UNIVERSITA' DEGLI STUDI DI MILANO

Dipartimento di Scienze dell'Informazione

RAPPORTO INTERNO N° RI 334-10

Rewriting-based Quantifier-free Interpolation for a Theory of Arrays (extended version)

Roberto Bruttomesso, Silvio Ghilardi, Silvio Ranise

Rewriting-based Quantifier-free Interpolation for a Theory of Arrays

Roberto Bruttomesso[1] and Silvio Ghilardi[2] and Silvio Ranise[3]
[1]Università della Svizzera Italiana, Formal Verification Group, Lugano (Switzerland)
[2]Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano (Italy)
[3]FBK (Fondazione Bruno Kessler), Trento, (Italy)

April 18, 2011

Abstract

The use of interpolants in model checking is becoming an enabling technology to allow fast and robust verification of hardware and software. The application of encodings based on the theory of arrays, however, is limited by the impossibility of deriving quantifier-free interpolants in general. In this paper, we show that, with a minor extension to the theory of arrays, it is possible to obtain quantifier-free interpolants. We prove this by designing an interpolating procedure, based on solving equations between array updates. Rewriting techniques are used in the key steps of the solver and its proof of correctness. To the best of our knowledge, this is the first successful attempt of computing quantifier-free interpolants for a theory of arrays.

This Technical Report is the extended version of a paper published in the proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA '11).

1 Introduction

After the seminal work of McMillan (see, e.g., [20]), Craig's interpolation [9] has become an important technique in verification. For example, the importance of computing *quantifier-free* interpolants to over-approximate the set of reachable states for model checking has been observed. Unfortunately, Craig's interpolation theorem does not guarantee that it is always possible to compute quantifier-free interpolants. Even worse, for certain first-order theories, it is known that quantifiers must occur in interpolants of quantifier-free formulae [15]. As a consequence, a lot of effort has been put in designing efficient procedures for the computation of quantifier-free interpolants for first-order theories which are relevant for verification (e.g., uninterpreted functions and fragments of Presburger arithmetics). Despite these efforts, so far, only the negative result in [15] is available for the computation of interpolants in the theory of arrays with extensionality, axiomatized by the following three sentences: $\forall y, i, e. rd(wr(y, i, e), i) = e$, $\forall y, i, j, e. i \neq j \Rightarrow rd(wr(y, i, e), j) = rd(y, j)$, and

$$\forall x, y. x \neq y \Rightarrow (\exists i.\ rd(x, i) \neq rd(y, i)),$$

where rd and wr are the usual operations for reading and updating arrays, respectively. This theory is important for both hardware and software verification, and a procedure for computing quantifier-free interpolants "*would extend the utility of interpolant extraction as a tool in the verifier's toolkit*" [20]. Indeed, the endeavour of designing such a procedure would be bound to fail (according to [15]) if we restrict ourselves to the original theory. To circumvent the problem, we replace the third axiom above with its Skolemization, i.e.,

$$\forall x, y. x \neq y \Rightarrow rd(x, \mathtt{diff}(x, y)) \neq rd(y, \mathtt{diff}(x, y))),$$

so that the Skolem function `diff` is supposed to return an index at which the elements stored in two distinct arrays are different. This variant of the theory of arrays admits quantifier-free interpolants for quantifier-free formulae. The main contribution of the paper is to prove this by designing an **algorithm for the generation of quantifier-free interpolants from finite sets** (intended conjunctively) **of literals in the theory of arrays with `diff`**. The algorithm uses as a sub-module a satisfiability procedure for sets of literals of the theory, based on a sequence of syntactic transformations organized in several groups. The most important group of such transformations is a Knuth-Bendix completion procedure (see, e.g., [2]) extended to solve an equation $a = wr(b, i, e)$ for b when this is required by the ordering defined on terms. The goal of these transformations is to produce a "modularized" constraint for which it is trivial to establish satisfiability. To compute interpolants, the satisfiability procedure is invoked on two mutually unsatisfiable sets A and B of literals. While running, the two instances of the procedure exchange literals on the common signature of A and B (similarly to the Nelson and Oppen combination method, see, e.g., [21]) and perform some additional operations. At the end of the computation, the execution trace is examined and the desired interpolant is built by applying simple rules manipulating Boolean combinations of literals in the common signature of A and B.

The paper is organized as follows. In §2, we recall some background notions and introduce the notation. In §3, we give the notion of modularized constraint and state its key properties. In §4, we describe the satisfiability solver for the theory of arrays with `diff` and extend it to produce interpolants in §5. Finally, we discuss the related work and conclude in §6. All proofs can be found in the Appendix below.

2 Background and Preliminaries

We assume the usual syntactic (e.g., signature, variable, term, atom, literal, formula, and sentence) and semantic (e.g., structure, truth, satisfiability, and validity) notions of first-order logic. The equality symbol "$=$" is included in all signatures considered below. For clarity, we shall use "$\equiv$" in the meta-theory to express the syntactic identity between two symbols or two strings of symbols.

A *theory* T is a pair (Σ, Ax_T), where Σ is a signature and Ax_T is a set of Σ-sentences, called the axioms of T (we shall sometimes write directly T for Ax_T). The Σ-structures in which all sentences from Ax_T are true are the *models* of T. A Σ-formula ϕ is *T-satisfiable* if there exists a model $\mathcal{M}$ of T such that ϕ is true in $\mathcal{M}$ under a suitable assignment $\mathtt{a}$ to the free variables of ϕ (in symbols, $(\mathcal{M}, \mathtt{a}) \models \phi$); it is *$T$-valid* (in symbols, $T \vdash \varphi$) if its negation is T-unsatisfiable or, equivalently, iff φ is provable from the axioms of T in a complete calculus for first-order logic. A formula φ_1 *T-entails* a formula φ_2 if $\varphi_1 \to \varphi_2$ is *T-valid*; the notation used for such T-entailment is $A \vdash_T B$ or simply $A \vdash B$, if T is clear from the context. The *satisfiability modulo the theory T ($SMT(T)$) problem* amounts to establishing the T-satisfiability of quantifier-free Σ-formulae.

Let T be a theory in a signature Σ; a *T-constraint* (or, simply, a constraint) A is a set of ground literals in a signature Σ' obtained from Σ by adding a set of free constants. Taking conjunction, we can see a finite constraint A as a single formula; thus, when we say that a constraint A is *T-satisfiable* (or just "satisfiable" if T is clear from the context), we mean that the associated formula (also called A) is satisfiable in a Σ'-structure which is a model of T. We have two notions of equivalence between constraints, which are summarized in the next definition:

Definition 2.1. Let A and B be finite constraints (or, more generally, first order sentences) in an expanded signature. We say that A and B are *logically equivalent* (modulo T) iff $T \vdash A \leftrightarrow B$; on the other hand, we say that they are *$\exists$-equivalent* (modulo T) iff $T \vdash A^{\exists} \leftrightarrow B^{\exists}$, where $A^{\exists}$ (and similarly $B^{\exists}$) is the formula obtained from A by replacing free constants with variables and then existentially quantifying them out.

Logical equivalence means that the constraints have the same semantic content (modulo T); $\exists$-equivalence is also useful because we are mainly interested in T-satisfiability of constraints and it is trivial to see that $\exists$-equivalence implies equi-satisfiability (again, modulo T). As an example, if we take a constraint A, we replace all occurrences of a certain term t in it by a fresh constant a and add the equality $a = t$, called the *(explicit) definition (of t)*, the constraint A' we obtain in this way is $\exists$-equivalent to A. As another example, suppose that $A \vdash_T a = t$, that a does not occur in t, and that A' is obtained from A by replacing a

by t everywhere; then the following four constraints are $\exists$-equivalent

$$A, \quad A \cup \{a = t\}, \quad A' \cup \{a = t\}, \quad A'$$

(the first three are also pairwise logically equivalent). The above examples show how explicit definitions can be introduced and removed from constraints while preserving $\exists$-equivalence.

Theories of Arrays. In this paper, we consider a variant of a three-sorted theory of arrays defined as follows. The McCarthy *theory of arrays* $\mathcal{AX}$ [17] has three sorts $\mathtt{ARRAY}, \mathtt{ELEM}, \mathtt{INDEX}$ (called "array", "element", and "index" sort, respectively) and two function symbols rd and wr of appropriate arities; its axioms are:

$$\forall y, i, e. \qquad rd(wr(y, i, e), i) = e \tag{1}$$

$$\forall y, i, j, e. \qquad i \neq j \Rightarrow rd(wr(y, i, e), j) = rd(y, j). \tag{2}$$

The theory of *arrays with extensionality* $\mathcal{AX}_{\text{ext}}$ has the further axiom $\forall x, y. x \neq y \Rightarrow (\exists i.\ rd(x, i) \neq rd(y, i))$ (called the 'extensionality' axiom). To build the *theory of arrays with* $\mathtt{diff}$ $\mathcal{AX}_{\mathtt{diff}}$, we need a further function symbol $\mathtt{diff}$ in the signature and we replace the extensionality axiom by its Skolemization

$$\forall x, y. \qquad x \neq y \Rightarrow rd(x, \mathtt{diff}(x, y)) \neq rd(y, \mathtt{diff}(x, y)). \tag{3}$$

As it is evident from axiom (3), the new symbol $\mathtt{diff}$ is a binary function of sort $\mathtt{INDEX}$ taking two arguments of sort $\mathtt{ARRAY}$: its semantics is a function producing an index where the input arguments differ (it has an arbitrary value in case the input arguments are equal).

We introduce here some notational conventions which are specific for constraints in our theory $\mathcal{AX}_{\mathtt{diff}}$. We use $a, b, \ldots$ to denote free constants of sort $\mathtt{ARRAY}$, $i, j, \ldots$ for free constants of sort $\mathtt{INDEX}$, and $d, e, \ldots$ for free constants of sort $\mathtt{ELEM}$; $\alpha, \beta, \ldots$ stand for free constants of any sort. Below, we shall introduce non-ground rewriting rules involving (universally quantified) variables of sort $\mathtt{ARRAY}$: for these variables, we shall use the symbols $x, y, z, \ldots$. We make use of the following abbreviations.

- [Nested write terms] By $wr(a, I, E)$ we indicate a nested write on the array variable a, where indexes are represented by the free constants list $I \equiv i_1, \ldots, i_n$ and elements by the free constants list $E \equiv e_1, \ldots, e_n$; more precisely, $wr(a, I, E)$ abbreviates the term $wr(wr(\cdots wr(a, i_1, e_1) \cdots), i_n, e_n)$. Notice that, whenever the notation $wr(a, I, E)$ is used, the lists I and E must have the same length; for empty I, E, the term $wr(a, I, E)$ conventionally stands for a.

- [Multiple read literals] Let a be a constant of sort $\mathtt{ARRAY}$, $I \equiv i_1, \ldots, i_n$ and $E \equiv e_1, \ldots, e_n$ be lists of free constants of sort $\mathtt{INDEX}$ and $\mathtt{ELEM}$, respectively; $rd(a, I) = E$ abbreviates the formula $rd(a, i_1) = e_1 \wedge \cdots \wedge rd(a, i_n) = e_n$.

Refl	$wr(a,I,E) = a \leftrightarrow rd(a,I) = E$ *Proviso*: $\mathit{Distinct}(I)$
Symm	$(wr(a,I,E) = b \wedge rd(a,I) = D) \leftrightarrow (wr(b,I,D) = a \wedge rd(b,I) = E)$ *Proviso*: $\mathit{Distinct}(I)$
Trans	$(a = wr(b,I,E) \wedge b = wr(c,J,D)) \leftrightarrow (a = wr(c, J \cdot I, D \cdot E) \wedge b = wr(c,J,D))$
Confl	$b = wr(a, I \cdot J, E \cdot D) \wedge b = wr(a, I \cdot H, E' \cdot F) \leftrightarrow$ $\leftrightarrow (b = wr(a,I,E) \wedge E = E' \wedge rd(a,J) = D \wedge rd(a,H) = F)$ *Proviso*: $\mathit{Distinct}(I \cdot J \cdot H)$
Red	$(a = wr(b,I,E) \wedge rd(b,i_k) = e_k) \leftrightarrow (a = wr(b, I{-}k, E{-}k) \wedge rd(b,i_k) = e_k)$ *Proviso*: $\mathit{Distinct}(I)$

Legenda: a and b are constants of sort `ARRAY`; $I \equiv i_1, \dots, i_n$, $J \equiv j_1, \dots, j_m$ and $H \equiv h_1, \dots, h_l$ are lists of constants of sort `INDEX`; $E \equiv e_1, \dots, e_n$, $E' \equiv e'_1, \dots, e'_n$, $D \equiv d_1, \dots, d_m$, and $F \equiv f_1, \dots, f_l$ are lists of constants of sort `ELEM`.

Figure 1: Key properties of write terms

- [Multiple equalities] If $L \equiv \alpha_1, \dots, \alpha_n$ and $L' \equiv \alpha'_1, \dots, \alpha'_n$ are lists of constants of the same sort, by $L = L'$ we indicate the formula $\bigwedge_{i=1}^n \alpha_i = \alpha'_i$.

- [Multiple distinctions] If $L \equiv \alpha_1, \dots, \alpha_n$ is a list of constants of the same sort, by $\mathit{Distinct}(L)$ we abbreviate the formula $\bigwedge_{i \neq j} \alpha_i \neq \alpha_j$.

- [Juxtaposition and subtraction] If $L \equiv \alpha_1, \dots, \alpha_n$ and $L' \equiv \alpha'_1, \dots, \alpha'_m$ are lists of constants, by $L \cdot L'$ we indicate the list $\alpha_1, \dots, \alpha_n, \alpha'_1, \dots, \alpha'_m$; for $1 \leq k \leq n$, the list $L - k$ is the list $\alpha_1, \dots, \alpha_{k-1}, \alpha_{k+1}, \dots, \alpha_n$.

Some key properties of equalities involving write terms are stated in the following lemma (see also Figure 1).

Lemma 2.2 (Key properties of write terms)**.** *The formulae in Figure 1 are all $\mathcal{AX}_{\mathtt{diff}}$-valid under the assumption that their provisoes - if any - hold (when we say that a formula ϕ is $\mathcal{AX}_{\mathtt{diff}}$-valid under the proviso π, we just mean that $\pi \vdash_{\mathcal{AX}_{\mathtt{diff}}} \phi$).*

A (ground) *flat* literal is a literal of the form $a = wr(b,I,E), rd(a,i) = e, \mathtt{diff}(a,b) = i, \alpha = \beta, \alpha \neq \beta$. Notice that replacing a sub-term t with a fresh constant α in a constraint A and adding the corresponding defining equation $\alpha = t$ to A always produces an $\exists$-equivalent constraint; by repeatedly applying this method, one can show that every constraint is $\exists$-equivalent to a *flat* constraint, i.e., to one containing only flat literals. We split a flat constraint

A into two parts, the *index* part A_I and the *main* part A_M: A_I contains the literals of the form $i = j, i \neq j, \mathtt{diff}(a, b) = i$, whereas A_M contains the remaining literals, i.e., those of the form $a = wr(b, I, E), a \neq b, rd(a, i) = e, e = d, e \neq d$ (atoms $a = b$ are identified with literals $a = wr(b, \emptyset, \emptyset)$). We write $A =< A_I, A_M >$ to indicate the two parts of the constraint A.

3 Constraints combination

We shall need basic term rewriting system terminology and results: the reader is referred to [2] for the required background. In the main part of a constraint, positive literals will be treated as rewrite rules; to get a suitable orientation, we use a *lexicographic path ordering* with a total precedence $>$ such that $a > wr > rd > \mathtt{diff} > i > e$, for all a, i, e of the corresponding sorts. This choice orients equalities $a = wr(b, I, E)$ *from left to right* when $a > b$; equalities like $a = wr(b, I, E)$ for $a < b$ or $a \equiv b$ will be called *badly orientable* equalities. Our plan to derive a quantifier-free interpolation procedure for $\mathcal{AX}_{\mathtt{diff}}$ relies on the notion of "modularized constraint": after introducing such constraints, we show that their satisfiability can be easily recognized (Lemma 3.4) and that they can be combined in a modular way (Proposition 3.5).

Definition 3.1. A constraint $A =< A_I, A_M >$ is said to be *modularized* iff it is flat and the following conditions are satisfied (we let $\tilde{I}, \tilde{E}$ be the sets of free constants of sort `INDEX` and `ELEM` occurring in A):

(o) no positive index literal $i = j$ occurs in A_I;

(i) no negative array literal $a \neq b$ occurs in A_M;

(ii) A_M does not contain badly orientable equalities;

(iii) the rewriting system A_R given by the oriented positive literals of A_M joined with the rewriting rules

$$rd(wr(x, i, e), j) \to rd(x, j) \qquad \text{for } i, j \in \tilde{I},\ e \in \tilde{E},\ i \not\equiv j \tag{4}$$

$$rd(wr(x, i, e), i) \to e \qquad \text{for } i \in \tilde{I},\ e \in \tilde{E} \tag{5}$$

$$wr(wr(x, i, e), j, d) \to wr(wr(x, j, d), i, e) \qquad \text{for } i, j \in \tilde{I},\ e, d \in \tilde{E},\ i > j \tag{6}$$

$$wr(wr(x, i, e), i, d) \to wr(x, i, d). \qquad \text{for } i \in \tilde{I},\ e, d \in \tilde{E} \tag{7}$$

is confluent and ground irreducible;[1]

[1]The latter means that no rule can be used to reduce the left-hand or the right-hand side of another ground rule. Notice that ground rules from A_R are precisely the rules obtained by orienting an equality from A_M (rules (4)-(7) are not ground as they contain one *variable*, namely the array variable x).

(iv) if $a = wr(b, I, E) \in A_M$ and i, e are in the same position in the lists I, E, respectively, then $rd(b, i) \not\downarrow_{A_R} e$ (we use $\downarrow_{A_R}$ for joinability of terms);

(v) $\{\mathtt{diff}(a, b) = i, \mathtt{diff}(a', b') = i'\} \subseteq A_I$ and $a \downarrow_{A_R} a'$ and $b \downarrow_{A_R} b'$ imply $i \equiv i'$;

(vi) $\mathtt{diff}(a, b) = i \in A_I$ and $rd(a, i) \downarrow_{A_R} rd(b, i)$ imply $a \downarrow_{A_R} b$.

Remark 3.2. Condition (o) means that the index constants occurring in a modularized constraint are implicitly assumed to denote distinct objects. This is confirmed also by the proof of Lemma 3.4 below: from which, it is evident that the addition of all the negative literals $i \neq j$ (for $i, j \in \tilde{I}$ with $i \not\equiv j$) does not compromise the satisfiability of a modularized constraint, precisely because such negative literals are implicitly part of the constraint.

In Condition (i), negative array literals $a \neq b$ are not allowed because they can be replaced by suitable literals involving fresh constants and the `diff` operation (see axiom (3)).

Rules (4) and (5) mentioned in condition (iii) reduce read-over-writes and rules (6) and (7) sort indexes in flat terms $wr(a, I, E)$ in ascending order. In addition, condition (iv) prevents further redundancies in our rules.

Conditions (v) and (vi) deal with `diff`: in particular, (v) says that `diff` is "well defined" and (vi) is a "conditional" translation of the contraposition of axiom (3).

Remark 3.3. The non-ground rules from Definition 3.1(iii) form a convergent rewrite system (critical pairs are confluent): this can be checked manually (and can be confirmed also by tools like SPASS or MAUDE). Ground rules from A_R are of the form

$$a \to wr(b, I, E), \tag{8}$$

$$rd(a, i) \to e, \tag{9}$$

$$e \to d. \tag{10}$$

Only rules of the form (10) can overlap with the non-ground rules (4)-(7), but the resulting critical pairs are trivially confluent. Thus, in order to check confluence of A_M, *only overlaps between ground rules* (8)-(10) *need to be considered* (this is the main advantage of our choice to orient equalities $a = wr(b, I, E)$ from left to right instead of right to left).

Lemma 3.4. *A modularized constraint A is $\mathcal{AX}_{\mathtt{diff}}$-satisfiable iff for no negative element equality $e \neq d$ from A_M, we have that $e \downarrow_{A_R} d$.*

Let A, B be two constraints in the signatures Σ^A, Σ^B obtained from the signature Σ by adding some free constants and let $\Sigma^C := \Sigma^A \cap \Sigma^B$. Given a term, a literal or a formula φ we call it:

- *AB-common* iff it is defined over Σ^C;

- *A-local* (resp. *B-local*) if it is defined over Σ^A (resp. Σ^B);
- *A-strict* (resp. *B-strict*) iff it is A-local (resp. B-local) but not AB-common;
- *AB-mixed* if it contains symbols in both $(\Sigma^A \setminus \Sigma^C)$ and $(\Sigma^B \setminus \Sigma^C)$;
- *AB-pure* if it does not contain symbols in both $(\Sigma^A \setminus \Sigma^C)$ and $(\Sigma^B \setminus \Sigma^C)$.

(Notice that, sometimes in the literature about interpolation, "A-local" and "B-local" are used to denote what we call here "A-strict" and "B-strict"). The following modularity result is crucial for establishing interpolation in $\mathcal{AX}_{\mathtt{diff}}$:

Proposition 3.5. *Let $A = \langle A_I, A_M \rangle$ and $B = \langle B_I, B_M \rangle$ be constraints in expanded signatures Σ^A, Σ^B as above (here Σ is the signature of $\mathcal{AX}_{\mathtt{diff}}$); let A, B be both consistent and modularized. Then $A \cup B$ is consistent and modularized, in case all the following conditions hold:*

(O) *an AB-common literal belongs to A iff it belongs to B;*

(I) *every rewrite rule in $A_M \cup B_M$ whose left-hand side is AB-common has also an AB-common right-hand side;*

(II) *if a, b are both AB-common and $\mathtt{diff}(a, b) = i \in A_I \cup B_I$, then i is AB-common too;*

(III) *if a rewrite rule of the kind $a \to wr(c, I, E)$ is in $A_M \cup B_M$ and the term $wr(c, I, E)$ is AB-common, so is the constant a.*

4 A Solver for Arrays with `diff`

In this section we present a solver for the theory $\mathcal{AX}_{\mathtt{diff}}$. The idea underlying our algorithm is to separate the "index" part (to be treated by guessing) of a constraint from the "array" and "elem" parts (to be treated with rewriting techniques). The problem is how, given a *finite* constraint A, to determine whether it is satisfiable or not by transforming it into a modularized $\exists$-equivalent constraint.

4.1 Preprocessing

In order to establish the satisfiability of a constraint A, we first need a pre-processing phase, consisting of the following sequential steps:

$\boxed{\textbf{Step 1}}$ Flatten A, by replacing sub-terms with fresh constants and by adding the related defining equalities.

$\boxed{\textbf{Step 2}}$ Replace array inequalities $a \neq b$ by the following literals (i, e, d are fresh)

$$\mathtt{diff}(a, b) = i, \quad rd(b, i) = e, \quad rd(a, i) = d, \quad d \neq e.$$

$\boxed{\textbf{Step 3}}$ Guess a partition of index constants, i.e., for any pair of indexes i, j add either $i = j$ or $i \neq j$ (but not both of them); then remove the positive literals $i = j$ by replacing i by j everywhere (if $i > j$ according to the symbol precedence, otherwise replace j by i); if an inconsistent literal $i \neq i$ is produced, try with another guess (and if all guesses fail, report `unsat`).

$\boxed{\textbf{Step 4}}$ For all a, i such that $rd(a, i) = e$ does not occur in the constraint, add such a literal $rd(a, i) = e$ with fresh e.

At the end of the preprocessing phase, we get a finite set of flat constraints; *the disjunction of these constraints is $\exists$-equivalent to the original constraint.* For each of these constraints, go to the completion phase: *if the transformations below can be exhaustively applied (without failure) to at least one of the constraints, report* `sat`, *otherwise report* `unsat`.

The reason for inserting Step 4 above is just to simplify Orientation and Gaussian completion below. Notice that, even if rules $rd(a, i) \to e$ can be removed during completion, the following **invariant** is maintained: *terms $rd(a, i)$ always reduce to constants of sort* `ELEM`.

4.2 Completion

The completion phase consists in various transformations that should be non-deterministically executed until no rule or a failure instruction applies. For clarity, we divide the transformations into five groups.

(I) Orientation. This group contains a single instruction: get rid of badly orientable equalities, by using the equivalences **Refl**exivity and **Symm**etry of Figure 1; a badly orientable equality $a = wr(b, I, E)$ (with $a < b$) is replaced by an equality of the kind $b = wr(a, I, D)$ and by the equalities $rd(a, I) = E$ (all "read literals" required by the left-hand side of **Symm** comes from the above invariant). A badly orientable equality $a = wr(a, I, E)$ is removed and replaced by read literals only (or by nothing if I, E are empty).

(II) Gaussian completion. We now take care of the confluence of A_R (i.e., point (iii) of Definition 3.1). To this end, we consider all the critical pairs that may arise among our rewriting rules (8)-(10) (recall that, by Remark 3.3, there is no need to examine overlaps involving the non ground rules (4)-(7)). To treat the relevant critical pairs, we combine standard Knuth-Bendix completion for congruence closure with a specific method ("Gaussian completion") based on equivalences **Symm**etry, **Trans**itivity and **Confl**ict of Figure 1.[2] The critical pairs are listed below. Two preliminary observations are in order. First, we normalize a critical pair by using $\to_*$ before recovering convergence by adding a suitably oriented equality and removing the parent equalities (the symbol $\to_*$ denotes the reflexive and transitive closure of the rewrite relation $\to$ induced by the rewrite rules $A_R \cup \{(4)-(7)\}$). Second, the provisoes of all the equivalences in Figure 1 used below (i.e., **Symm**, **Trans**, and **Confl**) are satisfied because of the pre-processing Step 3 above.

(C1) $\boxed{wr(b_1, I_1, E_1) \;{}_*\!\leftarrow wr(b'_1, I'_1, E'_1) \leftarrow a \to wr(b'_2, I'_2, E'_2) \to_* wr(b_2, I_2, E_2)}$

with $b_1 > b_2$. We proceed in two steps. First, we use **Symm** (from right to left) to replace the parent rule $a \to wr(b'_1, I'_1, E'_1)$ with

$$wr(a, I_1, F) = b_1 \wedge rd(a, I_1) = E_1$$

for a suitable list F of constants of sort `ELEM` (notice that the equalities $rd(b_1, I_1) = F$, which are required to apply **Symm**, are already available because terms of the form $rd(b_1, j)$ for j in I_1 always reduce to constants of sort `ELEM` by the invariant resulting from the application of Step 4 in the pre-processing phase). Then, we apply **Trans** to the previously derived equality $b_1 = wr(a, I_1, F)$ and to the normalized second equality of the critical pair (i.e., $a = wr(b_2, I_2, E_2)$) and we derive

$$b_1 = wr(b_2, I_2 \cdot I_1, E_2 \cdot F) \wedge a = wr(b_2, I_2, E_2). \tag{11}$$

Hence, we are entitled to replace $b_1 = wr(a, I_1, F)$ with the rule $b_1 \to wr(b_2, J, D)$, where J and D are lists obtained by normalizing the right-hand-side of the first equality of (11) with respect to the non-ground rules (6) and (7). To summarize: the parent rules are removed and replaced by the rules

$$b_1 \to wr(b_2, J, D), \quad a \to wr(b_2, I_2, E_2)$$

and a bunch of new equalities of the form $rd(a, i) = e$, giving rise, in turn, to rules of the form $rd(b_2, i) \to e$ or to rewrite rules of the form (10) after normalization of their left members.

(C2) $\boxed{wr(b, I_1, E_1) \;{}_*\!\leftarrow wr(b'_1, I'_1, E'_1) \leftarrow a \to wr(b'_2, I'_2, E'_2) \to_* wr(b, I_2, E_2)}$

Since identities like $wr(c, H, G) = wr(c, \pi(H), \pi(G))$ are $\mathcal{AX}_{\mathtt{diff}}$-valid for every permutation π (under the proviso $Distinct(H)$), it is harmless to suppose that the set of index

[2]The name "Gaussian" is due to the analogy with Gaussian elimination in Linear Arithmetic (see [1,4] for a generalization to the first-order context).

variables $I := I_1 \cap I_2$ coincides with the common prefix of the lists I_1 and I_2; hence we have $I_1 \equiv I \cdot J$ and $I_2 \equiv I \cdot H$ for suitable disjoint lists J and H. Then, let E and E' be the prefixes of E_1 and E_2, respectively, of length equal to that of I; and let $E_1 \equiv E \cdot D$ and $E_2 \equiv E' \cdot F$ for suitable lists D and F. At this point, we can apply **Confl** to replace both parent rules forming the critical pair with

$$a = wr(b, I, E) \wedge E = E' \wedge rd(b, J) = D \wedge rd(b, H) = F,$$

where the first equality is oriented from left to right (i.e., $a \to wr(b, I, E)$).

(III) Knuth-Bendix completion. The remaining critical pairs are treated by standard completion methods for congruence closure.

(C3) $\boxed{d \;{}_*\!\!\leftarrow rd(wr(b, I, E), i) \leftarrow rd(a, i) \to e' \to_* e}$

Remove the parent rule $rd(a, i) \to e'$ and, depending on whether $d > e, e > d$, or $d \equiv e$, add the rule $d \to e$, $e \to d$, or do nothing. (Notice that terms of the form $rd(b, j)$ are always reducible because of the invariant of Step 4 in the pre-processing phase; hence, $rd(wr(b, I, E), i)$ always reduces to some constant of sort `ELEM`.)

(C4) $\boxed{e \;{}_*\!\!\leftarrow e' \leftarrow rd(a, i) \to d' \to_* d}$

Orient the critical pair (if e and d are not identical), add it as a new rule and remove one parent rule.

(C5) $\boxed{d \;{}_*\!\!\leftarrow d' \leftarrow e \to d_1' \to_* d_1}$

Orient the critical pair (if d and d_1 are not identical), add it as a new rule and remove one parent rule.

(IV) Reduction. The instructions in this group simplify the current rewrite rules.

(R1) If the right-hand side of a current ground rewrite rule can be reduced, reduce it as much as possible, remove the old rule, and replace it with the newly obtained reduced rule. Identical equations like $t = t$ are also removed.

(R2) For every rule $a \to wr(b, I, E) \in A_M$, exhaustively apply **Red**uction in Figure 1 from left to right (this amounts to do the following: if there are i, e in the same position k in the lists I, E such that $rd(b, i) \downarrow_{A_R} e$, replace $a = wr(b, I, E)$ with $a = wr(b, I–k, E–k)$).

(R3) If `diff`$(a, b) = i \in A_I$, $rd(a, i) \downarrow_{A_R} rd(b, i)$ and $a > b$, add the rule $a \to b$; replace also `diff`$(a, b) = i$ by `diff`$(b, b) = i$ (this is needed for termination, it prevents the rule for being indefinitely applied).

(V) Failure. The instructions in this group aim at detecting inconsistency.

(U1) If for some negative literal $e \neq d \in A_M$ we have $e \downarrow_{A_R} d$, report `failure` and backtrack to Step 3 of the pre-processing phase.

(U2) If $\{\mathtt{diff}(a, b) = i, \mathtt{diff}(a', b') = i'\} \subseteq A_I$ and $a \downarrow_{A_R} a'$ and $b \downarrow_{A_R} b'$ for $i \not\equiv i'$, report `failure` and backtrack to Step 3 of the pre-processing phase.

Notice that the instructions in the last two groups may require a confluence test $\alpha \downarrow_{A_R} \beta$ that can be effectively performed in case the instructions from groups (II)-(III) have been exhaustively applied, because then all critical pairs have been examined and the rewrite system A_R is confluent. If this is not the case, one may pragmatically compute and compare any normal form of α and β, keeping in mind that the test has to be repeated when all completion instructions (II)-(III) have been exhaustively applied.

Theorem 4.1. *The above procedure decides constraint satisfiability in $\mathcal{AX}_{\mathtt{diff}}$.*

5 The Interpolation Algorithm

In the literature one can roughly distinguish two approaches to the problem of computing interpolants. In the former (see e.g. [19, 3]), an interpolating calculus is obtained from a standard calculus by adding decorations so as to enable the recursive construction of an interpolating formula from a proof; in the latter (see, e.g., [23, 11, 7]), the focus is on how to extend an available decision procedure to return interpolants. Our methodology is similar to the second approach, since we add the capability of computing interpolants to the satisfiability procedure in Section 4. However, we do this by designing a flexible and abstract framework, relying on the identification of *basic operations* that can be performed independently from the method used by the underlying satisfiability procedure to derive a refutation.

5.1 Interpolating Metarules

Let now A, B be constraints in signatures Σ^A, Σ^B expanded with free constants and $\Sigma^C := \Sigma^A \cap \Sigma^B$; we shall refer to the definitions of AB-common, A-local, B-local, A-strict, B-strict, AB-mixed, AB-pure terms, literals and formulae given in Section 3. Our goal is to produce, in case $A \wedge B$ is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable, a ground AB-common sentence ϕ such that $A \vdash_{\mathcal{AX}_{\mathtt{diff}}} \phi$ and $\phi \wedge B$ is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable.

Let us examine some of the transformations to be applied to A, B. Suppose for instance that the literal ψ is AB-common and such that $A \vdash_{\mathcal{AX}_{\mathtt{diff}}} \psi$; then we can transform B into $B' := B \cup \{\psi\}$. Suppose now that we got an interpolant ϕ for the pair A, B': clearly, we can derive an interpolant for the original pair A, B by taking $\phi \wedge \psi$. The idea is to collect some useful transformations of this kind. Notice that these transformations can also modify the signatures Σ^A, Σ^B. For instance, suppose that t is an AB-common term and that c is a fresh constant: then we can put $A' := A \cup \{c = t\}$, $B' := B \cup \{c = t\}$: in fact, if ϕ is an interpolant for A', B', then $\phi(t/c)$ is an interpolant for A, B.[3] The transformations we need are called *metarules* and are listed in Table 1 below (in the Table and more generally in this Subsection, we use the notation $\phi \vdash \psi$ for $\phi \vdash_{\mathcal{AX}_{\mathtt{diff}}} \psi$).

An *interpolating metarules refutation* for A, B is a labeled tree having the following properties: (i) nodes are labeled by pairs of finite sets of constraints; (ii) the root is labeled by A, B; (iii) the leaves are labeled by a pair A, B such that $\perp \in A \cup B$; (iv) each non-leaf node is the conclusion of a rule from Table 1 and its successors are the premises of that rule. The crucial properties of the metarules are summarized in the following two Propositions.

Proposition 5.1. *The unary metarules* $\frac{A \mid B}{A' \mid B'}$ *from Table 1 have the property that* $A \wedge B$ *is* $\exists$*-equivalent to* $A' \wedge B'$*; similarly, the* n*-ary metarules* $\frac{A_1 \mid B_1 \quad \cdots \quad A_n \mid B_n}{A \mid B}$ *from Table 1 have the property that* $A \wedge B$ *is* $\exists$*-equivalent to* $\bigvee_{k=1}^{n}(A_k \wedge B_k)$.

Proposition 5.2. *If there exists an interpolating metarules refutation for* A, B *then there is a quantifier-free interpolant for* A, B *(namely there exists a quantifier-free* AB*-common sentence* ϕ *such that* $A \vdash \phi$ *and* $B \wedge \phi \vdash \perp$*). The interpolant* ϕ *is recursively computed applying the relevant interpolating instructions from Table 1.*

5.2 The Interpolation Solver

The metarules are complete, i.e. if $A \wedge B$ is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable, then (since we shall prove that an interpolant exists) a single application of (Propagate1) and (Close2) gives an interpolating metarules refutation. This observation shows that metarules are by no means better than the brute force enumeration of formulae to find interpolants. However, metarules are useful to design an algorithm manipulating pairs of constraints based on transformation instructions. In fact, each of the transformation instructions can be *justified* by a metarule (or by a sequence of metarules): in this way, if our instructions form a complete and terminating algorithm, we can use Proposition 5.2 to get the desired interpolants. The main advantage of using metarules as justifications is that we just need to take care of the completeness and termination of the algorithm, and not about interpolants anymore. Here "completeness" means that our transformations should be able to bring a pair (A, B) of constraints into a

[3]Notice that the fresh constant c is now a shared symbol, because Σ^A is enlarged to $\Sigma^A \cup \{c\}$, Σ^B is enlarged to $\Sigma^B \cup \{c\}$ and hence $(\Sigma^A \cup \{c\}) \cap (\Sigma^B \cup \{c\}) = \Sigma^C \cup \{c\}$.

Close1	Close2	Propagate1	Propagate2
$\dfrac{}{A \mid B}$	$\dfrac{}{A \mid B}$	$\dfrac{A \mid B \cup \{\psi\}}{A \mid B}$	$\dfrac{A \cup \{\psi\} \mid B}{A \mid B}$
Prv.: A is unsat.	*Prv.*: B is unsat.	*Prv.*: $A \vdash \psi$ and ψ is AB-common.	*Prv.*: $B \vdash \psi$ and ψ is AB-common.
Int.: $\phi' \equiv \bot$.	*Int.*: $\phi' \equiv \top$.	*Int.*: $\phi' \equiv \phi \wedge \psi$.	*Int.*: $\phi' \equiv \psi \to \phi$.

Define0	Define1	Define2
$\dfrac{A \cup \{a = t\} \mid B \cup \{a = t\}}{A \mid B}$	$\dfrac{A \cup \{a = t\} \mid B}{A \mid B}$	$\dfrac{A \mid B \cup \{a = t\}}{A \mid B}$
Prv.: t is AB-common, a fresh.	*Prv.*: t is A-local and a is fresh.	*Prv.*: t is B-local and a is fresh.
Int.: $\phi' \equiv \phi(t/a)$.	*Int.*: $\phi' \equiv \phi$.	*Int.*: $\phi' \equiv \phi$.

Disjunction1	Disjunction2
$\dfrac{\cdots \quad A \cup \{\psi_k\} \mid B \quad \cdots}{A \mid B}$	$\dfrac{\cdots \quad A \mid B \cup \{\psi_k\} \quad \cdots}{A \mid B}$
Prv.: $\bigvee_{k=1}^{n} \psi_k$ is A-local and $A \vdash \bigvee_{k=1}^{n} \psi_k$.	*Prv.*: $\bigvee_{k=1}^{n} \psi_k$ is B-local and $B \vdash \bigvee_{k=1}^{n} \psi_k$.
Int.: $\phi' \equiv \bigvee_{k=1}^{n} \phi_k$.	*Int.*: $\phi' \equiv \bigwedge_{k=1}^{n} \phi_k$.

Redplus1	Redplus2	Redminus1	Redminus2
$\dfrac{A \cup \{\psi\} \mid B}{A \mid B}$	$\dfrac{A \mid B \cup \{\psi\}}{A \mid B}$	$\dfrac{A \mid B}{A \cup \{\psi\} \mid B}$	$\dfrac{A \mid B}{A \mid B \cup \{\psi\}}$
Prv.: $A \vdash \psi$ and ψ is A-local.	*Prv.*: $B \vdash \psi$ and ψ is B-local.	*Prv.*: $A \vdash \psi$ and ψ is A-local.	*Prv.*: $B \vdash \psi$ and ψ is B-local.
Int.: $\phi' \equiv \phi$.	*Int.*: $\phi' \equiv \phi$.	*Int.*: $\phi' \equiv \phi$.	*Int.*: $\phi' \equiv \phi$.

ConstElim1	ConstElim2	ConstElim0
$\dfrac{A \mid B}{A \cup \{a = t\} \mid B}$	$\dfrac{A \mid B}{A \mid B \cup \{b = t\}}$	$\dfrac{A \mid B}{A \cup \{c = t\} \mid B \cup \{c = t\}}$
Prv.: a is A-strict and does not occur in A, t.	*Prv.*: b is B-strict and does not occur in B, t.	*Prv.*: c, t are AB-common, c does not occur in A, B, t.
Int.: $\phi' \equiv \phi$.	*Int.*: $\phi' \equiv \phi$.	*Int.*: $\phi' \equiv \phi$.

Table 1: Interpolating Metarules: each rule has a proviso *Prv.* and an instruction *Int.* for recursively computing the new interpolant ϕ' from the old one(s) $\phi, \phi_1, \ldots, \phi_k$.

pair (A', B') that either matches the requirements of Proposition 3.5 or is explicitly inconsistent, in the sense that $\bot \in A' \cup B'$. The latter is obviously the case whenever the original pair (A, B) is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable and it is precisely the case leading to an interpolating metarules refutation.

The basic idea is that of invoking the algorithm of Section 4 on A and B separately and to propagate equalities involving AB-common terms. We shall assume *an ordering precedence making AB-common constants smaller than A-strict or B-strict constants of the same sort.* However, this is not sufficient to prevent the algorithm of Section 4 from generating literals and rules violating one or more of the hypotheses of Proposition 3.5: this is why the extra correcting instructions of group (γ) below are needed. Our interpolating algorithm has a pre-processing and a completion phase, like the algorithm from Section 4.

Pre-processing. In this phase the four Steps of Section 4.1 are performed both on A and on B; to justify these steps we need metarules (Define0,1,2), (Redplus1,2), (Redminus1,2), (Disjunction1,2), (ConstElim0,1,2), and (Propagate1,2) - the latter because if i, j are AB-common, the guessing of $i = j$ versus $i \neq j$ in Step 3 can be done, say, in the A-component and then propagated to the B-component. At the end of the preprocessing phase, the following properties (to be maintained as invariants afterwards) hold:

(i1) A (resp. B) contains $i \neq j$ for all A-local (resp. B-local) constants i, j of sort `INDEX` occurring in A (resp. in B);

(i2) if a, i occur in A (resp. in B), then $rd(a, i)$ reduces to an A-local (resp. B-local) constant of sort `ELEM`.

Completion. Some groups of instructions to be executed non-deterministically constitute the completion phase. There is however an important difference here with respect to the completion phase of Section 4.2: it may happen that we need some *guessing* also inside the completion phase (only the instructions from group (γ) below may need such guessings). Each instruction can be easily justified by suitable metarules (we omit the details for lack of space). The groups of instructions are the following:

(α) Apply to A or to B any instruction from the completion phase of Section 4.2.

(β) If there is an AB-common literal that belongs to A but not to B (or vice versa), copy it in B (resp. in A).

(γ) Replace *undesired literals*, i.e., those violating conditions (I)-(II)-(III) from Proposition 3.5.

To avoid trivial infinite loops with the (β) instructions, rules in (α) deleting an AB-common literal should be performed *simultaneously* in the A- and in the B-components (it can be easily checked - see the Appendix below - that this is always possible, for instance if rules in (β) and (γ) are given higher priority). Instructions (γ) need to be described in more details. Preliminarily, we introduce a technique that we call *Term Sharing.* Suppose that the A-component contains a literal $\alpha = t$, where the term t is AB-common but the free constant α is only A-local. Then it is possible to "make α AB-common" in the following way. First, introduce a fresh AB-common constant α' with the explicit definition $\alpha' = t$ (to be inserted both in A and in B, as justified by metarule (Define0)); then replace the literal $\alpha = t$ by $\alpha = \alpha'$ and replace α by α' everywhere else in A; finally, delete $\alpha = \alpha'$ too. The result is a pair (A, B) where basically nothing has changed but α has been renamed to an AB-common constant α'. Notice that the above transformations can be justified by metarules (Define0), (Redplus1), (Redminus1), (ConstElim1). We are now ready to explain instructions (γ) in details. First, consider undesired literals corresponding to the rewrite rules of the form

$$rd(c, i) \to d \tag{12}$$

in which the left-hand side is AB-common and the right-hand side is, say, A-strict. If we apply Term Sharing, we can solve the problem by renaming d to an AB-common fresh constant d'. We can apply a similar procedure to the rewrite rules

$$a \to wr(c, I, E) \tag{13}$$

in case the right-hand side is AB-common and the left-hand side is not; when we rename a to some fresh AB-common constant c', we must arrange the precedence so that $c' > c$ to orient the renamed literal as $c' \to wr(c, I, E)$. Then, consider the literals of the form

$$\mathtt{diff}(a, b) = k \tag{14}$$

in which the left-hand side is AB-common and the right-hand side is, say, A-strict. Again, we can rename k to some AB-common constant k' by Term Sharing. Notice that k' is AB-common, whereas k was only A-local: this implies that we might need to perform some guessing to maintain the invariant (i1). Basically, we need to repeat Step 3 from Section 4.1 till invariant (i1) is restored (k' must be compared for equality with the other B-local constants of sort `INDEX`). The last undesired literals to take care of are the rules of the form[4]

$$c \to wr(c', I, E) \tag{15}$$

having an AB-common left-hand side but, say, only an A-local right-hand side. Notice that from the fact that c is AB-common, it follows (by our choice of the precedence) that c' is AB-common too. We can freely suppose that I and E are split into sublists I_1, I_2 and

[4]Literals $d = e$ are automatically oriented in the right way by our choice of the precedence.

E_1, E_2, respectively, such that $I \equiv I_1 \cdot I_2$ and $E \equiv E_1 \cdot E_2$, where I_1, E_1 are AB-common, $I_2 \equiv i_1, \dots, i_n$, $E_2 \equiv e_1, \dots, e_n$ and for each $k = 1, \dots, n$ at least one from i_k, e_k is not AB-common. This n (measuring essentially the number of non AB-common symbols in (15)) is called the *degree* of the undesired literal (15): in the following, we shall see how to eliminate (15) or to replace it with a smaller degree literal. We first make a guess (see metarule (Disjunction1)) about the truth value of the literal $c = wr(c', I_1, E_1)$. In the first case, we add the positive literal to the current constraint; as a consequence, we get that the literal (15) is equivalent to $c = wr(c, I_2, E_2)$ and also to $rd(c, I_2) = E_2$ (see **Red** in Figure 1). In conclusion, in this case, the literal (15) is replaced by the AB-common rewrite rule $c \to wr(c', I_1, E_1)$ and by the literals $rd(c, I_2) = E_2$. In the second case, we guess that the negative literal $c \neq wr(c', I_1, E_1)$ holds; we introduce a fresh AB-common constant c'' together with the defining AB-common literal[5]

$$c'' \to wr(c', I_1, E_1) \tag{16}$$

(see metarule (Define0)). The literal (15) is replaced by the literal

$$c \to wr(c'', I_2, E_2). \tag{17}$$

We show how to make the degree of (17) smaller than n. In addition, we eliminate the negative literal $c \neq c''$ coming from our guessing (notice that, according to (16), c'' renames $wr(c', I_1, E_1)$). This is done as follows: we introduce fresh AB-common constants i, d, d'' together with the AB-common defining literals

$$\mathtt{diff}(c, c'') = i, \quad rd(c, i) \to d, \quad rd(c'', i) \to d'' \tag{18}$$

(see metarule (Define0)). Now it is possible to replace $c \neq c''$ by the literal $d \neq d''$ (see axiom (3)). Under the assumption $Distinct(I_2)$, the following statement is $\mathcal{AX}_{\mathtt{diff}}$ valid:

$$c = wr(c'', I_2, E_2) \wedge rd(c'', i) = d'' \wedge rd(c, i) = d \wedge d \neq d'' \to \bigvee_{k=1}^{n} (i = i_k \wedge d = e_k).$$

Thus, we get n alternatives (see metarule (Disjunction1)). In the k-th alternative, we can remove the constants i_k, e_k from the constraint, by replacing them with the AB-common terms i, d respectively (see metarules (Redplus1), (Redplus2), (Redminus1), (Redminus2),(ConstElim1),(ConstElim
notice that it might be necessary to complete the index partition. In this way, the degree of (17) is now smaller than n.

In conclusion, if we apply exhaustively Pre-Processing and Completion instructions above, starting from an initial pair of constraints (A, B), we can produce a tree, whose nodes are

[5]We put $c > c'' > c'$ in the precedence. Notice that invariant (i2) is maintained, because all terms $rd(c'', h)$ normalize to an element constant. In case I_1 is empty, one can directly take c' as c''.

labelled by pairs of constraints (the successor nodes are labelled by pairs of constraints that are obtained by applying an instruction). We call such a tree an *interpolating tree* for (A, B). The following result shows that we obtained an interpolation algorithm for $\mathcal{AX}_{\mathtt{diff}}$:

Theorem 5.3. *Any interpolation tree for (A, B) is finite; moreover, it is an interpolating metarules refutation (from which an interpolant can be recursively computed according to Proposition 5.2) precisely iff $A \wedge B$ is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable.*

From the above Theorem it immediately follows that:

Theorem 5.4. *The theory $\mathcal{AX}_{\mathtt{diff}}$ admits quantifier-free interpolants (i.e., for every quantifier free formulae ϕ, ψ such that $\psi \wedge \phi$ is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable, there exists a quantifier free formula θ such that: (i) $\psi \vdash_{\mathcal{AX}_{\mathtt{diff}}} \theta$; (ii) $\theta \wedge \phi$ is not $\mathcal{AX}_{\mathtt{diff}}$-satisfiable: (iii) only variables occurring both in ψ and in ϕ occur in θ).*

In the Appendix, we also give a direct (although non-constructive) proof of this theorem by using the model-theoretic notion of amalgamation.

5.3 An Example

To illustrate our method, we describe the computation of an interpolant for the mutually unsatisfiable sets $A \equiv \{a = wr(b, i, d)\}, B \equiv \{rd(a, j) \neq rd(b, j), rd(a, k) \neq rd(b, k), j \neq k\}$. Notice that i, d are A-strict constants, j, k are B-strict constants, and a, b are AB-common constants with precedence $a > b$. We first apply Pre-Processing instructions to obtain $A \equiv \{a = wr(b, i, d), rd(a, i) = e_5, rd(b, i) = e_6\}, B \equiv \{rd(a, j) = e_1, rd(b, j) = e_2, rd(a, k) = e_3, rd(b, k) = e_4, e_1 \neq e_2, e_3 \neq e_4, j \neq k\}$. Since $a = wr(b, i, d)$ is an undesired literal of the kind (15), we generate the two subproblems $\Pi_1 \equiv (A \cup \{rd(b, i) = d, a = b\}, B)$ and $\Pi_2 \equiv (A \cup \{a \neq b\}, B)$.[6]

Let us consider Π_1 first. Notice that $A \vdash a = b$, and $a = b$ is AB-common. Therefore we send $a = b$ to B, and we may derive the new equality $e_1 = e_2$ from the critical pair (C3) $e_1 \leftarrow rd(a, j) \rightarrow rd(b, j) \rightarrow e_2$, thus obtaining $A \equiv \{a = b, rd(b, i) = d, rd(a, i) = e_5, rd(b, i) = e_6\}$, $B \equiv \{rd(b, j) = e_2, rd(a, k) = e_3, rd(b, k) = e_4, e_1 \neq e_2, e_3 \neq e_4, j \neq k, a = b, e_1 = e_2\}$. Now B is inconsistent. The interpolant for Π_1 can be computed with the *interpolating instructions* of the metarules (Close1,Propagate1,Redminus1,Redplus1) resulting in $\varphi_1 \equiv (\top \wedge a = b) \equiv a = b$.

Then, let us consider branch Π_2. Recall that this branch originates from the attempt of removing the undesired rule $a \rightarrow wr(b, i, d)$. We introduce the AB-common defining literals $\mathtt{diff}(a, b) = l, rd(a, l) = f_1, rd(b, l) = f_2$, and $f_1 \neq f_2$, in order to remove $a \neq b$ from

[6]Notice that this is precisely the case in which there is no need of an extra AB-common constant c''.

A. These are immediately propagated to B: $A \equiv \{a = wr(b, i, d), rd(a, i) = e_5, rd(b, i) = e_6, \texttt{diff}(a, b) = l, rd(a, l) = f_1, rd(b, l) = f_2, f_1 \neq f_2\}, B \equiv \{rd(a, j) = e_1, rd(b, j) = e_2, rd(a, k) = e_3, rd(b, k) = e_4, e_1 \neq e_2, e_3 \neq e_4, j \neq k, \texttt{diff}(a, b) = l, rd(a, l) = f_1, rd(b, l) = f_2, f_1 \neq f_2\}$. Since $a = wr(b, i, d)$ contains only the index i, we do not have a real case split. Therefore we replace i with l, and d with f_1. At last, we propagate the AB-common literal $a = wr(b, l, f_1)$ to B. After all these steps we obtain: $A \equiv \{a = wr(b, l, f_1), rd(a, l) = e_5, rd(b, i) = e_6, \texttt{diff}(a, b) = l, rd(a, l) = f_1, rd(b, l) = f_2, f_1 \neq f_2\}$, $B \equiv \{rd(a, j) = e_1, rd(b, j) = e_2, rd(a, k) = e_3, rd(b, k) = e_4, e_1 \neq e_2, e_3 \neq e_4, j \neq k, \texttt{diff}(a, b) = l, rd(a, l) = f_1, rd(b, l) = f_2, f_1 \neq f_2, a = wr(b, l, f_1)\}$. Since we have one more AB-common index constant l, we complete the current index constant partition, namely $\{k\}$ and $\{j\}$: we have three alternatives, to let l stay alone in a new class, or to add l to one of the two existing classes. In the first alternative, because of the following critical pair (C3) $e_1 \leftarrow rd(a, j) \rightarrow rd(wr(b, l, f_1), j) \rightarrow e_2$, we add $e_1 = e_2$ to B, which becomes trivially unsatisfiable. The other two alternatives yield similar outcomes. For each subproblem the interpolant, reconstructed by reverse application of the interpolating instructions of (Define0) and (Propagate1), is $\varphi'_2 \equiv \{(a = wr(b, \texttt{diff}(a, b), rd(a, \texttt{diff}(a, b))) \wedge rd(a, \texttt{diff}(a, b)) \neq rd(b, \texttt{diff}(a, b)))\}$. The interpolant φ_2 for the branch Π_2 has to be computed by combining with (Disjunction2) three copies of φ'_2, and so $\varphi_2 \equiv \varphi'_2$.

The final interpolant is computed by combining the interpolants for Π_1 and Π_2 by means of (Disjunction1), yielding $\varphi \equiv \varphi_1 \vee \varphi_2 \equiv (a = b \vee (a = wr(b, \texttt{diff}(a, b), rd(a, \texttt{diff}(a, b))) \wedge rd(a, \texttt{diff}(a, b)) \neq rd(b, \texttt{diff}(a, b))))$, i.e. $a = wr(b, \texttt{diff}(a, b), rd(a, \texttt{diff}(a, b)))$.

6 Related work and Conclusions

There is a series of papers devoted to building satisfiability procedures for the theory of arrays with or without extensionality. The interested reader is pointed to, e.g., [12,10] for an overview. In the following, for lack of space, we discuss the papers more closely related to interpolation for the theory of arrays.

After McMillan's seminal work on interpolation for model checking [18,20], several papers appeared whose aim was to design techniques for the efficient computation of interpolants in first-order theories of interest for verification, mainly uninterpreted function symbols, fragments of Linear Arithmetic, or their combination. An interpolating theorem prover is described in [19], where a sequent-like calculus is used to derive interpolants from proofs in propositional logic, equality with uninterpreted functions, linear rational arithmetic, and their combinations. In [15], a method to compute interpolants in data structures theories, such as sets and arrays (with extensionality), by axiom instantiation and interpolant computation in the theory of uninterpreted functions is described. It is also shown that the theory of arrays with extensionality does not admit quantifier-free interpolation. The "split" prover in [13] applies a sequent calculus for the synthesis of interpolants along the lines of that in [19] and is tuned for predicate abstraction [22]. The "split" prover can handle a combination of theories among which also the theory of arrays without extensionality is considered. In [13], it is pointed out that the theory of arrays poses serious problems in deriving quantifier-free interpolants because it entails an infinite set of quantifier-free formulae, which is indeed problematic when interpolants are to be used for predicate abstraction. To overcome the problem, [13] suggests to constrain array valued terms to occur in equalities of the form $a = wr(a, I, E)$ in the notation of this paper. It is observed that this corresponds to the way in which arrays are used in imperative programs. Further limitations are imposed on the symbols in the equalities in order to obtain a complete predicate abstraction procedure. In [14], the method described in [13] is specialized to apply CEGAR techniques [8] for the verification of properties of programs manipulating arrays. The method of [13] is extended to cope with range predicates which allow one to describe unbounded array segments which permit to formalize typical programming idioms of arrays, yielding property-sensitive abstractions. In [16], a method to derive quantified invariants for programs manipulating arrays and integer variables is described. A resolution-based prover is used to handle an *ad hoc* axiomatization of arrays by using predicates. Neither McCarthy's theory of arrays nor one of its extensions are considered in [16]. The invariant synthesis method is based on the computation of interpolants derived from the proofs of the resolution-based prover and constraint solving techniques to handle the arithmetic part of the problem. The resulting interpolants may contain even alternation of quantifiers.

To the best of our knowledge, the interpolation procedure presented in this paper is the first to compute quantifier-free interpolants for a natural variant of the theory of arrays with extensionality. In fact, the variant is obtained by replacing the extensionality axiom with its Skolemization which should be sufficient when the procedure is used to detect unsatisfiability of formulae as it is the case in standard model checking methods for infinite state systems. Because our method is not based on a proof calculus, we can avoid the burden of generating a large proof before being able to extract interpolants. The implementation of our procedure is currently being developed in the SMT-solver OpenSMT [5] and preliminary experiments are encouraging. An extensive experimental evaluation is planned for the immediate future.

Acknowledgements. We wish to thank an anonymuous referee for many useful criticisms that helped improving the quality of the paper.

References

[1] F. Baader, S. Ghilardi, and C. Tinelli. A new combination procedure for the word problem that generalizes fusion decidability results in modal logics. *Inform. and Comput.*, 204(10):1413–1452, 2006.

[2] F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, Cambridge, 1998.

[3] A. Brillout, D. Kroening, P. Rümmer, and W. Thomas. An Interpolating Sequent Calculus for Quantifier-Free Presburger Arithmetic . In *IJCAR*, 2010.

[4] R. Bruttomesso. *Problemi di combinazione nella dimostrazione automatica e nella verifica del software.* Università degli Studi di Milano, 2004. Master Thesis.

[5] R. Bruttomesso, E. Pek, N. Sharygina, and A. Tsitovich. The OpenSMT Solver. In *TACAS*, pages 150–153, 2010.

[6] C. Chang and J. H. Keisler. *Model Theory.* North-Holland, Amsterdam-London, third edition, 1990.

[7] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient Interpolation Generation in Satisfiability Modulo Theories. *ACM Trans. Comput. Logic*, 12:1–54, 2010.

[8] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-Guided Abstraction Refinement. In *CAV*, pages 154–169, 2000.

[9] W. Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *J. Symb. Log.*, pages 269–285, 1957.

[10] L. de Moura and N. Bjørner. Generalized, Efficient Array Decision Procedures. In *FMCAD*, pages 45–52, 2009.

[11] A. Fuchs, A. Goel, J. Grundy, S. Krstić, and C. Tinelli. Ground Interpolation for the Theory of Equality. In *TACAS*, pages 413–427, 2009.

[12] S. Ghilardi, E. Nicolini, S. Ranise, and D. Zucchelli. Decision procedures for extensions of the theory of arrays. *Annals of Mathematics and Artificial Intelligence*, 50:231–254, 2007.

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

[14] R. Jhala and K. L. McMillan. Array Abstractions from Proofs. In *CAV*, pages 193–206, 2007.

[15] D. Kapur, R. Majumdar, and C. Zarba. Interpolation for Data Structures. In *SIGSOFT'06/FSE-14*, pages 105–116, 2006.

[16] L. Kovács and A. Voronkov. Finding Loop Invariants for Programs over Arrays Using a Theorem Prover. In *FASE*, pages 470–485, 2009.

[17] J. McCarthy. Towards a Mathematical Science of Computation. In *IFIP Congress*, pages 21–28, 1962.

[18] K. L. McMillan. Interpolation and SAT-Based Model Checking. In *CAV*, pages 1–13, 2003.

[19] K. L. McMillan. An Interpolating Theorem Prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.

[20] K. L. McMillan. Applications of Craig Interpolation to Model Checking. In *TACAS*, pages 1–12, 2005.

[21] S. Ranise, C. Ringeissen, and D. Tran. Nelson-Oppen, Shostak and the Extended Canonizer: A Family Picture with a Newborn. In *ICTAC*, pages 372–386, 2004.

[22] H. Saidi and S. Graf. Construction of abstract state graphs with PVS. In *CAV*, pages 72–83, 1997.

[23] G. Yorsh and M. Musuvathi. A Combination Method for Generating Interpolants. In *CADE*, pages 353–368, 2005.

A Proofs of the main results

In this Section, we report the proofs of all our results. Meanwhile, we also make some further observations that might be useful, but could not be introduced in the body of the paper for space reasons.

A.1 Constraints

The statements of Lemma 2.2 are all immediate. We just sketch the proof of **Trans**itivity, as an example: one side is by replacement of equals; for the-right-to-left side, notice that the equalities $a = wr(c, J \cdot I, D \cdot E)$ and $b = wr(c, J, D)$ can be used as rewrite rules to rewrite both members of $a = wr(b, I, E)$ to the same term.

Remark A.1. The standard models of our theories $\mathcal{AX}_{\text{ext}}$ and $\mathcal{AX}_{\texttt{diff}}$ interpret arrays as functions, rd as function application and wr as the update operation (i.e. $wr(a, i, e)$ is the same as a except for index i where the new value to be returned is e). However, $\mathcal{AX}_{\text{ext}}$ and $\mathcal{AX}_{\texttt{diff}}$ are first-order theories and their models are just the Tarski structures where the axioms of $\mathcal{AX}_{\text{ext}}$ and $\mathcal{AX}_{\texttt{diff}}$, respectively, are true. Because of the extensionality axiom, it is easy to see that every model of such theories embeds into a standard one (see below for the definition of an embedding), which means in other words that every model is isomorphic to a model in which arrays are interpreted as functions (although it might happens that not all functions are part of the model - the situation is similar to the Henkin semantics for second order logic). As a consequence, whenever we want to test validity of universal formulae or satisfiability of constraints, we can limit ourselves to standard models: this is the case for instance of the statements of Lemma 2.2 (notice also that the proof of Lemma 3.4 below builds a standard model). On the other hand, in the proof of Theorem 5.4, we need to show that amalgamation holds for all models, not just for standard ones.

Lemma 3.4 *A modularized constraint A is $\mathcal{AX}_{\texttt{diff}}$-satisfiable iff for no negative index literal $e \neq d$ from A_M, we have that $e \downarrow_{A_R} d$.*

Proof. Clearly, the satisfiability of A implies that for no negative index literal $e \neq d$ from A_M, we have that $e \downarrow_{A_R} d$. Assume the antecedent of the converse: our aim is to build a model for A. We can freely make the following *further assumption*: if a, i occur in A and a is in normal form, there is some e such that $rd(a, i) = e$ belongs to A (in fact, if this does not hold, it is sufficient to add a further equality $rd(a, i) = e$ - with fresh e - without destroying the modularized property of the constraint).

Let I^* be the set of constants of sort `INDEX` occurring in A and let E^* be the set of constants of sort `ELEM` in normal form occurring in A (we have $I^* = \tilde{I}$ and $E^* \subseteq \tilde{E}$). Finally, we let X be the set of free constants of sort ARRAY occurring in A which are in normal form.

We build a model $\mathcal{M}$ as follows (the symbol $+$ denotes disjoint union):[7]

- $\texttt{INDEX}^{\mathcal{M}} := I^* + \{*\}$;
- $\texttt{ELEM}^{\mathcal{M}} := E^* + X$;
- $\texttt{ARRAY}^{\mathcal{M}}$ is the set of total functions from $\texttt{INDEX}^{\mathcal{M}}$ to $\texttt{ELEM}^{\mathcal{M}}$, $rd^{\mathcal{M}}$ and $wr^{\mathcal{M}}$ are the standard read and write operations (i.e. $rd^{\mathcal{M}}$ is function application and $wr^{\mathcal{M}}$ is the operation of modifying the first argument function by giving it the third argument as a value for the second argument input);
- for a constant i of sort INDEX, $i^{\mathcal{M}} := i$ for all $i \in I^*$;
- for a constant e of sort ELEM, $e^{\mathcal{M}}$ is the normal form of e;
- for a constant a of sort ARRAY in normal form and $i \in I^*$, we put $a^{\mathcal{M}}(i)$ to be equal to the normal form of $rd(a, i)$ (this is some $e \in \texttt{ELEM}^{\mathcal{M}}$ by our further assumption above); we also put $a^{\mathcal{M}}(*) := a$;[8]
- for a constant a of sort ARRAY not in normal form, let $wr(c, I, E)$ be the normal form of a: we let $a^{\mathcal{M}}$ to be equal to $wr^{\mathcal{M}}(c^{\mathcal{M}}, I^{\mathcal{M}}, E^{\mathcal{M}})$;[9]
- we shall define $\texttt{diff}^{\mathcal{M}}$ later on.

It is clear that in this way we have that all constants α of sort ELEM or ARRAY are interpreted in such a way that, if $\hat{\alpha}$ is the normal form of α, then

$$\alpha^{\mathcal{M}} = \hat{\alpha}^{\mathcal{M}}. \tag{19}$$

Also notice that, by the definition of $a^{\mathcal{M}}$, if e is the normal form of $rd(a, i)$, then we have

$$rd(a, i)^{\mathcal{M}} = e^{\mathcal{M}} \tag{20}$$

in any case (whether a is in normal form or not). Finally, if $wr(c, I, E)$ is the normal form of a, then

$$a^{\mathcal{M}} = c^{\mathcal{M}} \quad \Rightarrow \quad (I = \emptyset \text{ and } E = \emptyset); \tag{21}$$

[7]In a model $\mathcal{M}$, the interpretation of a sort symbol S (resp. function symbol f, predicate symbol P) will be denoted as $S^{\mathcal{M}}$ (resp. $f^{\mathcal{M}}$, $P^{\mathcal{M}}$). Similarly, if t is a ground term, $t^{\mathcal{M}}$ denotes the value of t in $\mathcal{M}$.

[8]Notice that $\texttt{ELEM}^{\mathcal{M}} := E^* + X$, hence $a \in \texttt{ELEM}^{\mathcal{M}}$.

[9]The definition is correct because a and c cannot coincide: since $a < wr(a, I, E)$, the term $wr(a, I, E)$ cannot be the normal form of a.

this is because the only rule that can reduce a must have a as left-hand side and $wr(c, I, E)$ as right-hand side (rules are ground irreducible), thus in the rule $a \to wr(c, I, E) \in A_M$ we must have $I = \emptyset, E = \emptyset$ in case $a^{\mathcal{M}} = c^{\mathcal{M}}$ (recall Definition 3.1(iv)).[10]

Since A is modularized, literals in A are flat. It is clear that all negative literals from A are true: in fact, a modularized constraint does not contain inequalities between array constants, inequalities between index constants are true by construction and inequalities between element constants are true by the hypothesis of the Lemma. Let us now consider positive literals in A: those from A_M are equalities of terms of sort `ELEM` or `ARRAY` and consequently are of the kind

$$e = d, \qquad a = wr(c, I, E), \quad rd(a, i) = e.$$

Since ground rules are irreducible, d is the normal form of e and $wr(c, I, E)$ is the normal form of a, hence we have $e^{\mathcal{M}} = d^{\mathcal{M}}$ and $a^{\mathcal{M}} = wr(c, I, E)^{\mathcal{M}}$ by (19) above. For the same reason a and e are in normal form in $rd(a, i) = e$, hence $rd(a, i)^{\mathcal{M}} = e^{\mathcal{M}}$ follows by construction.

It remains to define $\texttt{diff}^{\mathcal{M}}$ in such a way that flat literals $\texttt{diff}(a, b) = i$ from A_I are true and the axiom (3) is satisfied. Before doing that, let us observe that for all free constants a, b occurring in A, *we have that $a^{\mathcal{M}} = b^{\mathcal{M}}$ is equivalent to $a \downarrow_{A_R} b$*. In fact, one side is by (19); for the other side, suppose that $a^{\mathcal{M}} = b^{\mathcal{M}}$ and that $wr(c, I, E)$, $wr(c', I', E')$ are the normal forms of a and b, respectively. Then c must be equal to c', otherwise $a^{\mathcal{M}}$ and $b^{\mathcal{M}}$ would differ at index $*$. If either a or b is equal to c, trivially $a \downarrow_{A_R} b$ follows from (21) (one of the two is the normal form of the other). Otherwise, a and b are both reducible in A_R and since ground rules are irreducible and the only rules that can reduce an array constant have the left-hand side equal to that array constant, we have that $a \to wr(c, I, E)$ and $b \to wr(c, I', E')$ are both rules in A_R: as such, they are subject to Condition (iv) from Definition 3.1. First observe that we must have that $I \equiv I'$: otherwise, if there is $i \in I \setminus I'$, we could infer the following: (i) by (19), $b^{\mathcal{M}}(i) = c^{\mathcal{M}}(i)$; (ii) $c^{\mathcal{M}}(i)$ is the normal form of $rd(c, i)$ by construction; (iii) by $a^{\mathcal{M}} = b^{\mathcal{M}}$, $c^{\mathcal{M}}(i)$ is also equal to the normal form of the e having in the list E the same position as i in the list I, contrary to Condition (iv) from Definition 3.1. Since terms are normalized with respect to rule (6), I and I' coincide not only as sets, but also as lists; this means that the lists E and E' coincide too (the terms $wr(c, I, E)$, $wr(c, I, E')$ are in normal form and we have $wr(c, I, E)^{\mathcal{M}} = wr(c, I, E')^{\mathcal{M}}$).[11] Thus $a \downarrow_{A_R} b$ holds.

[10]In more detail: suppose that I and E are not empty and take $i \in I$ and $e \in E$ in corresponding positions. We have that $rd(c, i)^{\mathcal{M}} = rd^{\mathcal{M}}(c^{\mathcal{M}}, i^{\mathcal{M}}) = c^{\mathcal{M}}(i^{\mathcal{M}}) = a^{\mathcal{M}}(i^{\mathcal{M}}) = rd^{\mathcal{M}}(a^{\mathcal{M}}, i^{\mathcal{M}}) = rd(a, i)^{\mathcal{M}}$ (we used the definition of interpretation of a ground term, the fact that $rd^{\mathcal{M}}$ is interpretated as functional application and that $a^{\mathcal{M}} = c^{\mathcal{M}}$). Now, since $rd(a, i)$ normalizes to e, applying (20), we get that $rd(c, i)^{\mathcal{M}} = e^{\mathcal{M}}$, which means, again by (20), that $rd(c, i)$ normalizes to e too (e is in normal form, thus if $\tilde{e}$ is the normal form of $rd(c, i)$, we have that $\tilde{e}^{\mathcal{M}} = e^{\mathcal{M}}$ implies $e \equiv \tilde{e}$). This is contrary to Definition 3.1(iv).

[11]In more detail: let $i, e, \tilde{e}$ be in the k-th positions in the lists I, E, E', respectively. From $wr(c, I, E)^{\mathcal{M}} =$

Among the elements of $\mathtt{ARRAY}^{\mathcal{M}}$, some of them are of the kind $a^{\mathcal{M}}$ for some free constant a of sort ARRAY occurring in A and some are not of this kind: we call the former 'definable' arrays. In principle, it could be that $a^{\mathcal{M}} = b^{\mathcal{M}}$ for different a, b, but we have shown that this is possible only when a and b have the same normal form.

We are ready to define $\mathtt{diff}^{\mathcal{M}}$: we must assign a value $\mathtt{diff}^{\mathcal{M}}(\mathrm{a}, \mathrm{b})$ to all pairs of arrays $\mathrm{a}, \mathrm{b} \in \mathtt{ARRAY}^{\mathcal{M}}$. If a or b is not definable or if there are no a, b defining them such that $\mathtt{diff}(a, b)$ occurs in A_I, we can easily find $\mathtt{diff}^{\mathcal{M}}(\mathrm{a}, \mathrm{b})$ so that axiom (3) is true for a, b: one picks an index where they differ if they are not identical, otherwise the definition can be arbitrary. So let us concentrate into the case in which a, b are defined by constants a, b such that the literal $\mathtt{diff}(a, b) = i$ occurs in A_I: in this case, we define $\mathtt{diff}^{\mathcal{M}}(a^{\mathcal{M}}, b^{\mathcal{M}})$ to be i: Condition (v) from Definition 3.1 (together with the above observation that two constants defining the same array in $\mathcal{M}$ must have an identical normal form) ensures that the definition is correct and that all literals $\mathtt{diff}(a, b) = i \in A_I$ becomes true. Finally, axiom (3) is satisfied by Condition (vi) from Definition 3.1 and the fact that $rd(a, i)^{\mathcal{M}} = rd(b, i)^{\mathcal{M}}$ is equivalent to $rd(a, i) \downarrow_{A_R} rd(b, i)$ (to see the latter, just recall (20)). □

Remark A.2. As we said, the importance of Definition 3.1 lies in Lemma 3.4 and in Proposition 3.5 below. On the other hand, it is not true that if A is modularized, then A entails (modulo $\mathcal{AX}_{\mathtt{diff}}$) a positive literal $t = v$ iff $t \downarrow_{A_R} v$, even in case t, v are ground flat terms.[12] This may look unusual, however recall that our aim is not to decide equality by normalization but to have algorithms for satisfiability and interpolation.

Remark A.3. (This remark could be useful for combined problems.) The theory $\mathcal{AX}_{\mathtt{diff}}$ is stably infinite (in all its sorts) but non-convex: this means that it is suitable for Nelson-Oppen combination, but that disjunctions of equalities (not just equalities) need to be propagated from an $\mathcal{AX}_{\mathtt{diff}}$-constraint, in case it is involved in a combined problem. Actually, this does not happen for modularized constraints, because the proof of Lemma 3.4 actually shows the following stronger fact. Suppose that A is a modularized constraint satisfying the condition of the Lemma; then not only A is $\mathcal{AX}_{\mathtt{diff}}$-satisfiable but also $A \cup \{i \neq j\}_{i,j} \cup \{e \neq d\}_{e,d} \cup \{a \neq b\}_{a,b}$ is $\mathcal{AX}_{\mathtt{diff}}$-satisfiable, varying i, j among the pairs of different index constants occurring in A, e, d among the pairs of non-joinable element constants occurring in A, and a, b among the pairs of non-joinable array constants occurring in A. In other words, no disjunction of

$wr(c, I, E')^{\mathcal{M}}$, applying $rd^{\mathcal{M}}(-, i^{\mathcal{M}})$, we get $e^{\mathcal{M}} = \tilde{e}^{\mathcal{M}}$, i.e. $e \downarrow_{A_R} \tilde{e}$, which means $e \equiv \tilde{e}$ because $wr(c, I, E)$, $wr(c, I, E')$ are in normal form (in particular, their subterms $e, \tilde{e}$ are not reducible).

[12]As a counterexample consider $A = \{rd(a, i) \to e\}$; we have $A \vdash_{\mathcal{AX}_{\mathtt{diff}}} a = wr(a, i, e)$ but $a \not\downarrow_{A_R} wr(a, i, e)$. However, the proof of Lemma 3.4 shows that the following weaker but still important property holds: if A is modularized and t, v are terms of the same sort *occurring in* A, then $A \vdash_{\mathcal{AX}_{\mathtt{diff}}} t = v$ iff $t \downarrow_{A_R} v$.

equalities needs to be propagated and only equalities that can be syntactically extracted from A need to be propagated.

Next, we prove Proposition 3.5:

Proposition 3.5 *Let $A = \langle A_I, A_M\rangle$ and $B = \langle B_I, B_M\rangle$ be constraints in signatures Σ^A, Σ^B expanded with free constants (here Σ is the signature of $\mathcal{AX}_{\texttt{diff}}$); let A, B be both consistent and modularized. Suppose also that*

(O) *an AB-common literal belongs to A iff it belongs to B;*

(I) *every rewrite rule in $A_M \cup B_M$ whose left-hand side is AB-common has also an AB-common right-hand side;*

(II) *if a, b are both AB-common and $\texttt{diff}(a, b) = i \in A_I \cup B_I$, then i is AB-common too;*

(III) *if a rewrite rule of the kind $a \to wr(c, I, E)$ is in $A_M \cup B_M$ and the term $wr(c, I, E)$ is AB-common, so does the constant a.*

Then $A \cup B$ is consistent and modularized.

Proof. Since we cannot rewrite AB-common terms to terms which are not, it is easy to see that $A_M \cup B_M$ is still convergent and ground irreducible; the other conditions from Definition 3.1 are trivial, except condition (v). The latter is guaranteed by the hypotheses (II)-(III) as follows: the relevant case is when, say $\texttt{diff}(a, b) = i \in A_I$ is A-local and $\texttt{diff}(a', b') = i' \in B_I$ is B-local. If $a \downarrow a'$, since A_M and B_M are ground irreducible, we have that a single rewrite step reduces both a and a' to their normal form, that is we have

$$a \to wr(c, I, E) \leftarrow a'.$$

Now $wr(c, I, E)$ is AB-common, because the rules $a \to wr(c, I, E), a' \to wr(c, I, E)$ are in A_M and in B_M, respectively. By hypothesis (III), we have that a and a' are AB-common too; the same applies to b, b' and hence to i, i' by (II). Thus $\texttt{diff}(a', b') = i'$ is AB-common and belongs to A_I, hence $i \equiv i'$ because A is modularized.

Since all conditions from Definition 3.1 are satisfied, $A \cup B$ is modularized. Lemma 3.4 applies, thus yielding consistency. $\square$

Remark A.4. The above proof is so easy mainly because ground rewrite rules cannot superpose with the non ground rewrite rules (4)-(7) (with the exception of the rewrite rules $e \to d$, that may superpose but with trivially confluent critical pairs): this is the main benefit of our choice of orienting equalities $a = wr(b, I, E)$ from left-to-right (and not from right-to-left).

A.2 Amalgamation and Interpolation

In this subsection, we give a semantc proof of Theorem 5.4 based on amalgamation arguments (this subsection can be skipped by readers interested only in algorithmic aspects).

We summarize some basic model-theoretic notions that will be used in the sequel (for more details, the interested reader is pointed to standard textbooks in model theory, such as [6]).

If Σ is a signature, we use the notation $\mathcal{M} = (M, \mathcal{I})$ for a Σ-structure, meaning that M is the support[13] of $\mathcal{M}$ and $\mathcal{I}$ is the related interpretation function for Σ-symbols.

A Σ-*embedding* (or, simply, an embedding) between two Σ-structures $\mathcal{M} = (M, \mathcal{I})$ and $\mathcal{N} = (N, \mathcal{J})$ is any mapping $\mu : M \longrightarrow N$ among the corresponding support sets satisfying the following three conditions: (a) μ is a (sort-preserving) injective function; (b) μ is an algebraic homomorphism, that is for every n-ary function symbol f and for every $a_1, \dots, a_n \in M$, we have $f^{\mathcal{N}}(\mu(a_1), \dots, \mu(a_n)) = \mu(f^{\mathcal{M}}(a_1, \dots, a_n))$; (c) μ preserve and reflects interpreted predicates, i.e. for every n-ary predicate symbol P, we have $(a_1, \dots, a_n) \in P^{\mathcal{M}}$ iff $(\mu(a_1), \dots, \mu(a_n)) \in P^{\mathcal{N}}$.

If $M \subseteq N$ and the embedding $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ is just the identity inclusion $M \subseteq N$, we say that $\mathcal{M}$ is a *substructure* of $\mathcal{N}$ or that $\mathcal{N}$ is an *superstructure* of $\mathcal{M}$. Notice that a substructure of $\mathcal{N}$ is nothing but a subset of the carrier set of $\mathcal{N}$ which is closed under the Σ-operations and whose Σ-structure is inherited from $\mathcal{N}$ by restriction. In fact, given $\mathcal{N} = (N, \mathcal{J})$ and $G \subseteq N$, there exists the smallest substructure of $\mathcal{N}$ containing G in its carrier set. This is called the substructure *generated by* G and its carrier set can be characterized as the set of the elements $b \in N$ such that $t^{\mathcal{N}}(\underline{a}) = b$ for some Σ-term t and some finite tuple $\underline{a}$ from G (when we write $t^{\mathcal{N}}(\underline{a}) = b$, we mean that $(\mathcal{N}, \mathsf{a}) \models t(\underline{x}) = y$ for an assignment a mapping the $\underline{a}$ to the $\underline{x}$ and b to y). An easy—but fundamental—fact is that the truth of a universal (resp. existential) sentence is preserved through substructures (resp. through superstructures). A *universal* (resp. *existential*) sentence is obtained by prefixing a string of universal (resp. existential) quantifiers to a quantifier-free formula. A theory T is *universal* iff Ax_T consists of universal sentences.

Let $\mathcal{M} = (M, \mathcal{I})$ be a Σ-structure which is generated by $G \subseteq M$. Let us expand Σ with a set of fresh free constants in such a way that in the expanded signature Σ_G there is a fresh free constant c_g for every $g \in G$. Let $\mathcal{M}^G$ be the Σ_G-structure obtained from $\mathcal{M}$ by interpreting each c_g as g. The Σ_G-*diagram* $\delta_{\mathcal{M}}(G)$ *of* $\mathcal{M}$ is the set of all ground Σ_G-literals L such $\mathcal{M}^G \models L$. When we speak of the diagram of $\mathcal{M}$ *tout court*, we mean the Σ_M-diagram $\delta_{\mathcal{M}}(M)$.

[13]In the many-sorted case, the support is the disjoint union of the interpretations of the sorts symbols of Σ.

The following celebrated result [6] is simple, but nevertheless very powerful and it will be used in the rest of the paper.

Lemma A.5 (Robinson Diagram Lemma)**.** *Let $\mathcal{M} = (M, \mathcal{I})$ be a Σ-structure which is generated by $G \subseteq M$ and $\mathcal{N} = (N, \mathcal{J})$ be another Σ-structure. Then, there is a bijective correspondence between Σ-embeddings $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ and Σ_G-expansions $\mathcal{N}^G = (N, \mathcal{J}^G)$ of $\mathcal{N}$ such that $\mathcal{N}^G \models \delta_{\mathcal{M}}(G)$. The correspondence associates with μ the extension of $\mathcal{J}$ to Σ_G given by $\mathcal{J}^G(c_g) := \mu(g)$.*

Notice that an embedding $\mu : \mathcal{M} \longrightarrow \mathcal{N}$ is uniquely determined, in case it exists, by the image of the set of generators G: this is because the fact that G generates $\mathcal{M}$ implies (and is equivalent to) the fact that every $c \in M$ is of the kind $t^{\mathcal{M}}(\underline{g})$, for some term t and some $\underline{g}$ from G.

A theory T is said to have the *amalgamation property* iff whenever we are given embeddings

$$\mu_1 : \mathcal{N} \longrightarrow \mathcal{M}_1, \qquad \mu_2 : \mathcal{N} \longrightarrow \mathcal{M}_2$$

among models $\mathcal{N}, \mathcal{M}_1, \mathcal{M}_2$ of T, then there exists a further model $\mathcal{M}$ of T endowed with embeddings

$$\nu_1 : \mathcal{M}_1 \longrightarrow \mathcal{M}, \qquad \nu_2 : \mathcal{M}_2 \longrightarrow \mathcal{M}$$

such that $\nu_1 \circ \mu_1 = \nu_2 \circ \mu_2$. Notice that, up to isomorphism, we can limit ourselves in the above definition to the case in which μ_1, μ_2 are inclusions, i.e. to the case in which $\mathcal{N}$ is just a substructure of both $\mathcal{M}_1, \mathcal{M}_2$ (in that case, $\mathcal{M}$ is said to be a T-amalgam of $\mathcal{M}_1$ and $\mathcal{M}_2$ over $\mathcal{N}$).[14]

Let a, b be elements from $\mathtt{ARRAY}^{\mathcal{M}}$ in a model $\mathcal{M}$ of the theory $\mathcal{AX}_{\mathtt{diff}}$; we say that a, b *are cardinality dependent* (written $\mathcal{M} \models |a - b| < \omega$) iff $\{i \in \mathtt{INDEX}^{\mathcal{M}} \mid \mathcal{M} \models rd(a, i) \neq rd(b, i)\}$ is finite.

The meaning of the following Lemma is that cardinality dependence can be expressed as an infinite disjunction of quantifier-free formulae, hence it is preserved by sub- and superstructures.

Lemma A.6. *Let $\mathcal{N}$, $\mathcal{M}$ be models of $\mathcal{AX}_{\mathtt{diff}}$ such that $\mathcal{M}$ is a substructure of $\mathcal{N}$. For $a, b \in \mathtt{ARRAY}^{\mathcal{M}}$, it holds that*

$$\mathcal{M} \models |a - b| < \omega \quad \text{iff} \quad \mathcal{N} \models |a - b| < \omega.$$

[14]In case the signature does not have ground terms of some sort, models $\mathcal{N}$ having empty domain(s) must be included in the definition of amalgamation property.

Proof. [15] Write $\mathcal{M} \models |a - b| \leq n$ to say that $\{i \in \mathtt{INDEX}^{\mathcal{M}} \mid \mathcal{M} \models rd(a,i) \neq rd(b,i)\}$ has cardinality at most n. We show that the relations $|x - y| \leq n$ are quantifier-free definable (from this, the statement of the Lemma will be immediate). Otherwise said, we exhibit a quantifier-free formula $\chi_n(x,y)$ such that

$$\mathcal{M} \models |a - b| \leq n \quad \text{iff} \quad \mathcal{M} \models \chi_n(a,b)$$

holds for all $\mathcal{M}, a, b$. For $n = 0$, let $\chi_0(x,y)$ be $x = y$. Now, assume by the induction hypothesis that, for all a, b, $\mathcal{M} \models |a - b| \leq k$ holds iff $\mathcal{M} \models \chi_k(a,b)$ holds. Then, for $n = k + 1$, we have

$$\mathcal{M} \models |a - b| \leq k + 1 \quad \text{iff} \quad \mathcal{M} \models |wr(a, \mathtt{diff}(a,b), rd(b, \mathtt{diff}(a,b))) - b| \leq k$$

which gives

$$\mathcal{M} \models |a - b| \leq k + 1 \quad \text{iff} \quad \mathcal{M} \models \chi_k(wr(a, \mathtt{diff}(a,b), rd(b, \mathtt{diff}(a,b))), b)$$

by induction hypothesis. To conclude the proof, define $\chi(x,y)$ as the infinite disjunction of $\chi_0(x,y)$, $\chi_1(x,y)$, ..., $\chi_k(x,y)$, ... and recall that satisfiability of (infinite) disjunctions of quantifier-free formulae is preserved by taking both sub- and super-structures. Thus, since $\mathcal{M}$ is a substructure of $\mathcal{N}$ (by the assumption of the Lemma), we have that $\mathcal{M} \models \chi(a,b)$ iff $\mathcal{N} \models \chi(a,b)$. □

We are now ready to show that

Theorem A.7. *The theory $\mathcal{AX}_{\mathtt{diff}}$ has the amalgamation property.*

Proof. Let $\mathcal{N}$ be a model of $\mathcal{AX}_{\mathtt{diff}}$ which is a common substructure of two further models $\mathcal{M}_1, \mathcal{M}_2$ of $\mathcal{AX}_{\mathtt{diff}}$ (we can freely suppose that the set-theoretic differences $\mathtt{INDEX}^{\mathcal{M}_1} \setminus \mathtt{INDEX}^{\mathcal{N}}$ and $\mathtt{INDEX}^{\mathcal{M}_2} \setminus \mathtt{INDEX}^{\mathcal{N}}$ are disjoint, and similarly for $\mathtt{ARRAY}, \mathtt{ELEM}$). To show that $\mathcal{AX}_{\mathtt{diff}}$ has the amalgamation property, we must show that there exist a model $\mathcal{M}$ of $\mathcal{AX}_{\mathtt{diff}}$ and two embeddings from $\mathcal{M}_1$ and $\mathcal{M}_2$ to $\mathcal{M}$ that commute with the inclusions of $\mathcal{N}$ in $\mathcal{M}_1$ and $\mathcal{M}_2$. To this end, we use the (Robinson Diagram) Lemma A.5, which allows us to equivalently show that $L_1 \cup L_2$ is consistent, where L_1, L_2 are the Robinson diagrams of $\mathcal{M}_1, \mathcal{M}_2$, respectively.

[15] (Added April 2011) The following is a simpler proof of the Lemma (that, however, does not show definability of cardinality dependence). The right-to-left side is trivial because if $\mathcal{M} \models |a - b| < \omega$ then $\mathcal{M} \models a = wr(b, I, E)$ for some I, E and consequently also $\mathcal{N} \models a = wr(b, I, E)$ because $\mathcal{M}$ is a substructure of $\mathcal{N}$. Vice versa, suppose that $\mathcal{M} \not\models |a - b| < \omega$. This means that there are infinitely many $i \in \mathtt{INDEX}^{\mathcal{M}}$ such that $rd^{\mathcal{M}}(a,i) \neq rd^{\mathcal{M}}(b,i)$. Since $\mathcal{M}$ is a substructure of $\mathcal{N}$, there are also infinitely many $i \in \mathtt{INDEX}^{\mathcal{N}}$ such that $rd^{\mathcal{N}}(a,i) \neq rd^{\mathcal{N}}(b,i)$, i.e. $\mathcal{N} \not\models |a - b| < \omega$.

In turn, this is proved by transforming L_1 and L_2 into two (possibly infinite) modularized constraints satisfying the conditions of Proposition 3.5. The details of the proof follows.

We use as free constants the elements of the supports of $\mathcal{M}_1, \mathcal{M}_2$. Notice that the cardinality dependency relation is an equivalence relation and hence induces a partition on $\mathtt{ARRAY}^{\mathcal{M}}$ in each model $\mathcal{M}$ of $\mathcal{AX}_{\mathtt{diff}}$. We choose a representative element for each equivalence class both in $\mathtt{ARRAY}^{\mathcal{M}_1}$ and in $\mathtt{ARRAY}^{\mathcal{M}_2}$, in such a way that the representative is in $\mathtt{ARRAY}^{\mathcal{N}}$ whenever the equivalence class intersects the support of $\mathcal{N}$; also, if a class of $\mathtt{ARRAY}^{\mathcal{M}_1}$ and a class of $\mathtt{ARRAY}^{\mathcal{M}_2}$ intersect $\mathtt{ARRAY}^{\mathcal{N}}$, their representatives should be the same. We choose a total well-founded ordering on our constants giving lower precedence to constants coming from the support of $\mathcal{N}$ (also, the minimum in a cardinality dependence equivalence class of arrays should be the representative of that class): this total well-founded ordering is used for defining the signature precedence indicated in Section 4.[16] Now, we get a modularized constraint A from L_1 in the following way. First we can remove from L_1 non flat literals and get a logically equivalent constraint (this is a general fact since L_1 is a Robinson diagram). Next, we drop from L_1 all literals except: (α) element distinctions $e \neq d$; (β) index distinctions $i \neq j$; (γ) diff assertions $\mathtt{diff}(a, b) = i$; (δ) read assertions $rd(c, i) = e$, in case c is representative of its own equivalence class and I is sorted in strictly ascending order; (ϵ) write assertions $a = wr(c, I, E)$, in case c is representative of its own equivalence class. It is clear that A and L_1 are logically equivalent, because all flat literals from $L_1 \setminus A$ are either equalities whose members reduce to the same normal form modulo A_M or are array inequalities $a \neq b$ that are implied by equalities (α), (γ), (δ), and the following logical consequences of $\mathcal{AX}_{\mathtt{diff}}$:

$$\mathtt{diff}(a, b) = i \wedge rd(a, i) = e \wedge rd(b, i) = d \wedge e \neq d \to a \neq b.$$

To bring A in modularized form, we need further deletions: we keep only the write assertions $a = wr(c, I, E)$ in which I is minimized ($a = wr(c, I, E)$ is *minimized* iff for all true literals $a = wr(c, I', E')$, we have $I \subseteq I'$). The existence of such a minimized true 'write literal' comes directly from Lemma 2.2 (**Confl**ict): if $a = wr(c, I, E)$ and $a = wr(c, I', E')$ are both true, one can get a 'smaller' true literal by taking the intersection of I and I'. Lemma 2.2 (**Confl**ict) guarantees also that non-minimal write literals can be deduced (modulo $\mathcal{AX}_{\mathtt{diff}}$) from the minimal ones and the true 'read literals' concerning representatives we kept in A. It is trivial to check that now A is modularized. We do the same with L_2 and get a constraint B.

At this point, we are left with the problem of checking that the hypotheses of Proposition 3.5 are satisfied. Condition (O) and conditions (II)-(III) are trivial. Condition (I) is also

[16]We use results about termination of rewrite systems on infinite signatures given in Middeldorp - Zantema, "Simple termination of rewrite systems", Theoret. Comput. Sci., vol.175, pp.127-158 (1997) (the signatures here can be infinite because we can have infinitely many free constants coming from the supports of $\mathcal{M}_1, \mathcal{M}_2$).

trivial for the rewrite rules coming from (δ). Consider now a rewrite rule of the form (ϵ), i.e.

$$a \to wr(c, I, E)$$

and suppose that a is AB-common, i.e. that a is from $\mathtt{ARRAY}^{\mathcal{N}}$. Since $a > c$, we get that c is also from $\mathtt{ARRAY}^{\mathcal{N}}$ because of our definition of the precedence relation on the symbols of the signature. Since $a = wr(c, I, E)$ is true in $\mathcal{M}$, we have $\mathcal{M} \models |a - c| < \omega$; then applying Lemma A.6, we derive that $\mathcal{N} \models |a - c| < \omega$, i.e.

$$\mathcal{N} \models a = wr(c, J, D)$$

for some $J \subseteq \mathtt{INDEX}^{\mathcal{N}}, D \subseteq \mathtt{ELEM}^{\mathcal{N}}$. Since $\mathcal{N}$ is a substructure of $\mathcal{M}$, this implies that $\mathcal{M} \models a = wr(c, J, D)$; by minimization property of the 'write' literals from (ϵ), we get that $I \subseteq J$ and $E \subseteq D$, that is the equality $a = wr(c, I, E)$ is AB-common. $\square$

Remark A.8. (Added April 2011) We give here *a sketch for an alternative proof of Theorem A.7* depending only on Lemma A.6 (and not also on algorithmic facts like Proposition 3.5); the proof is interesting in itself because it is based on facts giving new insight into the models of $\mathcal{AX}_{\mathtt{diff}}$ and the embeddings between them.[17]

We saw in Remark A.1 that every model of $\mathcal{AX}_{\mathtt{diff}}$ (or even of $\mathcal{AX}$) is isomorphic to a *functional* model, i.e. to a model where the sort ARRAY is interpreted as a set of fuction, rd is interpreted as function application and wr as the update operation. Let us call *full* (or standard) a functional model where $\mathtt{ARRAY}^{\mathcal{M}}$ is the set of *all* functions of domain $\mathtt{INDEX}^{\mathcal{M}}$ and codomain $\mathtt{ELEM}^{\mathcal{M}}$. It is not dificult to see that in order to produce (up to isomorphism) *any* model $\mathcal{N}$ of $\mathcal{AX}_{\mathtt{diff}}$ it is sufficient to take a full model $\mathcal{M}$, to let $\mathtt{INDEX}^{\mathcal{N}} := \mathtt{INDEX}^{\mathcal{M}}$, $\mathtt{ELEM}^{\mathcal{N}} := \mathtt{ELEM}^{\mathcal{M}}$ and to let $\mathtt{ARRAY}^{\mathcal{N}}$ be equal to any subset of $\mathtt{ARRAY}^{\mathcal{M}}$ that is *closed under cardinality dependence* (in the sense that if $a \in \mathtt{ARRAY}^{\mathcal{N}}$ and $\mathcal{M} \models |a - b| < \omega$, then b is also in $\mathtt{ARRAY}^{\mathcal{N}}$): only in this way in fact, it is possible to define $wr^{\mathcal{N}}$ is such way that it is the restriction of $wr^{\mathcal{M}}$. A similar remark holds for embeddings: suppose that $\mu : \mathcal{N} \longrightarrow \mathcal{M}$ is an embedding that restricts to an inclusion $\mathtt{INDEX}^{\mathcal{N}} \subseteq \mathtt{INDEX}^{\mathcal{M}}$, $\mathtt{ELEM}^{\mathcal{N}} \subseteq \mathtt{ELEM}^{\mathcal{M}}$. Then, the action of the embedding μ on $\mathtt{ARRAY}^{\mathcal{N}}$ can be charaterized as follows: take an element a for each cardinality dependence equivalence class, extends arbitrarily a to the set $\mathtt{INDEX}^{\mathcal{M}} \setminus \mathtt{INDEX}^{\mathcal{N}}$ to produce $\mu(a)$ and then define $\mu(b)$ for non representative b in the only possible way for wr to be preserved (i.e. if $\mathcal{N} \models b = wr(a, I, E)$ for a representative a, let $\mu(b)$ be $wr^{\mathcal{M}}(\mu(a), I, E)$).

[17]It can be shown that Lemma A.6 holds also for the theory $\mathcal{AX}$, via the argument of footnote 15 (notice however that amalgamation is not sufficient for establishing quantifier free interpolation for theories like $\mathcal{AX}$ which are not universal - and in fact $\mathcal{AX}$ is amagamable but does not have quantifier free interpolation).

Armed with the above information, we produce now a direct proof of the amalgamation property. Take two embeddings $\mu_0 : \mathcal{N} \longrightarrow \mathcal{M}_0$ and $\mu_1 : \mathcal{N} \longrightarrow \mathcal{M}_1$; as we saw, we can freely suppose that $\mathcal{N}, \mathcal{M}_0, \mathcal{M}_1$ are functional models, that μ_0, μ_1 restricts to inclusions for the sorts $\mathtt{INDEX}$ and $\mathtt{ELEM}$, and that $(\mathtt{ELEM}^{\mathcal{M}_0} \setminus \mathtt{ELEM}^{\mathcal{N}}) \cap (\mathtt{ELEM}^{\mathcal{M}_1} \setminus \mathtt{ELEM}^{\mathcal{N}}) = \emptyset$, $(\mathtt{INDEX}^{\mathcal{M}_0} \setminus \mathtt{INDEX}^{\mathcal{N}}) \cap (\mathtt{INDEX}^{\mathcal{M}_1} \setminus \mathtt{INDEX}^{\mathcal{N}}) = \emptyset$. To simplify our task, we can also freely suppose that for $i = 0, 1$ there is some $e_i \in (\mathtt{ELEM}^{\mathcal{M}_i} \setminus \mathtt{ELEM}^{\mathcal{N}})$ and some $j_i \in (\mathtt{INDEX}^{\mathcal{M}_i} \setminus \mathtt{INDEX}^{\mathcal{N}})$ (i.e. that these sets are not empty).[18] The amalgamated model $\mathcal{M}$ will be the full model over $\mathtt{INDEX}^{\mathcal{M}_0} \cup \mathtt{INDEX}^{\mathcal{M}_1}$ and $\mathtt{ELEM}^{\mathcal{M}_0} \cup \mathtt{ELEM}^{\mathcal{M}_1}$. We need to define $\nu_i : \mathcal{M}_i \longrightarrow \mathcal{M}$ $(i = 0, 1)$ in such a way that $\nu_0 \circ \mu_0 = \nu_1 \circ \mu_1$. The only relevant point is the action of ν_i on $\mathtt{ARRAY}^{\mathcal{M}_i}$: as observed above, in order to define it, it is sufficient to extend any $a \in \mathtt{ARRAY}^{\mathcal{M}_i}$ to the indexes $k \in (\mathtt{ELEM}^{\mathcal{M}_{1-i}} \setminus \mathtt{ELEM}^{\mathcal{N}})$:

(I) we let the value $\nu_i(a)(k)$ be e_i in case there is no c such that $\mathcal{M}_i \models |a - \mu_i(c)| < \omega$;

(II) otherwise, we can do the following: take any such c such that $\mathcal{M}_i \models |a - \mu_i(c)| < \omega$ and put $\nu_i(a)(k) := \mu_{1-i}(c)(k)$.

Notice that because of Lemma A.6 the choice of c in (II) above is immaterial[19] and this guarantees that we have $\nu_1 \circ \mu_1 = \nu_2 \circ \mu_2$.

In order to define $\mathtt{diff}^{\mathcal{M}}$ we can just extend $\mathtt{diff}^{\mathcal{M}_1} \cup \mathtt{diff}^{\mathcal{M}_2}$ in such a way that axiom 3 holds. More precisely we let $\mathtt{diff}^{\mathcal{M}}(a, b)$ be as follows: (i) if for some $i = 0, 1$, we have that $a = \nu_i(a')$ and $b = \nu_i(b')$, then $\mathtt{diff}^{\mathcal{M}}(a, b)$ is taken to be $\mathtt{diff}^{\mathcal{M}_i}(a', b')$; (ii) otherwise it is defined to be any i such that $a(i) \neq b(i)$ (it is arbitrary if $a = b$). For this definition of $\mathtt{diff}^{\mathcal{M}}$ to be correct, we only need to show the following

Claim: *if $a = \nu_0(a_0) = \nu_1(a_1)$, then there is c such that $a_0 = \mu_0(c)$ and $a_1 = \mu_1(c)$.*

To prove the claim, suppose that $a = \nu_0(a_0) = \nu_1(a_1)$. Then $\nu_0(a_0)$ and $\nu_1(a_1)$ must have been defined as in (II) above (otherwise they cannot coincide with each other at indexes j_0, j_1), which means that there exists c_i such that for $i = 0, 1$ we have $\mathcal{M}_i \models |a_i - \mu_i(c_i)| < \omega$. Since $\nu_0(a_0) = a = \nu_1(a_1)$, this means that $\nu_0(\mu_0(c_0)) = \nu_1(\mu_1(c_0))$ and a differ only at finitely many indexes; the same is true for $\nu_1(\mu_1(c_1))$ and a, which in turns implies that $\nu_1(\mu_1(c_0))$ and $\nu_1(\mu_1(c_1))$ differ only at finitely many indexes too. The same consequently holds for c_0, c_1 in $\mathcal{N}$ too, for $\mu_0(c_0)$ and $\mu_0(c_1)$ in $\mathcal{M}_0$ and for $\mu_1(c_0)$ and $\mu_1(c_1)$ in $\mathcal{M}_1$. Thus, since the choice of c is (II) is immaterial, we can freely suppose that $c := c_0 = c_1$. Then, by (II)

[18] If this further condition is not satisfied, it is sufficient to enlarge $\mathcal{M}_1, \mathcal{M}_2$ so that they fulfill it.

[19] Any different such c' differs from c only on a finite set of indices in $\mathcal{M}_i$; by Lemma A.6 this holds in $\mathcal{N}$ too, thus we have $\mathcal{N} \models c' = wr(c, I, E)$ for some $I \subseteq \mathtt{INDEX}^{N}$. The latter implies that $\mu_{1-i}(c)$ and $\mu_{1-i}(c')$ cannot differ at any $k \in (\mathtt{ELEM}^{\mathcal{M}_{1-i}} \setminus \mathtt{ELEM}^{\mathcal{N}})$.

applied to the definition of $\nu_1(a_1)$, we have that $\nu_0(\mu_0(c)) = \nu_1(\mu_1(c))$ and $a = \nu_1(a_1)$ cannot differ at any $k \in (\mathtt{ELEM}^{\mathcal{M}_0} \setminus \mathtt{ELEM}^{\mathcal{N}})$. Similarly, $\nu_0(\mu_0(c)) = \nu_1(\mu_1(c))$ and a cannot differ at any $k \in (\mathtt{ELEM}^{\mathcal{M}_1} \setminus \mathtt{ELEM}^{\mathcal{N}})$. Thus a and $\nu_0(\mu_0(c)) = \nu_1(\mu_1(c))$ possibly differ only for $k \in \mathtt{INDEX}^{\mathcal{N}}$ and actually only for finitely many such k. But $a = \nu_0(a_0) = \nu_1(a_1)$, so the values of a at any $k \in \mathtt{INDEX}^{\mathcal{N}}$ belongs $\mathtt{ELEM}^{\mathcal{M}_0} \cap \mathtt{ELEM}^{\mathcal{M}_1} = \mathtt{ELEM}^{\mathcal{N}}$, which means that a is equal to $wr^{\mathcal{M}}(\nu_0(\mu_0(c)), I, E) = \nu_0(\mu_0(wr^{\mathcal{N}}(c, I, E)))$ for $I \subseteq \mathtt{INDEX}^{\mathcal{N}}$ and $E \subseteq \mathtt{ELEM}^{\mathcal{N}}$. In conclusion, we have that a is of the kind $\nu_0(\mu_0(\tilde{c})) = \nu_1(\mu_1(\tilde{c}))$ and from $a = \nu_0(a_0) = \nu_1(a_1)$, we get $a_0 = \mu_0(\tilde{c})$ and $a_1 = \mu_1(\tilde{c})$ because ν_0, ν_1 are injective. $\dashv$

A theory T is said to *admit quantifier free interpolants* iff for every pair of quantifier free formulae ϕ, ψ such that $\psi \wedge \phi$ is not T satisfiable, there exists a quantifier free formula θ such that: (i) ψ T-entails θ; (ii) $\theta \wedge \phi$ is not T-satisfiable: (iii) only variables occurring both in ψ and in ϕ occur in θ.

The following characterization is well-known[20] (but we nevertheless report a proof):

Theorem A.9. *Let T be universal; then T admits quantifier free interpolants iff T has the amalgamation property.*

Proof. Suppose first that T has amalgamation; let A, B be quantifier-free formulae such that $A \wedge B$ is not T-satisfiable. Let us replace variables with free constants in A, B; let us call Σ^A the signature Σ expanded with the free constants from A and Σ^B the signature Σ expanded with the free constants from B (we put $\Sigma^C := \Sigma^A \cap \Sigma^B$). For reductio, suppose that there is no ground formula C such that: (a) A T-entails C; (b) $C \wedge B$ is T-unsatisfiable; (c) only free constants from Σ^C occur in C.

As a first step, we build a maximal T-consistent set Γ of ground $\Sigma \cup \Sigma^A$-formulae and a maximal T-consistent set Δ of ground $\Sigma \cup \Sigma^B$-formulae such that $A \in \Gamma$, $B \in \Delta$, and $\Gamma \cap \Sigma^C = \Delta \cap \Sigma^C$.[21] For simplicity[22] let us assume that Σ is at most countable, so that we can fix two enumerations

$$A_1, A_2, \dots \qquad B_1, B_2, \dots$$

of ground $\Sigma \cup \Sigma^A$- and $\Sigma \cup \Sigma^B$-formulae, respectively. We build inductively Γ_n, Δ_n such that for every n (i) Γ_n contains either A_n or $\neg A_n$; (ii) Δ_n contains either B_n or $\neg B_n$; (iii) there is no ground $\Sigma \cup \Sigma^C$-formula C such that $\Gamma_n \cup \{\neg C\}$ and $\Delta_n \cup \{C\}$ are not T-consistent. Once this is done, we can get our Γ, Δ as $\Gamma := \bigcup \Gamma_n$ and $\Delta := \bigcup \Delta_n$.

[20]See P.D.Bacsich "Amalgamation properties and interpolation theorems for equational theories", Algebra Universalis, vol. 5, pp. 45-55, (1975).

[21]By abuse, we use Σ^C to indicate not only the signature Σ^C but also the set of formulae in the signature Σ^C.

[22]This is just to avoid a (straightforward indeed) transfinite induction argument.

We let Γ_0 be $\{A\}$ and Δ_0 be $\{B\}$ (notice that (iii) holds by (a)-(b)-(c) above). To build Γ_{n+1} we have two possibilities, namely $\Gamma_n \cup \{A_n\}$ and $\Gamma_n \cup \{\neg A_n\}$. Suppose they are both unsuitable because there are $C_1, C_2 \in \Sigma \cup \Sigma^C$ such that the sets

$$\Gamma_n \cup \{A_n, \neg C_1\}, \quad \Delta_n \cup \{C_1\}, \quad \Gamma_n \cup \{\neg A_n, \neg C_2\}, \quad \Delta_n \cup \{C_2\}$$

are all T-inconsistent. If we put $C := C_1 \vee C_2$, we get that $\Gamma_n \cup \{\neg C\}$ and $\Delta_n \cup \{C\}$ are not T-consistent, contrary to induction hypothesis. A similar argument shows that we can also build Δ_n.

Let now $\mathcal{M}_1$ be a model of Γ and $\mathcal{M}_2$ be a model of Δ. Consider the substructures $\mathcal{N}_1, \mathcal{N}_2$ of $\mathcal{M}_1, \mathcal{M}_2$ generated by the interpretations of the constants from Σ^C: since the related diagrams are the same (because $\Gamma \cap \Sigma^C = \Delta \cap \Sigma^C$), we have that $\mathcal{N}_1$ and $\mathcal{N}_2$ are Σ_C-isomorphic. Up to renaming, we can suppose that $\mathcal{N}_1$ and $\mathcal{N}_2$ are just the same substructure (let us we call it $\mathcal{N}$ for short). Since the theory T is universal and truth of universal sentences is preserved by substructures, we have that $\mathcal{N}$ is a model of T. By the amalgamation property, there is a T-amalgam $\mathcal{M}$ of $\mathcal{M}_1$ and $\mathcal{M}_2$ over $\mathcal{N}$. Now A, B are ground formulae true in $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively, hence they are both true in $\mathcal{M}$, which is impossible because $A \wedge B$ was assumed to be T-inconsistent.

Suppose now that T has quantifier free interpolants. Take two models $\mathcal{M}_1 = (M_1, \mathcal{I}_1)$ and $\mathcal{M}_2 = (M_2, \mathcal{I}_2)$ of T sharing a substructure $\mathcal{N} = (N, \mathcal{J})$. In order to show that a T-amalgam of $\mathcal{M}_1, \mathcal{M}_2$ over $\mathcal{N}$ exists, it is sufficient (by Robinson Diagram Lemma A.5) to show that $\delta_{\mathcal{M}_1}(M_1) \cup \delta_{\mathcal{M}_2}(M_2)$ is T-consistent. If it is not, by the compactness theorem of first order logic, there exist a $\Sigma \cup M_1$-ground sentence A and a $\Sigma \cup M_2$-ground sentence B such that (i) $A \wedge B$ is T-inconsistent; (ii) A is a conjunction of literals from $\delta_{\mathcal{M}_1}(M_1)$; (iii) B is a conjunction of literals from $\delta_{\mathcal{M}_2}(M_2)$. By the existence of quantifier-free interpolants, taking free constants instead of variables, we get that there exists a ground $\Sigma \cup N$-sentence C such that A T-entails C and $B \wedge C$ is T-inconsistent. The former fact yields that C is true in $\mathcal{M}_1$ and hence also in $\mathcal{N}$ and in $\mathcal{M}_2$, because C is ground. However, the fact that C is true in $\mathcal{M}_2$ contradicts the fact that $B \wedge C$ is T-inconsistent. $\square$

We underline that the hypothesis that T is universal is indispensable for the above result to hold. By Theorems A.7 and A.9, we can now conclude that

Theorem 5.4 *The theory $\mathcal{AX}_{\mathtt{diff}}$ admits quantifier-free interpolants.*

A.3 Satisfiability and Interpolation Algorithms

Theorem 5.4 is proved by semantic arguments, hence it does not give a direct interpolation algorithm (it only guarantees that, by enumerating quantifier free formulae, one can find sooner or later the desired interpolant).

The first step towards a practical interpolation algorithm for $\mathcal{AX}_{\texttt{diff}}$ is represented by the solver introduced in Section 4:

Theorem 4.1 *The solver from Section 4 decides constraint satisfiability in* $\mathcal{AX}_{\texttt{diff}}$.

Proof. Correctness and completeness of the solver are clear: since all steps and instructions from Section 4 manipulate the constraint up to $\exists$-equivalence, it follows that if all guessings originated by Step 3 fail, the input constraint is unsatisfiable and, if one of them succeed, the exhaustive application of the completion instructions leads to a modularized constraint which is satisfiable by Lemma 3.4.

We must only consider termination; to show that any sequence of our instructions terminates, we use a standard technique. With every positive literal $l = r$ we associate the multiset of terms $\{l, r\}$; with every negative literal $l \neq r$, we associate the multiset of terms $\{l, l, r, r\}$. Finally, with a constraint A we associate the multiset $M(A)$ of the multisets associated with every literal from A. Now it is easy to see that such multiset decreases after the application of any instruction. $\square$

The second ingredient of our interpolation algorithm for $\mathcal{AX}_{\texttt{diff}}$ are the metarules presented in Subsection 5.1. Propositions 5.1 and 5.2 stated in Subsection 5.1 are both proved in a straightforward way. The following remark can be useful:

Remark A.10. We underline that metarules are applied **bottom-up** whereas interpolants are computed (from an interpolating refutation) in a **top-down** manner. We should have labeled nodes in an interpolating metarules refutation by 4-tuples $(\Sigma^A, A, \Sigma^B, B)$, where Σ^A, Σ^B are signatures expanded with free constants, A is a Σ^A-constraint and B is a Σ^B-constraint. The *shared signature* of the node labeled $(\Sigma^A, A, \Sigma^B, B)$ (i.e. the signature where interpolants are recursively computed) is taken to be $\Sigma^C := \Sigma^A \cap \Sigma^B$; the *root signature pair* is the pair of signatures comprising all symbols occurring in the original pair of constraints. We did not make all this explicit in order to avoid notation overhead. Notice that the only metarules that modify the signatures are (Define0), (Define1), (Define2) (which add a to $\Sigma^A \cap \Sigma^B, \Sigma^A, \Sigma^B$, respectively). Some other rules like (ConstElim0), (ConstElim1), (ConstElim2) could in principle restrict the signature, but signature restriction is not relevant for the computation of interpolants: there is no need that *all* AB-common symbols occur in the interpolants, but we certainly do not want *extra* symbols to occur in them, so only bottom-up signature expansion must be tracked.

The interpolating algorithm for $\mathcal{AX}_{\texttt{diff}}$ is introduced in Subsection 5.2 and consists of specific Pre-Processing and Completion instructions. If we apply them exhaustively, starting from an initial pair of constraints (A, B), we produce a tree, whose nodes are labelled by pairs of constraints (the successors nodes of a node labelled $(\tilde{A}, \tilde{B})$ are labelled by pairs of constraints that are obtained from $(\tilde{A}, \tilde{B})$ by applying an instruction).[23] We called such a tree an *interpolating tree* for (A, B).

Theorem 5.3 *Any interpolation tree for (A, B) is finite; moreover, it is an interpolationg metarules refutation (from which an interpolant can be recursively computed according to Proposition 5.2) precisely iff $A \wedge B$ is $\mathcal{AX}_{\mathtt{diff}}$-unsatisfiable.*

Proof. Since all instructions can be justified by metarules and since our instructions bring any pair of constraints into constraints which are either manifestly inconsistent (i.e. contain $\perp$) or satisfy the requirements of Proposition 3.5, the second part of the claim is clear. We only have to show that all branches are finite (then König lemma applies).

A complication that we may face here is due to the fact that during instructions (γ), the signature is enlarged. However, notice that our instructions may introduce genuinely new AB-common array constants, however *they can only rename index constants, element constants and non AB-common array constants.* Moreover: (1) Term Sharing decreases the number of the constants which are not AB-common; (2) each call in the recursive procedure for the elimination of literals (15), *either* (2.i) renames to AB-common constants some constants which were not AB-common before, *or* (2.ii) just replaces a literal of the kind $c = wr(c', I_1 \cdot I_2, E_1 \cdot E_2)$ by the literals

$$c = wr(c', I_1, E_1), \qquad rd(c', I_2) = E_2$$

(see the first alternative following the guessing about truth of the literal $c = wr(c', I_1, E_1)$). Since there are only finitely many non AB-common constants at all, after finitely many steps neither Term Sharing nor (2.i) apply anymore. We finally show that instructions (α), (β) and (2.ii) (that do *not* enlarge the signature) cannot be executed infinitely many times either. To this aim, it is sufficient to associate with each pair of constraints $(\tilde{A}, \tilde{B})$ the complexity measure given by the multiset of pairs (ordered lexicographically) $\langle m(L), N_L \rangle$ (varying $L \in \tilde{A} \cup \tilde{B}$), where $m(L)$ is the multiset of terms associated with the literal L and N_L is 1 if $L \in \tilde{A} \setminus \tilde{B}$, 2 if $L \in \tilde{B} \setminus \tilde{A}$, and 0 if $L \in \tilde{A} \cap \tilde{B}$. In fact, the second component in the above pairs takes care of instructions (β), whereas the first component covers all the remaining instructions. Notice that it is important that, whenever an AB-common literal is deleted, the deletion is simultaneous in both components:[24] in fact, it can be shown (by

[23]The branching in the tree is due to instructions that need a guessing. Notice that Pre-Processing instructions are applied only in the initial segment of a branch.

[24]Otherwise, the (β) instruction could re-introduce it, causing an infinite loop (our complexity measure does not decrease if an AB-common literal is replaced by smaller literals only in the A- or in the B-component).

inspecting the instructions from the completion phase of Subsection 4.2) that whenever an AB-common literal is deleted, the instruction that removes it involves only AB-common literals, if undesired literals are removed first.[25] Thus, if instructions in (β) and (γ) have priority (as required by our specifications in Subsection 5.2), AB-common literal deletions caused by (α) can be performed both in the A- and in the B-component (notice also that the instructions from (β) and (2ii) do not remove AB-common literals). $\square$

[25]Let us see an example: consider instruction (C3). This instructions removes a literal $rd(a, i) \to e'$ using a literal $a \to wr(b, I, E)$ (and possibly rewrite rules $rd(b, i) \to d'$ as well as rewrite rules that might reduce some of the e', d', E). Now, if $rd(a, i) \to e'$ is AB-common and all the other involved rules are not undesired literals, the instruction as a whole manipulates AB-common literals. As such, if (β) has been conveniently applied, the instruction can be performed consecutively in the A- and in the B-component and our specification is precisely to do that.