Moreover, it is the strongest interpolant of these formulas, that is, for every other interpolant I' we have $I \vdash_T I'$. $\square$

The strongest reverse interpolant I from this theorem is not a ground formula. However, if T has quantifier elimination, one can obtain an equivalent ground formula by quantifier elimination. A similar observation is made by Kapur et.al. [10]. This implies, for example, that the Fourier-Motzkin variable elimination procedure gives interpolants for the theory of linear rational arithmetic. One can also note that the interpolant I of Theorem 14 represents the image of the set of all states under the transition having A as its symbolic representation. The interesting point is that in practice the image turned out to be not a good formula for generating invariants, see e.g. [16], so it is only interpolants extracted from refutations that proved to be useful in verification and model checking.

Let us also point out that an interesting related notion has recently been introduced by Gulwani and Musuvathi [6]. Namely, they call a *cover* of $\exists \bar{x} A(\bar{x}, \bar{s}_1)$ the strongest ground formula C such that $\exists \bar{x} A(\bar{x}, \bar{s}_1) \vdash_T C$. In general covers do not necessarily exist. Evidently, in a theory with quantifier elimination every formula has a cover. Gulwani and Musuvathi [6] show that there are theories having no quantifier elimination property but having the cover property, that is every existential formula has a cover. It is not hard to argue that the notion of cover captures all ground interpolants:

THEOREM 15. Suppose that $\exists \bar{x} A(\bar{x}, \bar{s}_1)$ has a cover I. If $A(\bar{s}_0, \bar{s}_1)$ and $B(\bar{s}_1)$ have a ground interpolant I', then I is also a ground interpolant of these formulas and we have $I \vdash I'$. $\square$

6 Related Work

Most of the existing approaches for generating interpolants, for example Henzinger et.al. [7], McMillan [17], Jhala and McMillan [8] require explicit construction of resolution proofs in the combined theory of linear arithmetic with uninterpreted function symbols. Interpolants are then derived from these proofs. Their method of extraction is quite complex compared to ours, especially when equality reasoning is involved.

Cimatti, Griggio and Sebastiani [2] show how to extract interpolants in the combined theory of linear arithmetic with uninterpreted function symbols. The interpolants are extracted from proofs produced by an SMT solver.

Unlike the aforementioned works, Rybalchenko and Sofronie-Stokkermans [22] do not require a priori constructed proofs. Instead, they reduce the problem of generating interpolants in the combination of the theories of linear arithmetic with uninterpreted functions to solving constraints in these theories in the hierarchical style of [23]. An approach for constructing interpolants in combinations of theories over disjoint signatures is proposed by Yorsh and Musuvathi [25]. Note that [25, 22] do not present experiments of whether interpolants generated using their methods are useful for verification.

Kapur, Majumdar and Zarba [10] discuss connections of interpolation to quantifier elimination. Some of their results have been cited here.

7 Conclusion

Our results suggest that local proofs and symbol-eliminating inferences can be an interesting alternative to interpolation. Note that one can search for local proofs even for theories not having the interpolation property. For example, the theory of arrays does not have this property [10] but there exists a simple axiomatisation of arrays that can be used in a superposition prover [24, 4, 5]. This would give an (incomplete) procedure for generating interpolants or finding symbol-eliminating proofs in the combination of the theories of uninterpreted functions, linear rational arithmetic and arrays. The procedure adds an axiomatisation of arrays to the non-ground version of LASCA and searches for local proofs in the resulting theory. We believe it is an interesting direction for future research.

Acknowledgments. We thank Konstantin Korovin for stimulating discussions.

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. A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Interpolant Generation in Satisfiability Modulo Theories. In C. R. Ramakrishnan and J. Rehof, editors, *TACAS*, volume 4963 of *Lecture Notes in Computer Science*, pages 397–412. Springer, 2008.
3. 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.

4. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Deciding Extensions of the Theory of Arrays by Integrating Decision Procedures and Instantiation Strategies. In *JELIA*, pages 177–189, 2006.
5. S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decision Procedures for Extensions of the Theory of Arrays. *Ann. Math. Artif. Intell.*, 50(3-4):231–254, 2007.
6. S. Gulwani and M. Musuvathi. Cover Algorithms and Their Combination. In Sophia Drossopoulou, editor, *ESOP 2008*, volume 4960 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2008.
7. T. A. Henzinger, R. Jhala, R. Majumdar, and K. L. McMillan. Abstractions from Proofs. In *Proc. of POPL*, pages 232–244, 2004.
8. R. Jhala and K. L. McMillan. A Practical and Complete Approach to Predicate Refinement. In *Prc. of TACAS*, pages 459–473, 2006.
9. S. Kamin and J.-J. Lévy. Two Generalizations of the Recursive Path Ordering. Unpublished, January 1980.
10. D. Kapur, R. Majumdar, and C. G. Zarba. Interpolation for Data Structures. In M. Young and P. T. Devanbu, editors, *SIGSOFT FSE*, pages 105–116. ACM, 2006.
11. D. Knuth and P. Bendix. Simple Word Problems in Universal Algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.
12. K. Korovin and A. Voronkov. Integrating Linear Arithmetic into Superposition Calculus. In *Proc. of CSL*, volume 4646 of *LNCS*, pages 223–237, 2007.
13. L. Kovacs and A. Voronkov. Finding Loop Invariants for Programs over Arrays Using a Theorem Prover. In *Proc. of FASE*, 2009. To appear.
14. M. Ludwig and U. Waldmann. An Extension of the Knuth-Bendix Ordering with LPO-Like Properties. In N. Dershowitz and A. Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 348–362. Springer, 2007.
15. R.C. Lyndon. An Interpolation Theorem in the Predicate Calculus. *Pacific Journal of Mathematics*, 9:129–142, 1959.
16. K. L. McMillan. Interpolation and SAT-Based Model Checking. In *Proc. of CAV*, pages 1–13, 2003.
17. K. L. McMillan. An Interpolating Theorem Prover. In *Proc. of TACAS*, pages 16–30, 2004.
18. K. L. McMillan. Quantified Invariant Generation Using an Interpolating Saturation Prover. In *Proc. of TACAS*, volume 4963 of *LNCS*, pages 413–427, 2008.
19. N. Motohashi. Equality and Lyndon's Interpolation Theorem. *Journal of Symbolic Logic*, 49(1):123–128, 1984.
20. 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.
21. A. Riazanov and A. Voronkov. The Design and Implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.
22. A. Rybalchenko and V. Sofronie-Stokkermans. Constraint Solving for Interpolation. In *Proc. of VMCAI*, pages 346–362, 2007.
23. V. Sofronie-Stokkermans. Interpolation in Local Theory Extensions. In *Proc. of IJCAR*, pages 235–250, 2006.
24. A. Stump, C. W. Barrett, D. L. Dill, and J. Levitt. A Decision Procedure for an Extensional Theory of Arrays. In *LICS*, 2001.
25. G. Yorsh and M. Musuvathi. A Combination Method for Generating Interpolants. In *Proc. of CADE*, pages 353–368, 2005.