Model Based Interpolation for Uninterpreted Functions and Integer Linear Arithmetic

Nikolaj Bjorner, Arie Gurfinkel, Sharon Shoham and Yakir Vizel

May 7, 2023

Model Based Interpolation for Uninterpreted Functions and Integer Linear Arithmetic

Nikolaj Bjørner, Arie Gurfinkel, Sharon Shoham, and Yakir Vizel

Microsoft Research and University of Waterloo and TAU and Technion

Abstract

Interpolants are central for symbolic model-checking and synthesis of invariants in program verification. Over the past couple of decades several methods have been developed for extracting interpolants from proofs. They benefit from close proof theoretic connections between interpolants and proofs, but require a theorem prover to produce proof objects in a format the interpolation procedure understands. It adds a dependency between interpolation procedures and proof formats of the automated theorem prover. Can the dependency on proof objects be relaxed for interpolant generation? *Model-based* interpolation generation relies on exchanging models of formulas and unsatisfiable cores. We develop an algorithm for model-based interpolation for the theory of quantifier-free integer linear arithmetic with uninterpreted functions. The setting extends previous results for linear real arithmetic and uninterpreted functions, but in contrast to the linear real case cannot rely on uniform interpolation.

1 Introduction

Algorithms for computing interpolants have received significant attention in the past, almost, two decades thanks to their use in symbolic model checking [30]. A *reverse* interpolant of an unsatisfiable formula $A[x,y] \wedge B[x,z]$, where formulas A, B share some vocabulary x, is a formula $I[x]$, such that $A[x,y] \Rightarrow I[x]$ and $B[x,y] \Rightarrow \neg I[x]$, for every x, y, z. Proof based methods rely on solvers that generate proof objects in a form where they are suitable for computing an interpolant [29, 4, 2, 19]. These methods require a deep integration with proof generation. Generally, the dependency on proof objects is a source of brittleness as interpolation extraction has to evolve in lock step not only new proof formats, but also new ways proof rules are combined. In this paper we take as starting point model-producing solvers to infer interpolants. Model producing solvers are routinely used for propositional interpolants in symbolic model checkers [5]. Our quest is an extension to a quantifier-free combination of uninterpreted functions and linear integer arithmetic, *EUFLIA*, where models enjoy a similar straight-forward interface as propositional logic: they assign values to terms.

Interpolation procedure. An interpolation procedure is shown in Algorithm 1. It assumes that $A \wedge \neg B \equiv \bot$. Initially, $I := \bot$. In each iteration, a new cube is added as a disjunct to I. The procedure maintains the invariant $B \Rightarrow \neg I$ and it terminates when $A \Rightarrow I$. As long as $A \wedge \neg I$ is satisfiable, a model $M \models A \wedge \neg I$ is extracted and used to compute a model-based projection *proj* of A. The projection *proj* has two requirements. First, to ensure progress *proj* $\wedge \neg I$ is satisfiable. Thus, *proj* is not already blocked by $\neg I$. Second, every satisfying model of *proj* can be extended to a satisfying model of A. Therefore it must be the case that $B \wedge$ *proj* is unsatisfiable with an unsatisfiable core *core*. Then I is updated as follows: $I := I \vee$ *core*.

Algorithm 1: Interpolation procedure.

```
$I := \bot$
while $A \land \neg I$ is SAT do
    Let $M \models A \land \neg I$
    $proj :=$ Project$(A, \neg I, M)$
    $core :=$ GetUnsatCore$(B, proj)$
    $I := I \lor core$
end while
return $I$
```

Beyond Propositional Interpolation We examine using model producing solvers for interpolating the theory of uninterpreted functions, *EUF*, the theory of integer linear arithmetic, *LIA*, and their combination, *EUFLIA*. Both *EUF* and *LIA* admit uniform interpolation, That is, given a formula $A[x, y]$ there is a strongest formula $I[x]$, such that $A[x, y] \Rightarrow I[x]$. Then, whenever $A[x, y] \land B[x, z]$ is unsat, it is also the case that $B[x, z] \Rightarrow \neg I[x]$. Model producing solvers can be used to compute these uniform interpolants. Musuvathi and Gulwani [16] considered the combination *EUFLRA* of *EUF* and linear *real* arithmetic, and presented a procedure that computes a uniform interpolant given a EUFLRA formula. For the combination *EUFLIA* there is generally not a finite uniform interpolant. General conditions for when uniform interpolants exist in combinations of theories have recently been investigated in [13].

Example 1. *Let A be the formula $p(a) \land s < a < t$, where s, t are shared integer symbols and p is a shared predicate, and let B_1 be the formula $\neg p(s+1) \land s+2 \simeq t$. Then a reverse interpolant of A and B_1 is $\neg B_1$. But there is no finite quantifier-free formula based on projecting a from A that can act as a uniform interpolant. For instance, setting B_2 to $\neg p(s+1) \land \neg p(s+2) \land s+3 \simeq t$, then $\neg B_1$ cannot be used as interpolant but $\neg B_2$ can.*

Thus, the combination of *EUFLIA* cannot be a simple composition of interpolation for *EUF* and *LIA*. Our model-based interpolation procedure only requires to interface with models and uses unsatisfiable core reduction. It does not take dependencies on proofs. It comes at the cost of requiring multiple rounds of satisfiability checks and the justification for correctness is not straight-forward. In light of the benefits and despite the drawbacks, the model-based approach has been our method of choice in fulfilling a frequent Z3 user request, which is to re-expose interpolation. The main technical contributions develop a model-based interpolation procedure for *EUFLIA* and as a main result we show that there is a terminating procedure that relies on only a model-producing oracle for computing interpolants.

A Combined Procedure by Example Before presenting the various components relevant for the result, let us illustrate a model-based interpolantion procedure on an example. The example is used in [6] where a proof-based interpolation procedure is developed.

Example 2.

$$\begin{array}{rcl} A & : & s \le 2a \le t \land f(a) \simeq q \\ B & : & t \le 2b \le s+1 \land f(b) \not\simeq q \end{array}$$

The conjunction $A \land B$ is unsat. We compute an interpolant I over the shared vocabulary $\Sigma_s = \{f, s, t\}$. Initially, I is $\bot$.

1. *Assume we are given a model* $[s = a = t = q = 0, f(0) = 0] \models A$. *Consistent with this interpretation, we can eliminate* a *from the arithmetic inequalities in* A*, producing a solution (witness)* $a \mapsto t \text{ div } 2$ *(*t *divided by 2) based on the upper bound* $2a \leq t$*, together with the constraints* $s \leq t \wedge \mathit{even}(t)$*, where* $\mathit{even}(t)$ *is a shorthand for* $2 \mid t$ *(2 divides* t*). The witness term is then substituted for* a *in the term* $f(a)$ *that appears in the EUF literal of* A*. The resulting projection of* A *is* $s \leq t \wedge \mathit{even}(t) \wedge f(t \text{ div } 2) \simeq q$.

2. *By construction,* B *is unsatisfiable in conjunction with the projection above. The projection forms a core for* B *(i.e., its conjunction with* B *is unsatisfiable): since* $s \leq t$ *implies that* $t \leq 2b \leq t + 1$*, and since* $\mathit{even}(t)$*,* b *must be* $t \text{ div } 2$ *leading to a contradiction on whether* $f(t \text{ div } 2) \simeq q$*. So* I *is set to* $I \leftarrow (s \leq t \wedge \mathit{even}(t) \wedge f(t \text{ div } 2) \simeq q)$.

3. *Next, assume* $M \models A \wedge \neg I$*, for* $M = [s = a = q = 0, t = 1, f(0) = 0]$*. It is no longer consistent with* M *to have* $\mathit{even}(t)$ *as part of the projection of the arithmetic inequalities, but we can solve for* a *in* A *as follows* $a \mapsto t \text{ div } 2$*, this time with the projection* $s < t \wedge f(t \text{ div } 2) \simeq q$ *of* A.

4. *The literals* $s < t \wedge f(t \text{ div } 2) \simeq q$ *form a core for* B *because* $t \leq s + 1$*, thus,* $t = s + 1$*, which implies that* $2b = t$*. If* t *is odd, then* $2b$ *must be odd, a contradiction. If* t *is even,* $b = t \text{ div } 2$*, hence* $f(t \text{ div } 2) \not\simeq q$*, which contradicts* $f(t \text{ div } 2) \simeq q$*. The core* $(s < t \wedge f(t \text{ div } 2) \simeq q)$ *is added to* I.

5. *A new model* $[s = a = q = 0, t = 2, f(0) = 0, f(1) = 1] \models A \wedge \neg I$ *is used to compute a projection of* A*. First, from the literals* $f(t \text{ div } 2) \not\simeq q, f(a) \simeq q$ *of* $A \wedge \neg I$ *satisfied by the model we extract the disequality* $a \not\simeq t \text{ div } 2$*, which we have to solve for together with the other arithmetic constraints in* A *in order to ensure that the projection of* A *would be consistent with* $\neg I$ *(to ensure progress). Next, from the upper bound* $2a \leq t$ *in* A *together with* $a \not\simeq t \text{ div } 2$*, we compute the solution* $a \mapsto (t - 2) \text{ div } 2$ *together with the inequality* $s \leq t - 2$ *implied by* $s \leq 2a$ *in* A*, obtaining the projection* $s \leq t - 2 \wedge f((t - 2) \text{ div } 2) \simeq q$.

6. *These literals are inconsistent with* B *as well since* B *requires* $t \leq s + 1$*. The core for* B *is* $(s + 2 \leq t)$*, and is added to* I.

7. $A \wedge \neg I$ *is no longer satisfiable. The final computed interpolant is*

$$(s \leq t \wedge \mathit{even}(t) \wedge f(t \text{ div } 2) \simeq q) \vee \quad (s < t \wedge f(t \text{ div } 2) \simeq q) \vee \quad (s + 2 \leq t)$$

2 Equalities and Uninterpreted Functions

In this section we present a procedure for generating uniform interpolants for conjunctions of equalities and disequalities in *EUF*. The idea is to use models to identify partitions of terms. We show how an arbitrary partition of terms that satisfy the theory of *EUF* can be projected onto a shared signature. The projection preserves all equalities over the shared signature, and enough disequalities to satisfy all disequalities in the original formula.

We consider many-sorted *EUF*. In the following let Σ_s, Σ_p be disjoint signatures. By convention, the signature Σ_s is used for *shared* constants and functions, and Σ_p is used for *private* functions and constants. Constants are simply nullary functions. We use $\vec{s}, \vec{p}$ for constants from Σ_s, Σ_p, respectively. We write Σ_{ps} for $\Sigma_s \cup \Sigma_p$. We denote by T_{Σ_v} the set of all terms over Σ_v.

In the sequel, assume $L = E \wedge D$, where E is a conjunction of equalities from `QF_UF` and D a conjunction of disequalities, both over the signature Σ_{ps}. We denote by $\mathcal{T}$ the set of all subterms of L. We assume that L is ground, thus existentially quantified variables are Skolemized to constants.

Congruence preserving partitions. Let M be a model that satisfies L. The model M induces a partition $\mathcal{P}$ of $\mathcal{T}$, or an equivalence relation on $\mathcal{T}$, such that terms t and s in $\mathcal{T}$ belong to the same (equivalence) class in the partition iff $M(s) = M(t)$. Since $M \models E$, the partition $\mathcal{P}$ *preserves congruences* with respect to the set of equalities E. That is,

1. For every equation $s \simeq t$ in E, the terms t and s belong to the same class.
2. For every term $t := f(t_1, \ldots, t_n)$ and $s := f(s_1, \ldots, s_n)$ in $\mathcal{T}$, if t_i and s_i are in the same partitions for every $1 \leq i \leq n$, then t and s are in the same class.

Efficient algorithms for computing a *congruence closure* [11] of the equalities in L over $\mathcal{T}$. Congruence closure equates terms t, t' only if $E \models t \simeq t'$; we call such terms *congruent*. Hence, a congruence closure induces the unique *finest* partition that is preserved by every model M of L (i.e., the equivalences it induces are satisfied by every model of L).

Example 3. *Suppose we are given the equalities $E := y \simeq a \wedge f(y) \simeq b$ over the shared signature $\Sigma_s = \{f, a, b\}$. The set of subterms is $\mathcal{T} := \{a, b, y, f(y)\}$ and the congruence closure is $\{\{y, a\}, \{f(y), b\}\}$. However, a model may also interpret a and b to be equal, in which case it would induce a coarser partition where all terms are in the same class.*

In general, we consider congruence preserving partitions induced by models rather than the finest partition induced by a congruence closure to facilitate combining *EUF* projection with an arithmetic projection as we do later. Namely, when *EUF* is combined with *LIA*, we need to consider equalities and disequalities that are consistent with arithmetic constraints. We use $\mathcal{P}$ computed from a model rather than E to refer to a set of equalities. Since $\mathcal{P}$ is induced by a model $M \models E$, it is always the case that $\mathcal{P} \models E$. We say that t and t' are $\mathcal{P}$-congruent if $\mathcal{P} \models t \simeq t'$.

Definition 1 (Representation function)**.** *For a congruence preserving partition $\mathcal{P}$ over $\mathcal{T}$ defined using the signature Σ_{ps}, and for $\Sigma_s \subseteq \Sigma_{ps}$, define a* representation *function,* $\mathsf{rep}_{\mathcal{P}}$ *as a partial mapping from terms in $\mathcal{T}$ to equivalence class representatives in T_{Σ_s}. Fix an arbitrary order $(\cdot \prec \cdot)$ on terms (to ensure uniqueness). The mapping is defined inductively as follows, for $t' \in \mathcal{T}$.*

- *If there exists $t \in \mathcal{T} \cap T_{\Sigma_s}$ such that t, t' are in the same partition then $\mathsf{rep}_{\mathcal{P}}(t')$ is the $\prec$-minimal such t.*
- *Otherwise, if there exists $t := f(t_1, \ldots, t_n) \in \mathcal{T} \setminus T_{\Sigma_s}$ such that t, t' are in the same partition, $f \in \Sigma_s$, and for every i, $\mathsf{rep}(t_i)$ is defined, then $\mathsf{rep}_{\mathcal{P}}(t') = f(\mathsf{rep}_{\mathcal{P}}(t_1), \ldots, \mathsf{rep}_{\mathcal{P}}(t_n))$ for the $\prec$-minimal such t.*

Note that whenever $\mathsf{rep}_{\mathcal{P}}(t')$ is defined, then $\mathcal{P} \models t' \simeq \mathsf{rep}(t')$. In the following, we simply write rep instead of $\mathsf{rep}_{\mathcal{P}}$ when $\mathcal{P}$ is understood from context.

Example 4. *Let $E := y \simeq b \wedge f(y, a) \simeq f(a, y)$, where $\Sigma_s = \{a, b, f\}$. Then from a congruence closure of E we obtain representatives $\mathsf{rep}(y) = b$ and $\mathsf{rep}(f(y, a)) = \mathsf{rep}(f(y, a)) = f(a, b)$.*

The following lemma is helpful in the sequel. In particular, it ensures that a representative exists for every $t' \in \mathcal{T}$ that is $\mathcal{P}$-congruent to a term over Σ_s, even if the latter is not in $\mathcal{T}$. Since $\mathsf{rep}(t') \in T_{\Sigma_s}$ (whenever it is defined) and $\mathcal{P} \models t' \simeq \mathsf{rep}(t')$, this means that $\mathsf{rep}(t')$ is defined if and only if $\mathcal{P} \models t \simeq t'$ for some $t \in T_{\Sigma_s}$.

Lemma 1. *If $\mathcal{P} \models t \simeq t'$ for $t \in T_{\Sigma_s}$ and $t' \in \mathcal{T}$, then (i) each subterm of t is $\mathcal{P}$-congruent to some term in $\mathcal{T}$, and (ii) $\mathsf{rep}(t')$ is defined.*

All proofs in this section are deferred to Appendix A.

We use representatives to project L to the shared vocabulary. The projection consists of an equality projection and a disequality projection, both computed based on a congruence preserving partition $\mathcal{P}$.

Definition 2 (Equality Projection)**.** *Let $M \models L$ and let $\mathcal{P}$ be the partition induced by M over $\mathcal{T}$. The equality projection $\mathsf{rep}^{=}_{M}(L)$ of L onto T_{Σ_s} with respect to M is defined as follows: If $t := f(t_1, \ldots, t_n) \in \mathcal{T}$, $f \in \Sigma_s$ and $\mathsf{rep}_{\mathcal{P}}(t_i)$ is defined for each t_i, then $\mathsf{rep}^{=}_{M}(L)$ contains*

$$f(\mathsf{rep}_{\mathcal{P}}(t_1), \ldots, \mathsf{rep}_{\mathcal{P}}(t_n)) \simeq \mathsf{rep}_{\mathcal{P}}(t).$$

Note that Definition 2 implies that for every $t \in \mathcal{T}$, if $t \in T_{\Sigma_s}$, then $\mathsf{rep}^{=}_{M}(L)$ contains $t \simeq \mathsf{rep}(t)$. The base case is where t is a constant, i.e., a 0-ary function, in Σ_s. Furthermore, $\mathsf{rep}^{=}_{M}(L)$ does not necessarily contain equalities from E that are already over T_{Σ_s}. For example, E can be $a \simeq b, b \simeq c$, but $\mathsf{rep}^{=}_{M}(L)$ is $a \simeq b, a \simeq c$.

Example 5. *From the equalities in Example 3 and a model that only equates congruent terms, we obtain the equality projection $E' := b \simeq f(a)$.*

We next consider the disequalities determined by the partition. Projections for disequalities are no longer uniquely determined, even in the context of a partition.

Example 6. *The disequality $g(a, b) \not\simeq g(c, d)$ for $a, b, c, d \in \Sigma_s$ and $g \in \Sigma_p$ implies $a \not\simeq c \vee b \not\simeq d$.*

We therefore define a notion of *disequality certificate*. It is a sufficient set of disequalities that when preserved by a projection ensure that every model of the projection extends to a model of the disequalities being certified.

Definition 3 (Disequality Certificate)**.** *Let $\mathcal{P}$ be a congruence preserving partition of $\mathcal{T}$, and let $D' \in \mathcal{T}$ be a set of disequalities between terms in $\mathcal{T}$ that belong to different classes of $\mathcal{P}$. A* disequality certificate*, $DCert_{\mathcal{P}}(D')$, for $\mathcal{P}$ and D' is a set of disequalities closed under the following rules:*

- *$D' \subseteq DCert_{\mathcal{P}}(D')$.*
- *For each $(t \not\simeq t') \in DCert_{\mathcal{P}}(D')$, for every $f \in \Sigma_{ps}$, and matching applications $f(t_1, \ldots, t_n) \in \mathcal{T}$ and $f(t'_1, \ldots, t'_n) \in \mathcal{T}$, such that $f(t_1, \ldots, t_n)$ is in the same class as t and $f(t'_1, \ldots, t'_n)$ is in the same class as t', there is an index i, such that t_i and t'_i belong to different classes, and $(t_i \not\simeq t'_i) \in DCert_{\mathcal{P}}(D')$.*

Note that a disequality certificate always exists because $\mathcal{P}$ is congruence preserving. Congruence ensures the existence of a distinguishing argument for matching function applications that are disequal according to $\mathcal{P}$. Disequality certificates are not necessarily unique because there could be several choices for the index t_i. We define the *function $DCert_M(D')$* for a model M by using the partition $\mathcal{P}$ obtained from M and breaking ties by taking the smallest index i where t_i, t'_i belong to different sets from $\mathcal{P}$.

Example 7. *Let $M \models f(x) \simeq a \ \wedge\ f(y) \simeq b \ \wedge\ f(z) \simeq b \ \wedge\ a \not\simeq b$, and $\mathcal{P}$ its induced partition, then $DCert_{\mathcal{P}}(a \not\simeq b) = \{x \not\simeq y, x \not\simeq z, a \not\simeq b\}$.*

Disequality certificates are needed for the original disequalities in D, but also for implicit disequalities between *conflicting terms*:

Definition 4 (Conflicting Terms)**.** *Terms $t, t' \in \mathcal{T}$ are conflicting in $\mathcal{P}$ if they belong to different classes of $\mathcal{P}$ and they are both applications of the same function symbol, i.e., there exists $f \in \Sigma_{ps}$ such that $t = f(t'_1, \ldots, t'_n)$ and $t' = f(t'_1, \ldots, t'_n)$.*

Disequalities between conflicting terms may arise even when D is empty, and need to be taken into account in projections, as demonstrated by the following example.

Example 8. *Let $E := g(a) = b \wedge g(a') = b'$ for $a, b \in \Sigma_s$ and $g \in \Sigma_p$. Then E implies $a \not\simeq a' \vee b \simeq b'$. The disequality $a \not\simeq a'$ will be obtained from a disequality certificate for $g(a), g(a')$ in a partition that assigns them to different classes.*

Definition 5 (Disequality Projection)**.** *Let $M \models L$, where $L = E \wedge D$ as before, and let $\mathcal{P}$ be the partition induced by M over $\mathcal{T}$. The disequality projection $\mathsf{rep}^{\neq}_M(L)$ of L onto T_{Σ_s} with respect to M is given by*

$$\mathsf{rep}^{\neq}_M(L) := \left\{ \mathsf{rep}_{\mathcal{P}}(t) \not\simeq \mathsf{rep}_{\mathcal{P}}(t') \ \middle|\ \begin{array}{l} (t \not\simeq t') \in DCert_{\mathcal{P}}(D \cup C), \\ \mathsf{rep}_{\mathcal{P}}(t), \mathsf{rep}_{\mathcal{P}}(t') \text{ are both defined} \end{array} \right\}$$

where C is the set of all disequalities between conflicting terms in $\mathcal{P}$.

When combined with the equality projection, the disequality projection results in a *EUF* model projection:

Definition 6 (*EUF* Model Projection)**.** *Let $L = E \wedge D$ be a quantifier-free* EUF *conjunction of equalities and disequalities over $T_{\Sigma_{ps}}$ with model M. A EUF* model projection *onto T_{Σ_s} is defined as $\mathsf{rep}_M(L) := \mathsf{rep}^{=}_M(L) \wedge \mathsf{rep}^{\neq}_M(L)$.*

Every model satisfying $\mathsf{rep}_M(L)$ can be adapted to a model that satisfies L and the same set of equalities over T_{Σ_s}:

Lemma 2. *Let L as above, $M \models L$, and $M' \models \mathsf{rep}_M(L)$. Then, there is a model M'', such that $M'' \models L$ and $M''(t) = M'(t)$ for every $t \in T_{\Sigma_s}$.*

Note that M'' may interpret shared symbols differently than M', but both interpret terms over the shared signature in the same way. For example, suppose that $L := p(x)$, where p is a shared predicate symbol and x is private. Then $\mathsf{rep}_M(L) = \top$ (regardless of M). In particular, a model M' that interprets p as an empty relation satisfies $\mathsf{rep}_M(L)$, yet it cannot be extended to a model that satisfies $p(x)$. It can, however, be adapted into a model M'' that satisfies $p(x)$ and agrees with M' on equalities between (the empty set of) shared terms by interpreting p as a nonempty relation that includes the interpretation of x.

Since the set of *EUF* model projections of L is finite (as there are finitely many representatives for $\mathcal{T}$), we conclude the following.

Theorem 1 (Uniform Interpolation for *EUF*)**.** *Let L be a quantifier-free conjunction of* EUF *literals. There is a finite set of conjunctions $\{L_i\}$ over T_{Σ_s}, such that*

- $L \models \bigvee_i L_i$.

- *For every quantifier-free φ over Σ_s, $L \models \varphi$ if and only if $\bigvee_i L_i \models \varphi$.*

Example 9. *Take the following two formulas from [12]:*

$$\begin{aligned} A \quad &: \quad z_1 \simeq a_1 \wedge z_2 \simeq a_2 \wedge z_3 \simeq f(a_1) \wedge z_4 \simeq f(a_2) \wedge \\ & \quad\quad z_5 \simeq a_3 \wedge z_6 \simeq a_4 \wedge z_7 \simeq f(a_3) \wedge z_8 \simeq f(a_4) \\ B \quad &: \quad z_1 \simeq z_2 \wedge z_5 \simeq f(z_3) \wedge f(z_4) \simeq z_6 \wedge b_1 \simeq z_7 \wedge z_8 \simeq b_2 \wedge b_1 \not\simeq b_2 \end{aligned}$$

The shared symbols Σ_s are $\{z_1, \ldots, z_8, f\}$. The congruence closure over A does not introduce new equalities, but allows to compute representatives over Σ_s. Thus,

$$\mathsf{rep}(A) := z_8 \simeq f(z_6) \wedge z_7 \simeq f(z_5) \wedge z_3 \simeq f(z_1) \wedge z_4 \simeq f(z_2)$$

For B, $DCert(b_1 \not\simeq b_2)$ includes $z_7 \not\simeq z_8$, and based on the congruence closure over B we obtain:

$$\mathsf{rep}(B) := z_7 \not\simeq z_8 \wedge z_1 \simeq z_2 \wedge z_5 \simeq f(z_3) \wedge f(z_4) \simeq z_6$$

They are uniform (reverse) interpolants.

3 Arithmetic and Model-Based Witness Extraction

Linear integer arithmetic (*LIA*) admits quantifier elimination [32, 31]. This provides a tool for computing a uniform interpolant. Given a *LIA* formula $Ar[x, \vec{y}]$, quantifier elimination produces a formula $(\bigvee_i Ar_i[\vec{y}])$ such that $(\bigvee_i Ar_i[\vec{y}]) \equiv \exists x \,.\, Ar[x, \vec{y}]$. Models can be used as a guide to produce only the satisfiable Ar_i [25, 1]. The same procedure can produce witness terms for x, that can be substituted into the original formula to eliminate x [26, 33]. Quantifier elimination requires, beyond linear addition and inequalities, the use of a divisibility predicate $(d \mid t)$ (d divides t), where d is a positive integer value[1] and t is a linear term. Witnesses use terms $(t \operatorname{div} d)$ (the integer floor of t/d). Generally we consider a class of witness terms generated by the following grammar

$$w \quad ::= \quad t \;\mid\; w + a \;\mid\; b \cdot w \;\mid\; w \operatorname{div} b \tag{3.1}$$

where a, b are integer values, b is positive and divides the product of all values that appear an original formula Ar, and t is a term in Ar.

Theorem 2 summarizes the main properties needed for our main results.

Theorem 2. *For a formula $Ar[x, \vec{y}]$ over* LIA *there is a finite set of projections $\langle w_i, Ar_i \rangle$, such that*

$$\exists x \,.\, Ar[x, \vec{y}] \;\equiv\; \bigvee_i Ar[w_i[\vec{y}], \vec{y}] \;\equiv\; \bigvee_i Ar_i[\vec{y}] \tag{3.2}$$

Furthermore, for each i, $Ar_i \models Ar[w_i[\vec{y}], \vec{y}] \models \exists x \,.\, Ar$.

Let us assume that Ar is normalized such that all occurrences of x are isolated as $l \leq ax$, $bx \leq u, a \mid (bx + s)$, where $x \notin l, u, s$, a, b are positive integers. Then the terms w_i are generated by the grammar (3.1).

[1]We refer to interpreted arithmetic constants as *values* to distinguish them from uninterpreted arithmetic constants (or variables).

Quantifier elimination, extracting witness terms and model-based quantifier elimination for *LIA* have been well-studied. A self-contained proof of Theorem 2 is given in Appendix B.

Example 10. *To illustrate the equivalence of (3.2) consider the formula* $\exists a \,.\, s \le 2a \le t$ *from the introduction. Projection produced the witness pairs* $\langle t \text{ div } 2, s \le t \wedge 2 \uparrow t\rangle$ *and* $\langle t \text{ div } 2, s < t\rangle$*. The corresponding equivalences are:*

$$(\exists a \,.\, s \le 2a \le t) \;\equiv\; s \le 2(t \text{ div } 2) \le t \;\equiv\; ((s \le t \wedge 2 \uparrow t) \vee s < t)$$

Substituting in witness terms into $s \le 2a \le t$ *results in just one formula because the witness terms are the same in the two cases. On the other hand, our reference model-based projection procedure uses two formulas depending on whether* t *is even or odd.*

The shape of witness terms by (3.1) is justified by induction on witness terms produced by projection, see Appendix B. The important property of the grammar for witness terms is that it is closed under adding disequalities between witness terms and the variable they replace. For example, an inequality $x \not\simeq (t+a) \text{ div } b$ is turned into an inequality $x \ge 1 + (t+a) \text{ div } b$ or $x \le ((t+a) \text{ div } b) - 1$ during projection based on how a model M shows disequality between x and $(t+a) \text{ div } b$. The lower or upper bound could be used in a new witness term. Then new witness term is obtained from an old witness by adding or subtracting an offset.

4 *EUFLIA*

Theorems 1 and 2 establish that the quantifier free theories *EUF* and *LIA* separately admit uniform quantifier-free interpolation. Their combination, on the other hand, does not enjoy such a property.

Example 11. *Consider the formula* $A := p(a) \wedge s < a < t$ *from Example 1, where* s, t, p *are shared and* a *is private. Independent of the model* M *satisfying* A*, a projection of* $s < a < t$ *takes the shape* $\langle s+1, s+1 < t\rangle$*. Substituting the witness* $s+1$ *into* $p(a)$ *produces the joint projection* $p(s+1) \wedge s+1 < t$ *for* A*. It is not a uniform interpolant, nor is it an interpolant w.r.t.* $B := \neg p(s+1) \wedge s+2 \simeq t$*. Recall that for both* EUF *and* LIA *uniform interpolants were obtained by the (finite) disjunction of model-based projections. This example demonstrates that for* EUFLIA *formulas, even though there are still finitely many (joint) projections, they do not immediately give rise to interpolants.*

Nevertheless, as we show here, it is possible to finitely compute interpolants over the combined signature for every (reverse) interpolation problem $A \Rightarrow I$, $B \Rightarrow \neg I$ by carefully managing projections and cores. In the remainder of the section, we formally define joint projections for *EUFLIA* (Section 4.2), and show how they can be used to compute interpolants (Section 4.3). The next section introduces a formal tool, purification, that allows us to treat compound terms as constant symbols.

4.1 Purification

Purified formulas are of the form $L \wedge \mathit{Defs}$, where L is a conjunction of literals where the only terms of sort integer are uninterpreted constant symbols and Defs is a conjunction of equalities $v \simeq f(\vec{v})$ that define the arithmetic constant symbol v using constant symbols $\vec{v}$ and function f (which can be uninterpreted or arithmetical). In the sequel, we refer to uninterpreted constants of sort integer as arithmetic constants. Such constants are considered as both *LIA* terms and *EUF* terms.

Example 12. *A normalized form of $f(x) \simeq y+1 \wedge p(g(x))$ is $v_1 \simeq v_2 \wedge p(v_3) \wedge v_1 \simeq f(x) \wedge v_2 \simeq y+1 \wedge v_3 \simeq g(x)$.*

This format paves the way for theory local decomposition:

Definition 7 (Purified Normal Form)**.** *We say that a formula over* EUFLIA *is in* purified normal form *if it is of the form $Ar \wedge F$, where $Ar := L_{Ar} \wedge Defs_{Ar}$ is a conjunction of literals and definitions over* LIA *and $F := L_F \wedge Defs_F$ is a conjunction of literals and definitions over* EUF *(potentially including uninterpreted arithmetic constants). The only symbols that may appear in both Ar and F are (uninterpreted) arithmetic constants.*

Example 13. *The purified normal form of the Example 12 uses $Ar := v_1 \simeq v_2 \wedge v_2 \simeq y+1$, $F := p(v_3) \wedge v_1 \simeq f(x) \wedge v_3 \simeq g(x)$. Alternatively, the equality $v_1 \simeq v_2$ could be part of F, or be repeated in both F and Ar.*

4.2 Joint Projection

The joint projection of *EUFLIA* combines arithmetic projection (Section 3) with *EUF* projection (Section 2). An outline of our method for joint projection of A in *EUFLIA* is as follows. Each projection is computed through a sequence:

- Extract a satisfiable conjunction of literals $Ar \wedge F$ that implies A and is purified; such that Ar uses only arithmetical symbols and F constrains only uninterpreted symbols, and such that Ar and F both contain enough equalities and disequalities over (uninterpreted) arithmetic constants to ensure that models for the separate formulas can be combined. Let $M \models Ar \wedge F$.
- Project each private arithmetic constant from Ar based on M, producing witness terms $\vec{w}$ and formula Ar' using the procedure from Section 3.
- Produce F_1 by projecting private functions and constants from F based on M using the procedure from Section 2, except for private arithmetic constants that appear in Ar and are temporarily added to the shared vocabulary Σ_s.
- Substitute the resulting witness terms $\vec{w}$ for each private arithmetic constant in F_1, producing F'.
- The resulting projection of A is $Ar' \wedge F'$.

It is important not to under-constrain Ar when performing projection. For example, suppose $A = Ar \wedge F$ for $F = f(x) \simeq a \wedge f(y) \simeq b \wedge a \not\simeq b$, where x, y are arithmetic constants. The constraint $x \not\simeq y$ is entailed by F. If it is not included in Ar when x, y are projected, it could result in the witness 0 for both x and y and resulting in a joint projection of A that is not satisfiable. A similar phenomenon may happen with equalities. To ensure that projections of Ar do not contradict F we therefore strengthen Ar with equalities and disequalities over arithmetic constants that appear in both Ar and F based on a model M.

Definition 8 (EUF completion)**.** *Let $M \models Ar \wedge F$, and let D' be the set of all disequalities in F together with disequalities between conflicting terms in the partition induced by M on the terms of F (see Definition 4). Let $\vec{a}$ denote the private arithmetic constants that appear in both Ar and F. The* EUF completion *of Ar w.r.t. M and F, denoted $U_M(Ar, F)$, consists of the following equalities and disequalities over $\vec{a}$:*

- $\{a_i \not\simeq a_j \mid t \not\simeq t' \in DCert_M(D') \wedge M(a_i) = M(t) \wedge M(a_j) = M(t')\}$, *and*
- $\{a_i \simeq a_j \mid M(a_i) = M(a_j)\}$.

EUF completion provides a sufficient condition for solutions from projecting Ar not to contradict F.

Lemma 3 (Theory Combination)**.** *Let $M \models Ar \wedge F$ and let $\langle \vec{w}, Ar' \rangle$ be a projection of $Ar \wedge U_M(Ar, F)$. Then $\vec{w} \simeq \vec{a}$ is satisfiable with F.*

We can now formalize the definition of *Joint Projection* that was sketched in the outline.

Definition 9 (Joint Projection)**.** *Given a formula A and a model $M \models A$ such that $A := Ar \wedge F$ is in purified normal form, with private arithmetic constants $\vec{a}$, let $\langle \vec{w}, Ar' \rangle$ be witness terms and a projection onto Σ_s for $Ar \wedge U_M(F, Ar)$ and M based on Theorem 2 and let F_1 be a projection onto $\Sigma_s \cup \vec{a}$ for F and M using Theorem 1, and let $F' := F_1[\vec{w}]$. Then $Ar' \wedge F'$ is a* joint projection *of A based on M.*

Lemma 4 (Projection preserves satisfiability)**.** *Let $M \models Ar \wedge F$ and let $Ar' \wedge F'$ be a corresponding joint projection. Then $Ar' \wedge F'$ is satisfiable. Further, if $M' \models Ar' \wedge F'$, then there exists a model M'' such that $M'' \models Ar \wedge F$, and $M''(t) = M'(t)$ for every term $t \in T_{\Sigma_s}$.*

4.3 Interpolation for *EUFLIA*

We generate interpolants for *EUFLIA* following the recipe of Algorithm 1. At a high level we iteratively compute model-based projections $proj_1, proj_2, \ldots$ of A, and accumulate their cores $core_1, core_2, \ldots$ for B as disjuncts in I. As illustrated by Example 11, a naive implementation does not work. A summary of sufficient criteria for ensuring progress, and ultimately termination is:

Consistent projections Projections of A need to be consistent with $\neg I$. Ensuring this requires strengthening A before projection.

Minimal cores Projection of A and core extraction w.r.t B produces minimal cores $core_i[\vec{w}_i]$, obtained from $core_i[\vec{a}]$ by substituting witnesses $\vec{w}_i$ for $\vec{a}$.

Finite basis The cores $core_i[\vec{a}]$ (before witness terms are substituted) are drawn from the finite set of projections of A.

Distinct witnesses An unbounded sequence of cores contains an unbounded set of necessarily distinct witness terms: they contain different offset values and can therefore not be equal.

Essential witnesses Uses of $f(w_{ij})$ in projections are *essential*: replacing $f(w_{ij})$ with a fresh variable and projecting makes the projection consistent with B.

Finite arithmetic projections There is a finite set of distinct other occurrences of w_{ij} (not inside an uninterpreted function) in arithmetical literals.

Projections that are consistent with $\neg I$ are needed to ensure progress. The idea with forcing enough distinct w_{ij} is that B cannot block an arbitrary number of different $f(w_{ij})$. It only contains a finite set of subterms with f. If there is an unbounded number of distinct w_{ij} then for some core literal the occurrence of $f(w_{ij})$ is indistinguishable from a fresh variable when it comes to B. If the occurrences of $f(w_{ij})$ are essential and indistinguishable from a fresh variable, $core_i$ cannot be a minimal core. This is a contradiction. The other possible scenario for an infinite set of distinct w_{ij} is when they occur exclusively in arithmetic constraints. We describe an approach to ensure this does not happen either. We now examine conditions for enforcing the above criteria.

4.3.1 Consistent Projections and Distinct Witnesses

In contrast to the theories EUF and LIA in isolation, Example 11 illustrates that projecting A with respect to models of $A \wedge \neg I$ may produce projections that are inconsistent with $\neg I$, hindering progress of the interpolation procedure. To avoid that problem, we track equalities between arithmetic constants from $\vec{a}$ and witness terms $\vec{w}$ during projection. For this purpose we define

Definition 10 (Witness Protection Set)**.** *For a model M of $\bigwedge_i \neg core_i$ select one literal that is satisfied by M from each clause $\neg core_i$. The set of these literals is called Ls. For a sequence of witness terms $\vec{w}_i$ for private arithmetic constants $\vec{a}$ among the selected literals define the* witness protection set D *as*

$$D := \bigwedge_{i,j} \{a_j \not\simeq w_{ij} \mid w_{ij} \in \vec{w}_i, f(w_{ij}) \text{ a subterm in } Ls, M(a_j) \neq M(w_{ij})\}.$$

Example 14. *For A and $proj := p(s+1) \wedge s+1 < t$ from Example 11, a model $M' \models A \wedge \neg proj$ must be such that $M'(a) \neq M'(s+1)$, introducing the disequality $a \not\simeq s+1$ to the witness protection set. When we apply the second projection on $A \wedge a \not\simeq s+1$ rather than on A, we are guaranteed to obtain a different projection, e.g., $proj_2 := p(s+2) \wedge s+2 < t$.*

Adding the witness protection set when computing projections ensures progress of Algorithm 1, while also ensuring that models of the projection can be adapted to models of $A \wedge \neg I$ where $\neg I = \bigwedge_i \neg core_i[\vec{w}_i]$.

Lemma 5 (Witness Protection)**.** *Let $M \models A \wedge \bigwedge_i \neg core_i[\vec{w}_i]$, where $\neg core_i[\vec{a}]$ is a negated core based on projections from A and $\neg core_j[\vec{w}_j]$ for $j < i$ and uses only subterms from Σ_{sa}. Assume $Ar' \wedge F'$ is a projection with respect to M of $A \wedge D$, where D is a witness protection set. Then satisfiability of $B \wedge Ar' \wedge F'$ implies satisfiability of $B \wedge A \wedge \bigwedge_i \neg core_i[\vec{w}_i]$.*

Due to the witness protection sets, the projections computed by Algorithm 1 may introduce new terms. This makes the sequence of projections potentially infinite. For example, in the case of $A := p(a) \wedge s < a < t$, iteratively computing projections can yield the infinite sequence $\{p(s+i) \wedge s+i < t\}_{i \in \mathbb{N}}$, which is in line with the property that a uniform interpolant does not exist for A.

Next we establish that for every B only a finite set of terms need to be introduced. Thus, considering cores with respect to B ensures that Algorithm 1 terminates.

Example 15. *For $A := p(a) \wedge s < a < t$ and $B := \neg p(s+1) \wedge s+2 \simeq t$, the core of the first projection $proj_1 := p(s+1) \wedge s+1 < t$ is $core_1 := p(s+1)$, and the core of the second projection $proj_2 := p(s+2) \wedge s+2 < t$ is $core_2 := s+2 < t$, yielding the interpolant $I := p(s+1) \vee s+2 < t$.*

We first formally define the projections and core sequences produced by the interpolation algorithm and establish partial correctness.

Definition 11 (Projection/Core Sequences)**.** *Let $A_1 := Ar \wedge F$ be a* EUFLIA *formula, and for $i \geq 1$,*

- *M_i a model of A_i,*
- *D_i a witness protection set for A_i (based on M_i and witness terms $\vec{w}_1, \dots, \vec{w}_{i-1}$),*
- *A'_i a projection of $A_1 \wedge D_i$ (based on M_i),*
- *$core_i \subseteq A'_i$, and*
- *$A_{i+1} := A_i \wedge \neg core_i$.*

Then $A'_1, A'_2, \dots$ is a projection sequence *and $core_1, core_2, \dots$ its corresponding* core sequence*. They are defined together so we refer to the sequences as a* projection/core sequence*.*

Lemma 4 directly implies that the projections are always contradictory with B provided $A \wedge B$ is unsatisfiable. Thus, a terminating sequence of cores produces a correct interpolant.

Corollary 1 (Partial Correctness)**.** *If a projection and core sequence terminates with A_k unsatisfiable, where the sequence of cores is $core_1, \dots, core_k$, then $I := core_1 \vee core_2 \vee \dots \vee core_k$ forms a reverse interpolant such that $A \Rightarrow I$ and $B \Rightarrow \neg I$.*

Next we set out to establish termination of the interpolation algorithm. The set of witness terms are not directly limited, but if a projection/core sequence produces an infinite set of witness terms, it contains an infinite subset of distinct witnesses.

Lemma 6 (Term Diversification)**.** *For an infinite sequence of witness terms $\vec{w}_1, \vec{w}_2, \dots$, originating from a projection/core sequence, there is an index $0 \leq j < |\vec{w}|$ into the tuple $\vec{w}$, such that $w_{1j}, w_{2j}, w_{3j}, \dots$ contains an infinite subset that has distinct values under every interpretation.*

Recall that witness terms are substituted for private symbols in A. We consider the special case where a private symbol a occurs under $f(a)$; generalization to nested terms and multi-arity functions is straight-forward. Then following substitution the projection contains terms of the form $f(w_k)$. These terms may *collide* with terms in B in the sense that f cannot be given an interpretation on $f(w_k)$ independently of satisfying B. But when B is quantifier free formula over *EUFLIA*, this situation is only possible for a finite set of witness terms. Lemma 7 formulates a condition that ensures interpretations of $f(w_k)$ can eventually be defined independently of B.

Lemma 7 (Compactness)**.** *Let B be a satisfiable quantifier-free formula over* EUFLIA*, t be a fixed term, and $w_1, \dots, w_k, \dots$ be an infinite set of necessarily distinct witness terms. There exists a k such that $B \wedge t \simeq f(w_k)$ is satisfiable.*

It is important that B is quantifier free. If B is quantified, the compactness lemma no longer holds. For example B could be $\forall x \;.\; f(x) \not\simeq 0$.

4.3.2 Finite Basis

The construction of the witness protection set from Lemma 5 contributes with arithmetic inequalities for projection. They are based on a finite set of terms t and a finite set of base projections of the original formula A. The projections are computed by using successively tighter bounds for variables. Projection incorporates these tighter bounds by producing witness terms with increasing offsets, but over a fixed basis of projections.

4.3.3 Essential witnesses

The point of generating distinct witness terms was to ensure that the interpretation of $f(w_{ij})$ is eventually unconstrained by B and can therefore be treated as a fresh variable. Fresh variables can be projected recursively, such that cores do not depend on $f(w_{ij})$. Cores are not automatically independent of $f(w_{ij})$.

Example 16. *Let* $A := a \simeq f(x) \simeq b$ *and* $B := a \not\simeq b$*. Each new projection in the sequence* $a \simeq f(w_i) \wedge f(w_i) \simeq b$*, with* $w_i := i$*, is unsatisfiable with* B*, while the negated cores* $\neg(a \simeq f(w_i) \wedge f(w_i) \simeq b)$ *remain satisfiable with* A*, resulting in an infinite projection (and core) sequence.*

Example 17. $A := 4 \uparrow (2f(a) + s)$ *and* $B := 4 \uparrow (s + 3)$*, where* a *is a private variable. Every instantiation* $4 \uparrow (2f(w_i) + s)$ *is contradictory with* B*. The consequence* $2 \uparrow s$ *of* A *already contradicts* B*.*

One way to deal with non-essential witnesses is to augment each projection that contains $f(w_{ij})$ with a projection where these occurrences are treated as fresh variables. If $f(w_{ij})$ is non-essential, the projection that does not contain $f(w_{ij})$ is used for the core. We therefore establish termination only for *essential* projection/*fair* core sequences, where

Definition 12 (Essential Projection/Fair Core Sequences)**.** *An* essential projection/fair core sequence A'_i *does not contain an infinite number of non-essential witness terms and does not infinitely avoid all minimal cores.*

4.3.4 Finite arithmetic projections

The witness protection set D from Definition 10 is constructed by selecting terms that occur under uninterpreted functions and used by a projection. If the witness terms $f(w_{ij})$ remain essential for an unbounded set of w_{ij}, we noted that occurrences of $f(w_{ij})$ could be treated as fresh variables and themselves be projected paving way for cores without $f(w_{ij})$. Without occurrences of $f(w_{ij})$ the witness protection set D does not contain disequalities $a_j \not\simeq w_{ij}$.

We have now detailed relevant criteria for termination and claim:

Corollary 2 (Total Correctness)**.** *If* $A \wedge B$ *is unsatisfiable, then every essential projection/fair core sequence terminates with a set of cores,* $core_1, \ldots, core_k$*, and* $I := core_1 \vee core_2 \vee \ldots \vee core_k$ *forms a reverse interpolant such that* $A \Rightarrow I$ *and* $B \Rightarrow \neg I$*.*

5 Related work

Interpolation is essential for most infinite-state model checking algorithms. It is used for predicate discovery in predicate abstraction, lemma generalization in IC3-style algorithms, and construction of bounded safety proofs in Interpolation-based Model Checking (IMC). Efficient interpolation generation has received ample attention from the research community, ranging from the basic problem of interpolation construction, to interpolation modulo additional structural constraints (e.g., quantifier-free, strength, allowed numerical coefficients, etc.)

Starting from the pioneering work of McMillan [27, 28] most interpolation algorithms are *proof-based*. An automated theorem prover is required to produce a refutation proof that is traversed to extract the interpolant. This is *feasible*, if an interpolant can be constructed in polynomial time in the size of the proof.

While theoretically attractive, proof-based interpolation comes with its own hurdles. First, it requires efficient proof logging. Second, proof calculi may not support feasible interpolation [29]. Thus, interpolation generation might be as complex as satisfiability checking. While interpolation is a widely requested feature, it is supported only by very few efficient SMT solvers: MathSAT5 [8], and SMTInterpol [7] support interpolation for *EUFLIA*, Princess supports interpolation for *LIA* and handles functions by translation into predicates [3], and OpenSMT [21] supports interpolation for *EUFLRA*.

Finally, a tight coupling between proofs, interpolation, and solving heuristics is a source of brittleness. Any change to the core solver might lead to proof calculi that are difficult to interpolate, breaking interpolation in potentially fundamental ways. A proof-based interpolation procedure [29] has been deprecated form Z3 for this reason. In contrast, we present a *proof-less* procedure that combines proof search and interpolation, uses only the conventional SMT solver interface, and only requires support for unsat cores.

Quantifier elimination provides an easy way to interpolate *LRA*. Thus, given two *LRA* formulas $A \wedge B$, an interpolant can be computed by finding an unsat core $\hat{A}$ of A and existentially eliminating all local symbols from $\hat{A}$. Proof-based procedures further use refutation proofs of $A \wedge B \vdash \bot$ to speed up quantifier elimination. *EUF* does not admit quantifier elimination, but admits uniform interpolation. A *uniform* interpolation procedure similar to *LRA*, uses so called *cover* instead of quantifier elimination [16]. *EUFLRA* is the most widely supported interpolating combined theory. Cimatti et al. [9] provide a great overview of this and other similar combinations in the context of SMT. The recent exposition [13] establishes that the combination of two convex theories have uniform interpolation and notes that the combination of EUF with integer difference logic does not have uniform interpolants similar to our example.

Interpolation for *LIA* is more complex. First, *LIA*, under its usual signature, does not admit quantifier elimination. One approach, e.g., [28, 29, 23], is to avoid this issue by allowing interpolants to be quantified. Another, is to extend the signature to include divisibility predicates as in [3] and in our paper. Unfortunately, this might lead to interpolants that are exponential in the size of the proof. Other extensions to the signature are possible. For example, [15] suggests adding *ceiling* function instead of divisibility predicates. Interpolation for *EUFLIA* is complicated even further since it does not admit quantifier elimination nor uniform interpolation, even in the signature extended with divisibility predicates. A detailed overview of a proof-based interpolation for *EUFLIA* is given in [6]. While many tools claim to support *EUFLIA* interpolation, the degree of that support is hard to evaluate. Interpolation for many other theories can be reduced to that of *EUFLIA* [23]. In fact, we hope to report on extending our procedure to extensional arrays in future work using a reduction approach and taking inspiration from [20, 14].

Interpolation is naturally reduced to finding solutions for satisfiability of recursion-free Constrained Horn Clauses (CHC) [17]. This is currently the suggested method of computing interpolants in Z3. Recursion-free (and recursive) CHCs are handled by Spacer [24] engine of Z3 that supports CHC over the combined theory of *LIA* and Arrays. Spacer constructs quantifier free interpolants for *LIA*, but quantified interpolants for the combination of *LIA* and Arrays [18]. This is sufficient to encode interpolation for *EUF*, but not for *EUFLIA*.

6 Summary

We presented a first fully model-based interpolation procedure for *EUFLIA*. In future work we hope to explore integrating the procedure with combinations of quantifiers and theories, noteworthy, arrays, algebraic data-types, non-linear arithmetic and bit-vectors. Model-based methods allow to dispense with proof objects, but a good integration requires an engine that finds sufficiently general models to avoid enumerating superfluous cubes.

References

[1] N. Bjørner and M. Janota. Playing with quantified satisfaction. In A. Fehnker, A. McIver, G. Sutcliffe, and A. Voronkov, editors, *20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning - Short Presentations, LPAR 2015, Suva, Fiji, November 24-28, 2015.*, volume 35 of *EPiC Series in Computing*, pages 15–27. EasyChair, 2015.

[2] M. P. Bonacina and M. Johansson. On interpolation in automated theorem proving. *J. Autom. Reasoning*, 54(1):69–97, 2015.

[3] A. Brillout, D. Kroening, P. Rümmer, and T. Wahl. An interpolating sequent calculus for quantifier-free presburger arithmetic. *J. Autom. Reasoning*, 47(4):341–367, 2011.

[4] R. Bruttomesso, S. Rollini, N. Sharygina, and A. Tsitovich. Flexible interpolation with local proof transformations. In L. Scheffer, J. R. Phillips, and A. J. Hu, editors, *2010 International Conference on Computer-Aided Design, ICCAD 2010, San Jose, CA, USA, November 7-11, 2010*, pages 770–777. IEEE, 2010.

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

[6] J. Christ. *Interpolation Modulo Theories.* PhD thesis, Albert-Ludwigs-Universität Freiburg, 2015.

[7] J. Christ, J. Hoenicke, and A. Nutz. Smtinterpol: An interpolating SMT solver. In A. F. Donaldson and D. Parker, editors, *Model Checking Software - 19th International Workshop, SPIN 2012, Oxford, UK, July 23-24, 2012. Proceedings*, volume 7385 of *Lecture Notes in Computer Science*, pages 248–254. Springer, 2012.

[8] A. Cimatti, A. Griggio, B. J. Schaafsma, and R. Sebastiani. The mathsat5 SMT solver. In N. Piterman and S. A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7795 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2013.

[9] A. Cimatti, A. Griggio, and R. Sebastiani. Efficient generation of craig interpolants in satisfiability modulo theories. *ACM Trans. Comput. Log.*, 12(1):7:1–7:54, 2010.

[10] L. M. de Moura and N. Bjørner. Model-based theory combination. *Electr. Notes Theor. Comput. Sci.*, 198(2):37–49, 2008.

[11] P. Downey, R. Sethi, and R. Tarjan. Variations on the common subexpression problem. *Journal of the ACM*, 27(4):758–771, 10 1980.

[12] A. Fuchs, A. Goel, J. Grundy, S. Krstic, and C. Tinelli. Ground interpolation for the theory of equality. In S. Kowalewski and A. Philippou, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings*, volume 5505 of *Lecture Notes in Computer Science*, pages 413–427. Springer, 2009.

[13] S. Ghilardi and A. Gianola. Interpolation and uniform interpolation in quantifier-free fragments of combined first-order theories. *Mathematics*, 10(3), 2022.

[14] S. Ghilardi, A. Gianola, and D. Kapur. Interpolation and amalgamation for arrays with maxdiff. In S. Kiefer and C. Tasson, editors, *Foundations of Software Science and Computation Structures - 24th International Conference, FOSSACS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 -*

April 1, 2021, Proceedings, volume 12650 of *Lecture Notes in Computer Science*, pages 268–288. Springer, 2021.

[15] A. Griggio, T. T. H. Le, and R. Sebastiani. Efficient interpolant generation in satisfiability modulo linear integer arithmetic. *Logical Methods in Computer Science*, 8(3), 2010.

[16] S. Gulwani and M. Musuvathi. Cover algorithms and their combination. In S. Drossopoulou, editor, *Programming Languages and Systems, 17th European Symposium on Programming, ESOP 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4960 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2008.

[17] A. Gupta, C. Popeea, and A. Rybalchenko. Solving recursion-free horn clauses over LI+UIF. In H. Yang, editor, *Programming Languages and Systems - 9th Asian Symposium, APLAS 2011, Kenting, Taiwan, December 5-7, 2011. Proceedings*, volume 7078 of *Lecture Notes in Computer Science*, pages 188–203. Springer, 2011.

[18] A. Gurfinkel, S. Shoham, and Y. Vizel. Quantifiers on demand. In S. K. Lahiri and C. Wang, editors, *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*, volume 11138 of *Lecture Notes in Computer Science*, pages 248–266. Springer, 2018.

[19] K. Hoder, L. Kovács, and A. Voronkov. Playing in the grey area of proofs. In J. Field and M. Hicks, editors, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 259–272. ACM, 2012.

[20] J. Hoenicke and T. Schindler. Efficient interpolation for the theory of arrays. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 549–565. Springer, 2018.

[21] A. E. J. Hyvärinen, M. Marescotti, L. Alt, and N. Sharygina. Opensmt2: An SMT solver for multi-core and cloud computing. In N. Creignou and D. L. Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 547–553. Springer, 2016.

[22] D. Jovanovic and C. Barrett. Sharing is caring: Combination of theories. In C. Tinelli and V. Sofronie-Stokkermans, editors, *Frontiers of Combining Systems, 8th International Symposium, FroCoS 2011, Saarbrücken, Germany, October 5-7, 2011. Proceedings*, volume 6989 of *Lecture Notes in Computer Science*, pages 195–210. Springer, 2011.

[23] D. Kapur, R. Majumdar, and C. G. Zarba. Interpolation for data structures. In M. Young and P. T. Devanbu, editors, *Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2006, Portland, Oregon, USA, November 5-11, 2006*, pages 105–116. ACM, 2006.

[24] A. Komuravelli, A. Gurfinkel, and S. Chaki. Smt-based model checking for recursive programs. In A. Biere and R. Bloem, editors, *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, volume 8559 of *Lecture Notes in Computer Science*, pages 17–34. Springer, 2014.

[25] A. Komuravelli, A. Gurfinkel, S. Chaki, and E. M. Clarke. Automatic Abstraction in SMT-Based Unbounded Software Model Checking. In *CAV*, pages 846–862, 2013.

[26] V. Kuncak, M. Mayer, R. Piskac, and P. Suter. Complete functional synthesis. In B. G. Zorn and A. Aiken, editors, *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2010, Toronto, Ontario, Canada, June 5-10, 2010*, pages 316–329. ACM, 2010.

[27] K. L. McMillan. Interpolation and sat-based model checking. In W. A. H. Jr. and F. Somenzi, editors, *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*, volume 2725 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2003.

[28] K. L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.

[29] K. L. McMillan. Interpolants from Z3 proofs. In P. Bjesse and A. Slobodová, editors, *International Conference on Formal Methods in Computer-Aided Design, FMCAD '11, Austin, TX, USA, October 30 - November 02, 2011*, pages 19–27. FMCAD Inc., 2011.

[30] K. L. McMillan. Interpolation and model checking. In E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, editors, *Handbook of Model Checking*, pages 421–446. Springer, 2018.

[31] T. Nipkow. Linear quantifier elimination. *J. Autom. Reasoning*, 45(2):189–212, 2010.

[32] W. Pugh. The omega test: a fast and practical integer programming algorithm for dependence analysis. In J. L. Martin, editor, *Proceedings Supercomputing '91, Albuquerque, NM, USA, November 18-22, 1991*, pages 4–13. ACM, 1991.

[33] A. Reynolds, T. King, and V. Kuncak. Solving quantified linear arithmetic by counterexample-guided instantiation. *Formal Methods in System Design*, 51(3):500–532, 2017.

A *EUF*

We prove the lemmas where $\mathcal{P}$ is obtained from a congruence closure of equalities E. This reduces the need to additionally refer to $\mathcal{P}$. A version of the proofs for arbitrary partitions is obtained by sprinking $\mathcal{P}$ in opportune places instead of E.

Of Lemma 1. We first establish by induction on the depth of t that each subterm of t has to be to congruent to some term in $\mathcal{T}$. The base case is where t is a 0-ary function symbol (representing a constant), and has no subterm. For the induction case, if $t \in \mathcal{T}$, the claim holds trivially (since $\mathcal{T}$ is closed under subterms). Otherwise, assume t is of the form $f(t_1, \dots, t_n)$ and that $E \models t \simeq t'$ for some term $t' \in \mathcal{T}$, i.e., t is congruent to some term in $\mathcal{T}$. Suppose some subterm t_i is not congruent to any term in $\mathcal{T}$. We can then create an interpretation of f that distinguishes $f(t_1, \dots, t_n)$ from every term in $\mathcal{T}$, while still satisfying E. This contradicts the assumption that $E \models t \simeq t'$ for some $t' \in \mathcal{T}$.

We now show that $\mathsf{rep}(t')$ is defined, again by induction on the depth of t. If $t \in \mathcal{T}$, the claim holds trivially. Otherwise, $t = f(t_1, \dots, t_n)$ where $f \in \Sigma_s$, and $t_i \in T_{\Sigma_s}$, and by Item i, $E \models t_i \simeq t'_i$ for some $t'_i \in \mathcal{T}$. Therefore, by the induction hypothesis $\mathsf{rep}(t'_i)$ is defined for every i, and hence so is $\mathsf{rep}(t')$. □ □

Lemma 8. *Let E be a set of equalities over Σ_{ps}, and let $E' := \mathsf{rep}(E)$. If $E \models t \simeq t'$ for $t \in T_{\Sigma_s}, t' \in \mathcal{T}$ then $E' \models t \simeq \mathsf{rep}(t')$.*

Note that by Lemma 1, $\mathsf{rep}(t')$ is necessarily defined for t' as in the lemma.

Of Lemma 8. Let E' be given as $\mathsf{rep}(E)$. We establish the lemma by induction on the depth of t. Assume the hypothesis of the lemma: $E \models t \simeq t'$. Let t be of the form $f(t_1, \dots, t_n)$, where $f \in \Sigma_s$ and $t_i \in T_{\Sigma_s}$, and assume, by Lemma 1(i), that for every t_i, $E \models t_i \simeq t'_i$ for some $t'_i \in \mathcal{T}$. By Lemma 1(ii), $\mathsf{rep}(t'_i)$ is defined. Hence, $E \models t_i \simeq \mathsf{rep}(t'_i)$ (since $E \models t'_i \simeq \mathsf{rep}(t'_i)$). Then the equivalence class for t' must contain a term congruent to $f(\mathsf{rep}(t'_1), \dots, \mathsf{rep}(t'_n))$. Otherwise, create an interpretation of f that distinguishes t from t', which would contradict our hypothesis. Thus, E' contains the equality $f(\mathsf{rep}(t'_1), \dots, \mathsf{rep}(t'_n)) \simeq \mathsf{rep}(t')$, and therefore the equality $f(t_1, \dots, t_n) \simeq \mathsf{rep}(t')$ can be derived from E'. □ □

Of Lemma 2. We construct M'' from M' by first ensuring M' and M'' agree on all terms in T_{Σ_s}. For equalities in $\mathcal{P}$ (recall that these are equalities over $\mathcal{T}$) that are not implied by $\mathsf{rep}^{=}_{M}(L)$, we update, if necessary, M' for symbols in Σ_p to enforce them, while still satisfying D.

This is done as follows. Let $\equiv_{M'}$ be the equivalence relation over T_{Σ_s} induced by M'. Let R be the minimal congruence relation on $T_{\Sigma_s} \cup \mathcal{T}$ such that $\mathcal{P} \cup \equiv_{M'} \subseteq R$ (we abuse notation and view $\mathcal{P}$ as a relation over terms). For the sake of the proof, we extend the definition of $\mathsf{rep}_{\mathcal{P}}(t)$ to terms in $T_{\Sigma_s} \setminus \mathcal{T}$, and define $\mathsf{rep}_{\mathcal{P}}(t)$ for such terms to be t. (Thus, the only terms for which $\mathsf{rep}_{\mathcal{P}}(t)$ may be undefined are terms in $\mathcal{T} \setminus T_{\Sigma_s}$.) We first claim that the only equalities in $R \setminus (\mathcal{P} \cup \equiv_{M'})$ are $t_1 \simeq t_2$ such that (i) $\mathsf{rep}_{\mathcal{P}}(t_1)$ and $\mathsf{rep}_{\mathcal{P}}(t_2)$ are both defined, and (ii) $(\mathsf{rep}_{\mathcal{P}}(t_1), \mathsf{rep}_{\mathcal{P}}(t_2))$ is already in $\equiv_{M'}$. To prove this, we recall that we can obtain R by an algorithm that merges classes based on transitivity (i.e., when they have a common term) and subsequently based on congruence. Applications of the transitivity rule on $\mathcal{P} \cup \equiv_{M'}$ preserve the claim by induction on the length of the derivation. The base case consists of applying transitivity directly on $\mathcal{P} \cup \equiv_{M'}$, where the proof relies on the property that if $(t, t') \in \mathcal{P}$ and $t \in T_{\Sigma_s}$ then $\mathsf{rep}_{\mathcal{P}}(t')$ is defined (Lemma 1). The induction step is trivial. Next, we consider equalities obtained by merging classes based on the congruence rule (starting from the classes derived by the transitivity rule, as above). We show that the congruence rule cannot merge any (new) classes. Assume to the contrary that congruence merges the classes of $t = f(t_1, \ldots, t_n)$ and $t = f(t'_1, \ldots, t'_n)$, and let these classes be the first to be merged by congruence. If at least one of t, t' is in T_{Σ_s}, then f is shared, and $(\mathsf{rep}_{\mathcal{P}}(t_i), \mathsf{rep}_{\mathcal{P}}(t'_i)) \in \equiv_{M'}$ for every i, hence the same congruence is already applicable w.r.t. $\equiv_{M,'}$. If t, t' are both in $\mathcal{T} \setminus T_{\Sigma_s}$, then they are conflicting terms in $\mathcal{P}$, and their witness arguments t_i, t'_i must have been merged by transitivity. However, this means that $\mathsf{rep}_{\mathcal{P}}(t_i)$ and $\mathsf{rep}_{\mathcal{P}}(t'_i)$ are defined, and $(\mathsf{rep}_{\mathcal{P}}(t_1), \mathsf{rep}_{\mathcal{P}}(t_2)) \in \equiv_{M'}$, in contradiction to the fact that M' is a model of $\mathsf{rep}^{\neq}_{M}(L)$, which includes the disequality $\mathsf{rep}_{\mathcal{P}}(t_1) \not\simeq \mathsf{rep}_{\mathcal{P}}(t_2)$.

The above claim implies that projecting R to T_{Σ_s} yields $\equiv_{M'}$, i.e., the only equalities that are added are over terms not in T_{Σ_s}. The interpretation of Σ_p in M'' is determined based on R. Since $\mathcal{P} \models E$ and $\mathcal{P} \subseteq R$, we have that $M'' \models E$. Finally, we can verify that all disequalities in D are satisfied. First, disequalities in $\mathsf{rep}^{\neq}_{M}(D)$ are not compromised, since R introduces no new equalities over T_{Σ_s}. Second, every disequality in $DCert_M(D)$, but not in $\mathsf{rep}^{\neq}_{M}(D)$, will not be in R since R only adds equalities between terms that have representatives. □ □

Of Theorem 1. We are given L a conjunction of equalities and disequalities over Σ_{ps}. From definition 6 construct a UF model projection $L_i := \mathsf{rep}(E) \wedge \mathsf{rep}_{M_i}(D)$, given a model M_i such that $M_i \models L$. There are only finitely many possible $\mathsf{rep}_{M_i}(D)$ because the set of terms $\mathcal{T}$ over L is finite. We can now check that the resulting formulas $L_i := \mathsf{rep}(E) \wedge \mathsf{rep}_{M_i}(D)$ satisfy the conditions of the lemma. The first condition follows directly because each model of L is responsible for constructing some L_i that is satisfied by that model. This also automatically establishes that $\bigvee_i L_i \models \varphi$ implies $L \models \varphi$. So assume that $L \models \varphi$. We want to establish that $L_i \models \varphi$ for every M_i. That is, $\mathsf{rep}(E) \wedge \mathsf{rep}_{M_i}(D) \models \varphi$ for every model $M_i \models L$. Lemma 2 established that a model M' of $\mathsf{rep}(E) \wedge \mathsf{rep}_{M_i}(D) \wedge \neg\varphi$ can be adapted to a model M'' of $E \wedge D$ that agrees with M' on all symbols in $\neg\varphi$ (since φ is quantifier-free). This contradicts the assumption that $L \models \varphi$. □ □

B Arithmetic

of Theorem 2. We prove this property constructively by introducing *model-based witness* extraction, Mbw, for a set of literals satisfied by a given model M. The result of Mbw from a model M is a witness term $t[\vec{y}]$ and another set of literals that is satisfied by M and implies $Ar[t[\vec{y}], \vec{y}]$. The definition of Mbw, provided in Figure 1, partitions models of Ar into a finite set of equivalence classes corresponding to generated projections. This ensures that the disjunction of generated projections is equivalent to $\exists x \, . \, Ar[x, \vec{y}]$.

$$\begin{array}{rcl}
Mbw(M, x, \top) & = & \langle 0, \top \rangle \\
Mbw(M, x, ax \not\simeq t \wedge L) & = & Mbw(M, x, t+1 \leq ax \wedge L) \text{ if } \; M(ax) > M(t) \\
Mbw(M, x, ax \not\simeq t \wedge L) & = & Mbw(M, x, ax \leq t-1 \wedge L) \text{ if } \; M(ax) < M(t) \\
Mbw(M, x, \bigwedge_i t_i \leq a_i x) & = & \langle (t_0 + a_0 - 1) \text{ div } a_0, \bigwedge_i a_i t_0 \geq a_0 t_i \rangle \\
 & & \text{if } M(a_i t_0) \geq M(a_0 t_i), \forall i \\
Mbw(M, x, \bigwedge_j b_j x \leq s_j) & = & \langle s_0 \text{ div } b_0, \bigwedge_j b_j s_0 \leq b_0 s_j \rangle \\
 & & \text{if } M(b_j s_0) \geq M(b_0 s_j), \forall j \\
Mbw\left(M, x, \begin{array}{ll} \bigwedge_i t_i \leq a_i x & \wedge \\ \bigwedge_j b_j x \leq s_j & \end{array}\right) & = & \left\langle \begin{array}{l} (t_0 + a_0 - 1) \text{ div } a_0, \\ \qquad \bigwedge_i t_i a_0 \leq t_0 a_i \\ \wedge \quad \bigwedge_j resolve(M, t_0 \leq a_0 x, b_j x \leq s_j) \end{array} \right\rangle \\
 & & \text{if} M(t_0/a_0) \geq M(t_i/a_i), \forall i. \\
Mbw(M, x, \bigwedge_{i=1}^{n} d_i \mid (a_i x + t_i) \wedge L) & = & \langle dp + u, L' \wedge \bigwedge_{i=1}^{n} d_i \mid (a_i u + t_i) \rangle \\
 & & \text{where} \\
\langle p, L' \rangle & = & Mbw(M, x', L[u + d \cdot x'/x]), \\
 & & d = \text{lcm}(d_1, \ldots, d_n), u = M(x) \mod d
\end{array}$$

Figure 1: Rules for Model-based witness extraction.

The predicate $resolve(M, t \leq ax, bx \leq s)$ is defined by the cases:

$$\begin{array}{lll}
bt + (a-1)(b-1) \leq as & \textit{if} & (a-1)(b-1) \leq M(bt - as) \\
bt \leq as \wedge a | (t+d) \wedge b(t+d) \leq as) & \textit{if} & b \geq a, d := M(-t) \mod a \\
bt \leq as \wedge b | (s-d) \wedge bt \leq a(s-d) & \textit{ow} & (a > b, d := M(s) \mod b)
\end{array}$$

Intuitively, *resolve* ensures that the lower and upper bounds for x are sufficiently separated. The corresponding resolution rule for linear real arithmetic is of course much simpler: $resolve(M, t \leq x, x \leq s) = t \leq s$ when x is a variable of sort Real. We do not consider the case of mixed integer real linear arithmetic in this paper. For integer arithmetic we distinguish the cases where s and t are far apart and the case where they are close and x has to align with the values allowed by the coefficients a, b. $\square$

Model-based witness extraction extends to a set of variables, e.g., let $\langle w_x, L' \rangle := Mbw(M, x, L)$, $\langle w_y, L'' \rangle := Mbw(M, y, L')$, and $w'_x := w_x[y \mapsto w_y]$, then $Mbw(M, xy, L) := \langle w'_x w_y y, L'' \rangle$. It can be applied to formulas in negation normal form by setting L to be a subset of literals that are true in M and imply the negation normal form formula.

C Combination

of Lemma 4 - idea. A model of the projection is adapted to a model of the original formula by setting the interpretation of eliminated symbols $\vec{a}$ to the values of the corresponding witness terms $\vec{w}$. It is adapted also to an interpretation of private uninterpreted symbols that satisfies the same partition as the model M used to compute the projection (as in Lemma 2). To establish that M' can be adapted to satisfy the original formula note that EUF completion ensures that $\vec{w}$ preserves equalities and disequalities required for satisfying F', and that Ar' is not under-constrained with respect to F. Conversely, F' is not under-constrained with respect to Ar since it is computed based on the partition induced by M and the witness terms $\vec{w}$. Thus, it is consistent with all equalities and disequalities enforced by Ar' and therefore also Ar. □

of Lemma 5 - idea. By the way projection formulas are created from a model M, a model of the projection $Ar' \wedge F'$ evaluates subterms over Σ_s in $\neg core_i[\vec{w}_i]$ according to M. The non-shared arguments to functions satisfy the same disequalities as M. Note that $core_i$ was obtained from literals that occur in projections of A and $core_j$, for $j < i$. In particilar, since D is implied by the projection it satisfies the disequalities from D. An uninterpreted function $f(a_j)$ that occurs in $core_i[\vec{a}]$ with argument $a_j \in \vec{a}$ is then given an interpretation for $f(w_{ij})$. The argument w_{ij} is distinct from variables in $\vec{a}$ whenever M requires it and therefore the interpretation of f at $f(w_{ij})$ is unconstrained by $\vec{a}$ and can be adjusted to follow how M satisfied each $\neg core_i[\vec{w}_i]$. The same argument applies to nested occurrences of uninterpreted functions, such as $f(g(a_j))$, and to functions of arity greater than one. □

of Lemma 6. Assume we are given an infinite projection/core sequence $A'_i/core_i$. A model $M_i \models A_i$, must satisfy the negation of at least one literal in each core. This is either the negation of a literal from Ar'_i or the negation of a literal from F' or an equality $a_j \simeq w_{ij}$. The literals from Ar'_i can only be blocked finitely because they are either derived from atoms in Ar or they are obtained from blocking comparison between lower and upper bounds to an eliminated symbol a. An infinite sequence therefore has to contain blocked term partitions. There is a finite set of partitions over the original set of terms so an infinite sequence must involve witness terms. It therefore follows from Lemma 5 and Definition 11 that for an infinite sequence of projections there is an a_j with an unbounded number of different w_{ij}. The sequence cannot converge on a fixed set of witness terms because each core $core_i$ blocks the same projection at later stages. Theorem 2 ensures that the set of witness terms are based on a finite set of bounds in Ar and differ only by a constant offset. So among an infinite set of syntactically different witness terms w_{ij} there is some fixed term t based on Ar and coefficients $\vec{b}_i$ and $\vec{a}_i$ (not to be confused with the private arithmetic constants), such that w_{ij} is formed using the grammar (3.1) with t as the base case and $\vec{b}_i$ for the multipliers and divisors and $\vec{a}_i$ for the offsets. It follows that the set $\{\vec{a}_i\}$ is infinite (the set $\{\vec{b}_i\}$ came from a finite set of least common multiples, so is finite). Therefore w_{ij} contains an infinite set of distinct terms.

□

of Lemma 7. We show a stronger claim. Let α_k be defined as $f(w_k) \simeq t \wedge \bigwedge_{i<k} f(w_i) \not\simeq t$. Then there is a $k \geq 1$, such that every model $M \models B$ can be modified to a model $M' \models B \wedge \alpha_k$ by only changing the interpretation of f on $M(w_k)$.

Every model $M \models B$ determines the graph of f relevant to satisfiability of B. The size of this graph can be bounded by the number of subterms of B, which we denote $|B|$, because the evaluation of B only depends on how M behaves on the fixed set of ground terms in B. The graph of f outside of these values can be changed at will without affecting $M \models B$. Therefore, if the graph of f cannot be adjusted to satisfy $f(w_k) \simeq t$, then it is because all models for B determine the graph of f on w_k. However, each model of B determines the graph of f on at most $|B|$ terms. Since all w_i are distinct, there has to be some $k \leq |B| + 1$, such that when $M \models B$, the graph can be adjusted such that $B \wedge \alpha_k$ is satisfied by M' that differs at most from M by the interpretation on w_k. □

D An Implementation

An implementation of our algorithm is available with z3 on `https://github.com/z3prover/z3`. It computes the advertised interpolants over *EUFLIA* on instances collected from the literature and SMTInterpol regressions. It works directly on arbitrary formulas and uses dual propagation to extract small implicants. In contrast to the presentation, our implementation does not use a witness protection transformation, but instead relies directly on the current cubes for the next interpolant. We conjecture that this approach is equivalent to applying witness protection. It does not ensure essential witnesses. The implementaiton uses model-based projection facility for *LIA*, already used by the SPACER solver. In contrast to SPACER, it does not (yet) leverage proof traces and Farkas lemma to weaken cores. A thorough evaluation is beyond the scope of this paper: the value of interpolation procedures is best evaluated in context of their use, which has dominantly been in symbolic model checking.

There is currently no SMTLIB2 standard for interpolation, and our format relies on adding a command, **get-interpolant**. In contrast to other formats, ours does not rely on enabling proof producing traces. An example, illustrating the input format is provided below.

```
(declare-const s Int)   (declare-const t Int)
(declare-const a Int)   (declare-const b Int)
(declare-const q Int)   (declare-fun f (Int) Int)
(get-interpolant
   (and (<= s (* 2 a)) (<= (* 2 a) t) (= (f a) q))
   (and (<= t (* 2 b)) (<= (* 2 b) (+ s 1)) (not (= (f b)))))
```

Our sample experiments with the *EUFLIA* interpolation implementaiton bears witness to how the number of cubes produced during interpolation is highly sensitive to how models enumerate partitions. Z3 uses a congruence closure algorithm for pure *EUF*, so the partitions are optimal in the sense of coarseness for conjunctions of *EUF* formulas. For arithmetical terms, on the other hand, the procedure has to rely on the partition imposed by a current model. The more equalities and disequalities that are used by projection, the stronger the projection becomes and the more cubes are enumerated for the intepolant. Possible optimizations for for reducing the number of cubes includes *freedom* intervals [10] used by Z3's model-based theory combination approach. Dejan Jovanovich and Clark Barrett [22] developed several conditions for tempering equality sharing for theory combinations, and ideas related to these techniques may well be very suitable for tuning a model-based interpolation procedure.