Decomposing Farkas Interpolants

Martin Blicha[1,2(✉)], Antti E. J. Hyvärinen[1], Jan Kofroň[2], and Natasha Sharygina[1]

[1] Università della Svizzera italiana (USI), Lugano, Switzerland
{martin.blicha,antti.hyvaerinen,natasha.sharygina}@usi.ch
[2] Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
{martin.blicha,jan.kofron}@d3s.mff.cuni.cz

Abstract. Modern verification commonly models software with Boolean logic and a system of linear inequalities over reals and over-approximates the reachable states of the model with Craig interpolation to obtain, for example, candidates for inductive invariants. Interpolants for the linear system can be efficiently constructed from a Simplex refutation by applying the Farkas' lemma. However, Farkas interpolants do not always suit the verification task and in the worst case they may even be the cause of divergence of the verification algorithm. This work introduces the decomposed interpolants, a fundamental extension of the Farkas interpolants obtained by identifying and separating independent components from the interpolant structure using methods from linear algebra. We integrate our approach to the model checker Sally and show experimentally that a portfolio of decomposed interpolants results in immediate convergence on instances where state-of-the-art approaches diverge. Being based on the efficient Simplex method, the approach is very competitive also outside these diverging cases.

Keywords: Model checking · Satisfiability modulo theory · Linear arithmetic · Craig interpolation

1 Introduction

A central task in model checking systems with respect to safety properties [27] consists of proving facts and attempting to generalize the obtained proofs. The generalizations serve as a basis for inductive invariants needed for guiding the search for a correctness proof in approaches such as IC3 [8] and k-induction [39], both known to scale to the verification of highly complex systems.

Finding good proofs and generalizing them is hard. A widely used approach, Satisfiability Modulo Theories (SMT) [7,13], models a system with propositional logic and a range of first-order logics. Solvers for SMT combine a resolution-based variant of the DPLL-algorithm [11,12,40] for propositional logic with decision procedures for first-order logics. A vast range of first-order logics is maintained as part of the SMT-LIB Initiative [6]. What is common to these logics is that their solving requires typically only a handful of algorithms. Arguably, the two most important algorithms are a congruence closure algorithm for deciding quantifier-free equality logic with uninterpreted functions [31], and a Simplex-based procedure for linear arithmetic over real or rational numbers [16].

Generalizing proofs to inductive invariants is commonly done by Craig interpolation [10]. Here, the model is split into two parts, say, A and B, resulting in an *interpolation problem* (A, B). The proof of unsatisfiability for $A \wedge B$ is used to extract an *interpolant* I, a formula that is defined over the common symbols of A and B, is implied by A, and is unsatisfiable with B. Several interpolants can be computed for a given interpolation problem, and not all of them are useful for proving safety. Typically, this is a phenomenon used to construct a *portfolio* [20] of interpolation algorithms that is then applied in the hopes of aiding to find the safety proof.

The approaches to interpolation based on Farkas' lemma construct an LRA interpolant by summing all inequalities appearing in A into a single inequality. We call the resulting interpolant the *Farkas interpolant*. While a single inequality is desirable in some cases, it prevents IC3-style algorithms from converging in other ones [36]. We present how methods from linear algebra can be applied on a Farkas interpolant to obtain *decomposed interpolants* that do not consist of a single inequality and guarantee the convergence of the model-checking algorithm for some of the cases where Farkas interpolants do not converge. A major advantage of decomposed interpolants is that they can be computed using Simplex-based decision procedures as a black box, allowing us to make use of the highly tuned implementations present in many state-of-the-art SMT solvers.

Intuitively, while computing the decomposed interpolants we do not directly sum the inequalities in A, but, instead, we split the sum into sub-sums. The result is an interpolant that is a conjunction of often more than one component of the Farkas interpolant. This allows us not only to solve the convergence problem observed in model checking examples, but also to gain more control over the strength of LRA interpolants. In summary, the main contributions of this paper are

1. a new Farkas-lemma-based interpolation algorithm for LRA that is able to deal with convergence problems in model-checking benchmarks while still relying on a highly efficient Simplex-based decision procedure,
2. establishing properties regarding logical strength of interpolants produced by our interpolation algorithm with respect to the original Farkas interpolants,
3. implementation of our new interpolation algorithm in OpenSMT, our SMT solver, and integration of our approach with the model checker sally
4. experiments showing that the new approach is efficient in model checking, in particular in showing systems unsafe.

While the underlying intuition is simple, we quote here Jean D'Alembert (1717–1783) in saying that *Algebra is generous; she often gives more than is asked of her*: Our detailed analysis in Sects. 4 and 5 shows that the structure of the problem is surprisingly rich. Our experiments in Sect. 6 verify that the phenomena are practically relevant. Overall a portfolio constructed from our interpolation algorithm is significantly better than a portfolio based purely on Farkas interpolants. We furthermore show for individual instances that the effect is consistent instead of arising from random effects.

Related Work. The work on interpolation in LRA dates back to [32]. A compact set of rules for deriving LRA interpolants from the proof of unsatisfiability in an inference system was presented in [29]. The interpolants in these works were the Farkas interpolants. Current methods usually compute Farkas interpolants from explanations of unsatisfiability extracted directly from the Simplex-based decision procedure inside the SMT solver [16]. Recently in [3], we presented a way of computing an infinite family of interpolants between a primal and a dual interpolant with variable strength. However, those interpolants are still restricted to single inequalities.

The work most closely related to ours is [36] where the authors independently recognized the weakness of interpolation based on Farkas coefficients. They introduce a new interpolation procedure that gives guarantees of convergence of a special sequence of interpolation problems often occurring in model checking problems. However, this interpolation algorithm is based on a different decision procedure, called conflict resolution [26], which, based on the results reported in [36], is not as efficient as the Simplex-based decision procedure. In contrast, we show how the original approach based on the Simplex-based decision procedure and Farkas coefficients can be modified to produce interpolants not restricted to the single-inequality form, while additionally obtaining strength guarantees with respect to the original Farkas interpolants.

Other work on LRA interpolants include e.g. [1,35,37]. Both [1] and [37] focus on producing simple overall interpolants by attempting to reuse (partial) interpolants from pure LRA conflicts. Our focus is not on the overall interpolant, but on a single LRA conflict. However, in the context of interpolants from proofs produced by SMT solvers, our approach also has a potential for re-using components of interpolants for LRA conflicts across the whole proof. Beside algorithms for interpolants for LRA conflicts, there exist a large body of work on propositional interpolation [2,14,19,23].

The structure of the paper is as follows. In Sect. 2 we provide a concrete example model-checking problem where our approach guarantees immediate convergence but Farkas interpolation diverges. In Sect. 3 we define the notation used in the paper, and in Sects. 4 and 5 detail our main theoretical contribution. We provide experimental results in Sect. 6, and finally conclude in Sect. 7.

2 Motivation

Consider the transition system $S = (I, T, Err)$, where I and Err are, respectively, predicates that capture the initial and error states, and T is the transition function. The symbols x, y are real variables, and x', y' are their next-state versions.[1]

$$S = \begin{cases} I \equiv (x = 0) \wedge (y = 0), \\ T \equiv (x' = x + y) \wedge (y' = y + 1), \\ Err \equiv (x < 0) \end{cases} \tag{1}$$

[1] This example was first brought to our attention by Prof. Arie Gurfinkel. A similar example appears in [36].

The system is one variant from a family of similar transition systems that are known to not converge in straightforward implementations of IC3-based algorithms using LRA interpolation. For example, both SPACER [25] (using interpolation algorithm of Z3 [30]) and SALLY [24] (using interpolation algorithm of MATHSAT [9]) fail to compute a safe inductive invariant for this transition system. However, SALLY with our interpolation algorithm succeeds in computing the safe inductive invariant.[2] Closer examination of SALLY and SPACER reveals that the tools in their default configurations produce a divergent series of candidate invariants of the form $0 \leq kx + y$ for $k = 1, 2, 3, \ldots$. The reason for producing such a series is that both tools rely on Farkas interpolants that always consist of a single inequality. Instead of generalizing the Farkas interpolants, an approach advocated in this work, interpolation based on a different decision procedure was proposed for SALLY in [36], whereas SEAHORN [18] with SPACER as its underlying reasoning engine solves this issue with abstract interpretation.

In this work we show how to modify the interpolation algorithm to produce in the general case a *conjunction* of multiple inequalities, leading, in this case, to the discovery of an inductive safe invariant $x \geq 0 \wedge y \geq 0$. To avoid here a lengthy discussion on internals of IC3 but nevertheless provide a concrete example of the power of decomposed interpolants, we apply decomposed interpolants in a simple, interpolation-based procedure for computing inductive invariants for transition systems. This approach is a simplified version of k-induction (see, e.g., [28]). When applied to the system in Eq. (1), we show that computing the Farkas interpolant fails and decomposed interpolant succeeds in producing a safe inductive invariant. A safe, inductive invariant for (I, T, Err) is a predicate R that satisfies (1) $I(X) \rightarrow R(X)$, (2) $R(X) \wedge T(X, X') \rightarrow R(X)$, and (3) $R(X) \wedge Err(X) \rightarrow \bot$. We may opportunistically try to synthesise R by interpolating over the interpolation problem $(I(X), T(X, X') \wedge Err(X'))$. Using the system S of Eq. (1), we obtain $(x \geq 0 \wedge y \geq 0, x' = x + y \wedge y' = y + 1 \wedge x' < 0)$. A Farkas interpolant, the sum of the components from the A-part, is $x + y \geq 0$, which is neither safe nor inductive for S. However, the *decomposed interpolant* $x \geq 0 \wedge y \geq 0$ is an inductive invariant.

3 Preliminaries

We work in the domain of *Satisfiability Modulo Theories* (SMT) [7,13], where satisfiability of formulas is determined with respect to some background theory. In particular, we are concerned with the *lazy* approach to SMT, that combines SAT solver dealing with the propositional structure of a formula and *theory* solver for checking consistency of a conjunction of theory literals. The proof of unsatisfiability in this approach is basically a propositional proof that incorporates *theory lemmas* learnt by the theory solver and propagated to the SAT solver.

[2] Current implementation of SPACER does not support conjunctions of inequalities as interpolants, and therefore we are at the moment unable to try our approach on SPACER.

The proof-based interpolation algorithm then combines any propositional-proof-based interpolation algorithm with *theory interpolator*. Theory interpolator provides an interpolant for each theory conflict—an unsatisfiable conjunction of theory literals.

Linear Arithmetic and Linear Algebra. We use the letters x, y, z to denote variables and c, k to denote constants. Vector of n variables is denoted by $\mathbf{x} = (x_1, \ldots, x_n)^\intercal$ where n is usually known from context. $\mathbf{x}[i]$ denotes the element of $\mathbf{x}$ at position i, i.e. $\mathbf{x}[i] = x_i$. The vector of all zeroes is denoted as $\mathbf{0}$ and $\mathbf{e_i}$ denotes the unit vector with $\mathbf{e_i}[i] = 1$ and $\mathbf{e_i}[j] = 0$ for $j \neq i$. For two vectors $\mathbf{x} = (x_1, \ldots, x_n)^\intercal$ and $\mathbf{y} = (y_1, \ldots, y_n)^\intercal$ we say that $\mathbf{x} \leq \mathbf{y}$ iff $x_i \leq y_i$ for each $i \in \{1, \ldots, n\}$. $\mathbb{Q}$ denotes the set of rational numbers, $\mathbb{Q}^n$ the n-dimensional vector space of rational numbers and $\mathbb{Q}^{m \times n}$ the set of rational matrices with m rows and n columns. A transpose of matrix M is denoted as $M^\intercal$. A kernel (also nullspace) of a matrix M is the vector space $ker(M) = \{\mathbf{x} \mid M\mathbf{x} = \mathbf{0}\}$.

We adopt the notation of matrix product for linear arithmetic. For a linear term $l = c_1x_1 + \cdots + c_nx_n$, we write $\mathbf{c}^\intercal\mathbf{x}$ to denote l. Without loss of generality we assume that all linear inequalities are of the form $\mathbf{c}^\intercal\mathbf{x} \bowtie c$ with $\bowtie \in \{\leq, <\}$. By linear system over variables $\mathbf{x}$ we mean a finite set of linear inequalities $S = \{C_i \mid i = 1, \ldots, m\}$, where each C_i is a linear inequality over $\mathbf{x}$. Note that from the logical perspective, each C_i is an atom in the language of the theory of linear arithmetic, thus system S can be expressed as a formula $\bigwedge_{i=1}^{m} C_i$ and we use these representations interchangeably. A linear system is satisfiable if there exists an evaluation of variables that satisfies all inequalities; otherwise, it is unsatisfiable. This is the same as the (un)satisfiability of the formula representing the system.

We extend the matrix notation also to the whole linear system. For the sake of simplicity we use $\leq$ instead of $\bowtie$, even if the system contains a mix of strict and non-strict inequalities. The only important difference is that a (weighted) sum of a linear system (as defined below) results in a strict inequality, instead of a non-strict one, when at least one strict inequality is present in the sum with a non-zero coefficient. The theory, proofs and algorithm remain valid also in the presence of strict inequalities. We write $C\mathbf{x} \leq \mathbf{c}$ to denote the linear system S where C denotes the matrix of all coefficients of the system, $\mathbf{x}$ are the variables and $\mathbf{c}$ is the vector of the right sides of the inequalities. With the matrix notation, we can easily express the sum of (multiples) of inequalities. Given a system of inequalities $C\mathbf{x} \leq \mathbf{c}$ and a vector of "weights" (multiples) of the inequalities $\mathbf{k} \geq \mathbf{0}$, the inequality that is the (weighted) sum of the system can be expressed as $\mathbf{k}^\intercal C\mathbf{x} \leq \mathbf{k}^\intercal\mathbf{c}$.

Craig Interpolation. Given two formulas $A(\mathbf{x}, \mathbf{y})$ and $B(\mathbf{y}, \mathbf{z})$ such that $A \wedge B$ is unsatisfiable, a *Craig interpolant* [10] is a formula $I(\mathbf{y})$ such that $A \implies I$ and $I \implies \neg B$.

The pair of formulas (A, B) is also referred to as an *interpolation problem*. In linear arithmetic, the interpolation problem is a linear system S partitioned into two parts: A and B.

One way to compute a solution to an interpolation problem in linear arithmetic, used in many modern SMT solvers, is based on Farkas' lemma [17,38]. Farkas' lemma states that for an unsatisfiable system of linear inequalities $S \equiv C\mathbf{x} \leq \mathbf{c}$ there exist *Farkas* coefficients $\mathbf{k} \geq \mathbf{0}$ such that $\mathbf{k}^\intercal C\mathbf{x} \leq \mathbf{k}^\intercal \mathbf{c} \equiv 0 \leq -1$. In other words, the weighted sum of the system given by the Farkas coefficients is a contradictory inequality. If a strict inequality is part of the sum, the result might also be $0 < 0$.

The idea behind the interpolation algorithm based on Farkas coefficients is simple. Intuitively, given the partition of the linear system into A and B, we compute only the weighted sum of A. It is not hard to see that this sum is an interpolant. It follows from A because a weighted sum of a linear system with non-negative weights is always implied by the system. It is inconsistent with B because its sum with the weighted sum of B (using Farkas coefficients) is a contradictory inequality by Farkas lemma. Finally, it cannot contain any A-local variables, because in the weighted sum of the whole system all variables are eliminated, A-local variables are not present in B, so they must be eliminated already in the weighted sum of A.

More formally, for an unsatisfiable linear system $S \equiv C\mathbf{x} \leq \mathbf{c}$ over n variables, where $C \in \mathbb{Q}^{m\times n}, \mathbf{c} \in \mathbb{Q}^m$, and its partition to $A \equiv C_A\mathbf{x} \leq \mathbf{c_A}$ and $B \equiv C_B\mathbf{x} \leq \mathbf{c_B}$, where $C_A \in \mathbb{Q}^{k\times n}$, $C_B \in \mathbb{Q}^{l\times n}$, $\mathbf{c_A} \in \mathbb{Q}^k$, $\mathbf{c_B} \in \mathbb{Q}^l$ and $k + l = m$, there exist Farkas coefficients $\mathbf{k}^\intercal = (\mathbf{k}_\mathbf{A}^\intercal \; \mathbf{k}_\mathbf{B}^\intercal)$ such that

$$(\mathbf{k}_\mathbf{A}^\intercal \; \mathbf{k}_\mathbf{B}^\intercal) \begin{pmatrix} C_A \\ C_B \end{pmatrix} = 0, (\mathbf{k}_\mathbf{A}^\intercal \; \mathbf{k}_\mathbf{B}^\intercal) \begin{pmatrix} \mathbf{c_A} \\ \mathbf{c_B} \end{pmatrix} = -1,$$

and the *Farkas interpolant* for (A, B) is the inequality

$$I^f \equiv \mathbf{k}_\mathbf{A}^\intercal C_A \mathbf{x} \leq \mathbf{k}_\mathbf{A}^\intercal \mathbf{c_A} \tag{2}$$

4 Decomposed Interpolants

In this section, we present our new approach to computing interpolants in linear arithmetic based on Farkas coefficients. The definition of Farkas interpolant of Eq. (2) corresponds to the weighted sum of A-part of the unsatisfiable linear system. This sum can be decomposed into j sums by decomposing the vector $\mathbf{k_A}$ into j vectors

$$\mathbf{k_A} = \sum_{i=1}^{j} \mathbf{k_{A,i}} \tag{3}$$

such that $\mathbf{0} \leq \mathbf{k_{A,i}} \leq \mathbf{k_A}$ for all i, thus obtaining j inequalities

$$I_i \equiv \mathbf{k}_\mathbf{A,i}^\intercal C_A \mathbf{x} \leq \mathbf{k}_\mathbf{A,i}^\intercal \mathbf{c_A} \tag{4}$$

If $\mathbf{k_{A,i}}$ are such that the left-hand side of the inequalities I_i contains only shared variables, the decomposition has an interesting application in interpolation, as illustrated below.

Definition 1 (decomposed interpolants). *Given an interpolation instance* (A, B)*, if there exists a sum of the form Eq.* (3) *such that the left side of Eq.* (4) *contains only shared variables for all* $1 \leq i \leq j$*, then the set of inequalities* $S = \{I_1, \ldots, I_j\}$ *is a* decomposition. *In that case the formula* $\bigwedge_{i=1}^{j} I_i$ *is a* decomposed interpolant *(DI) of size* j *for* (A, B)*.*

The decomposed interpolants are proper interpolants, as stated in the following theorem.

Theorem 1. *Let* (A, B) *be an interpolation problem in linear arithmetic. If* $S = \{I_1, \ldots, I_k\}$ *is a decomposition, then* $I^{DI} = I_1 \wedge \ldots \wedge I_k$ *is an interpolant for* (A, B)*.*

Proof. Let $I^{DI} = I_1 \wedge \ldots \wedge I_k$. First, $A \implies I^{DI}$ holds since for all I_i, $A \implies I_i$. This is immediate from the fact that A is a system of linear inequalities $C_A\mathbf{x} \leq \mathbf{c_A}$, $I_i \equiv \mathbf{k}^\intercal_{\mathbf{A},\mathbf{i}} C_A \mathbf{x} \leq \mathbf{k}^\intercal_{\mathbf{A},\mathbf{i}} \mathbf{c_A}$ and $\mathbf{0} \leq \mathbf{k_{A,i}}$. Second, $I^{DI} \wedge B \implies \bot$ since I^{DI} implies Farkas interpolant I^f. This holds because $\mathbf{k_A} = \sum_i \mathbf{k_{A,i}}$ and $\mathbf{0} \leq \mathbf{k_{A,i}}$. Third, I^{DI} contains only shared variables by the definition of decomposition (Definition 1). Therefore, I^{DI} is an interpolant. □

Each interpolation instance has a *DI* of size one, a *trivial* decomposition, corresponding to the Farkas interpolant of Eq. (2). However, interpolation problems in general can admit bigger decompositions. In the following we give a concrete example of an instance with decomposition of size two.

Example 1. Let (A, B) be an interpolation problem in linear arithmetic with $A = (x_1 + x_2 \leq 0) \wedge (x_1 + x_3 \leq 0) \wedge (-x_1 \leq 0)$ and $B = (-x_2 - x_3 \leq -1)$. The linear systems corresponding to A and B are

$$C_A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ -1 & 0 & 0 \end{pmatrix}, \quad \mathbf{c_A} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad \text{and} \quad C_B = \begin{pmatrix} 0 & -1 & -1 \end{pmatrix}, \quad \mathbf{c_B} = \begin{pmatrix} -1 \end{pmatrix}.$$

Farkas coefficients are

$$\mathbf{k}^\intercal_{\mathbf{A}} = \begin{pmatrix} 1 & 1 & 2 \end{pmatrix} \text{ and } \mathbf{k}^\intercal_{\mathbf{B}} = \begin{pmatrix} 1 \end{pmatrix},$$

while Farkas interpolant for (A, B) is the inequality $I^f \equiv x_2 + x_3 \leq 0$. However, if we decompose $\mathbf{k_A}$ into

$$\mathbf{k}^\intercal_{\mathbf{A},\mathbf{1}} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \text{ and } \mathbf{k}^\intercal_{\mathbf{A},\mathbf{2}} = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix},$$

we obtain the decomposition $\{x_2 \leq 0, x_3 \leq 0\}$ corresponding to the decomposed interpolant $I^{DI} \equiv x_2 \leq 0 \wedge x_3 \leq 0$ of size two.

4.1 Strength-Based Ordering of Decompositions

Decomposition of Farkas coefficients for a single interpolation problem is in general not unique. However, we can provide some structure to the space of possible interpolants by ordering interpolants with respect to their logical strength. To achieve this, we define the *coarseness* of a decomposition based on its ability to partition the terms of the interpolant into finer sums, and then prove that coarseness provides us with a way of measuring the interpolant strength.

Definition 2. *Let D_1, D_2 denote two decompositions of the same interpolation problem of size m, n, respectively, where $n < m$. Let $(\mathbf{q_1}, \dots, \mathbf{q_m})$ denote the decomposition of Farkas coefficients corresponding to D_1 and let $(\mathbf{r_1}, \dots, \mathbf{r_n})$ denote the decomposition of Farkas coefficients corresponding to D_2. We say that decomposition D_1 is* finer *than D_2 (or equivalently D_2 is* coarser *than D_1) and denote this as $D_1 \prec D_2$ when there exists a partition $P = \{p_1, \dots, p_n\}$ of the set $\{\mathbf{q_1}, \dots, \mathbf{q_m}\}$ such that for each i with $1 \le i \le n$, $\mathbf{r_i} = \sum_{\mathbf{q} \in p_i} \mathbf{q}$.*

Interpolants of decompositions ordered by their coarseness can be ordered by logical strength, as stated by the following lemma:

Lemma 1. *Assume D_1, D_2 are two decompositions of the same interpolation problem such that $D_1 \prec D_2$. Let I^{D_1}, I^{D_2} be the decomposed interpolants corresponding to D_1, D_2. Then I^{D_1} implies I^{D_2}.*

Proof. Informally, the implication follows from the fact that each linear inequality of I^{D_2} is a sum of some inequalities in I^{D_1}.

Formally, let I_i denote the i-th inequality in I^{D_2}. Then $I_i \equiv \mathbf{r_i^\intercal} C_A \mathbf{x} \le \mathbf{r_i^\intercal} \mathbf{c_A}$. Since $D_1 \prec D_2$, there is a set $\{I_{i_1}, \dots, I_{i_j}\} \subseteq D_1$ such that for each k with $1 \le k \le j$, $I_{i_k} \equiv \mathbf{q_{i_k}^\intercal} C_A \mathbf{x} \le \mathbf{q_{i_k}^\intercal} \mathbf{c_A}$ and $\mathbf{r_i} = \sum_{k=1}^{j} \mathbf{q_{i_k}}$.

Since $\mathbf{q_{i_k}} \ge \mathbf{0}$, it holds that $I_{i_1} \wedge \dots \wedge I_{i_j} \implies I_i$. This means that I^{D_1} implies every conjunct of I^{D_2}. □

Note that the trivial, single-element decomposition corresponding to Farkas interpolant is the greatest element of this decomposition ordering. Also, for any decomposition of size more than one, replacing any number of elements by their sum yields a coarser decomposition. A possible reason to use a coarser decomposition may be that summing up some of the elements of a decomposition may result in eliminating a shared variable from the decomposition.

4.2 Strength of the Dual Interpolants

Let *Itp* denote an interpolation procedure and let *Itp*(A, B) stand for the interpolant computed by *Itp* for an interpolation problem (A, B). Then by *Itp*$'$ we denote the *dual* interpolation procedure, which works as follows: *Itp*$'(A, B) = \neg$*Itp*(B, A). The duality theorem for interpolation states that *Itp*$'$ is correct interpolation procedure. This can be shown by verifying that the three interpolation conditions hold for *Itp*$'(A, B)$, given they hold for *Itp*(B, A).

Let us denote the interpolation procedure based on Farkas' lemma as Itp_F and the interpolation procedure computing decomposed interpolants as Itp_{DI}. The relation between Itp_F and its dual Itp'_F has been established in [3], namely that $Itp_F(A,B) \implies Itp'_F(A,B)$. We have shown in Lemma 1 that decomposed interpolant always implies Farkas interpolant computed from the same Farkas coefficients. This means that $Itp_{DI}(A,B) \implies Itp_F(A,B)$.

We can use this result to establish similar result for the dual interpolation procedures. Since $Itp_{DI}(B,A) \implies Itp_F(B,A)$, it follows that $\neg Itp_F(B,A) \implies \neg Itp_{DI}(B,A)$ and consequently $Itp'_F(A,B) \implies Itp'_{DI}(A,B)$.

Putting all the results on logical strength together, we obtain

$$Itp_{DI}(A,B) \implies Itp_F(A,B) \implies Itp'_F(A,B) \implies Itp'_{DI}(A,B).$$

Note that while both Itp_F and Itp'_F produce interpolants which are a single inequality and interpolants produced by Itp_{DI} are *conjunctions* of inequalities, interpolants produced by Itp'_{DI} are *disjunctions* of inequalities.

In the following section, we describe the details of the Itp_{DI} interpolation procedure.

5 Finding Decompositions

In this section we present our approach for finding decompositions for linear arithmetic interpolation problems given their Farkas coefficients.

We focus on the task of finding decomposition of $\mathbf{k}_\mathbf{A}^\intercal C_A \mathbf{x}$. Recall that $C_A \in \mathbb{Q}^{l \times n}$ and $\mathbf{x}$ is a vector of variables of length n. Without loss of generality assume that there are no B-local variables since columns of C_A corresponding to B-local variables would contain all zeroes by definition in any case.

Furthermore, without loss of generality, assume the variables in the inequalities of A are ordered such that all A-local variables are before the shared ones. Then let us write

$$C_A = \begin{pmatrix} L & S \end{pmatrix}, \quad \mathbf{x}^\intercal = \begin{pmatrix} \mathbf{x}_L{}^\intercal & \mathbf{x}_S{}^\intercal \end{pmatrix} \tag{5}$$

with $\mathbf{x}_L$ the vector of A-local variables of size p, $\mathbf{x}_S$ the vector of shared variables of size q, $n = p + q$, $L \in \mathbb{Q}^{l \times p}$ and $S \in \mathbb{Q}^{l \times q}$. We know that $\mathbf{k}_\mathbf{A}^\intercal L = \mathbf{0}$ and the goal is to find $\mathbf{k}_{\mathbf{A},\mathbf{i}}$ such that $\sum_i \mathbf{k}_{\mathbf{A},\mathbf{i}} = \mathbf{k}_\mathbf{A}$ and for each i $\mathbf{0} \leq \mathbf{k}_{\mathbf{A},\mathbf{i}} \leq \mathbf{k}_\mathbf{A}$ and $\mathbf{k}_{\mathbf{A},\mathbf{i}}^\intercal L = \mathbf{0}$.

In the following we will consider two cases for computing the decompositions. We first study a common special case where the system A contains rows with no local variables, and give a linear-time algorithm for computing the decompositions. We then move to the general case where the rows of A contain local variables, and provide a decomposition algorithm based on computing a vector basis for a null space of a matrix obtained from A.

5.1 Trivial Elements

First, consider a situation where there is a linear inequality with no local variables. This means there is a row j in C_A (denoted as C_{A_j}) such that all entries in columns corresponding to local variables are 0, i.e., $L_j = \mathbf{0}^\mathsf{T}$. Then $\{I_1, I_2\}$ for $\mathbf{k_{A,1}} = \mathbf{k_A}[j] \times \mathbf{e_j}$ and $\mathbf{k_{A,2}} = \mathbf{k_A} - \mathbf{k_{A,1}}$ is a decomposition. Intuitively, any linear inequality that contains only shared variables can form a stand-alone element of a decomposition. When looking for finest decomposition, we do this iteratively for all inequalities with no local variables. In the next part we show how to look for a non-trivial decomposition when dealing with local variables.

5.2 Decomposing in the Presence of Local Variables

For this section, assume that L has no zero rows (we have shown above how to deal with such rows). We are going to search for a non-trivial decomposition starting with the following observation:

Observation. $\mathbf{k_A^\mathsf{T}} L = 0$. Equivalently, there are no A-local variables in the Farkas interpolant. It follows that $L^\mathsf{T}\mathbf{k_A} = 0$ and $\mathbf{k_A}$ is in the *kernel* of L^T.

Let us denote by $\mathbb{K} = \mathit{ker}(L^\mathsf{T})$ the kernel of L^T.

Theorem 2. *Let* $\mathbf{v_1}, \dots, \mathbf{v_n}$ *be* n *vectors from* $\mathbb{K}$ *such that* $\exists \alpha_1, \dots, \alpha_n$ *with* $\alpha_i \mathbf{v_i} \geq \mathbf{0}$ *for all* i *and* $\mathbf{k_A} = \sum_{i=1}^{n} \alpha_i \mathbf{v_i}$. *Then* $\{\mathbf{w_1}, \dots, \mathbf{w_n}\}$ *for* $\mathbf{w_i} = \alpha_i \mathbf{v_i}$ *is a decomposition of* $\mathbf{k_A}$ *and* $\{I_1, \dots, I_n\}$ *for* $I_i \equiv \mathbf{w_i} C_A \mathbf{x} \leq \mathbf{c_A}$ *is a decomposition.*

Proof. The theorem follows from the definition of decomposition (Definition 1). From the assumptions of the theorem we immediately obtain $\mathbf{k_A} = \sum_{i=1}^{n} \mathbf{w_i}$ and $\mathbf{w_i} \geq \mathbf{0}$. Moreover, $\mathbf{w_i} \in \mathbb{K}$, since $\mathbf{v_i} \in \mathbb{K}$ and $\mathbf{w_i} = \alpha_i \mathbf{v_i}$. As a consequence, $L^\mathsf{T}\mathbf{w_i} = 0$ and it follows that there are no A-local variables in $\mathbf{w_i}^\mathsf{T} C_A \mathbf{x}$. $\square$

Note that if the vectors are not linearly independent then the decomposition contains redundant elements. For example, if $w_3 = w_1 + w_2$ then $I_1 \wedge I_2 \implies I_3$ and I_3 is a redundant conjunct in the corresponding decomposed interpolant.

Good candidates that satisfy most of the assumptions of Theorem 2 (and avoid redundancies) are bases of the vector space $\mathbb{K}$. If $B = \{\mathbf{b_1}, \dots, \mathbf{b_n}\}$ is a basis of $\mathbb{K}$ such that $\mathbf{k_A} = \sum_{i=1}^{n} \alpha_i \mathbf{b_i}$ with $\alpha_i \mathbf{b_i} \geq \mathbf{0}$ for all i, then $\{\alpha_1 \mathbf{b_1}, \dots, \alpha_n \mathbf{b_n}\}$ is a decomposition. Moreover, the decomposition generated by a basis cannot be refined (in the sense of the decomposition order $\prec$) without introducing redundancies. This follows from the fact that replacing one generator in a basis by more that one vector necessarily introduces linear dependency between the generators of the vector space. Thus, the decomposed interpolant from a basis has *maximal* logical strength. The search for a decomposition of Farkas coefficients $\mathbf{k_A}$ by computing a basis of the kernel of the matrix of A-local variables L is described in Algorithm 1.

Function `Nullity` returns the dimension of the kernel. This can be efficiently computed for example using *Rank-Nullity Theorem* by computing Row Echelon Form of M by Gaussian elimination. Only if nullity is at least 2, we can hope to find any non-trivial decomposition. Function `KernelBasis` returns a basis of the kernel of a given matrix while function `Coordinates` returns the coordinates of the given vector with respect to the given basis. An algorithm to compute a basis of the kernel of a matrix can be found in any good introductory book on Linear

```
input  : matrix $M$, vector $\mathbf{v}$ such that $\mathbf{v} \in ker(M)$ and $\mathbf{v} > \mathbf{0}$
output: $(\mathbf{w_1}, \ldots, \mathbf{w_m})$, a decomposition of $\mathbf{v}$, such that $\mathbf{w_i} \in ker(M), \mathbf{w_i} \geq \mathbf{0}$
        and $\sum \mathbf{w_i} = \mathbf{v}$
1 $n \leftarrow$ Nullity($M$)
2 if $n \leq 1$ then return $(\mathbf{v})$
3 $(\mathbf{b_1}, \ldots, \mathbf{b_n}) \leftarrow$ KernelBasis($M$)
4 $(\alpha_1, \ldots, \alpha_n) \leftarrow$ Coordinates($\mathbf{v}, (\mathbf{b_1}, \ldots, \mathbf{b_n})$)
5 $(\mathbf{w_1}, \ldots, \mathbf{w_n}) \leftarrow (\alpha_1 \mathbf{b_1}, \ldots, \alpha_n \mathbf{b_n})$
6 if $\mathbf{w_i} \geq \mathbf{0}$ for each $i$ then return $(\mathbf{w_1}, \ldots, \mathbf{w_n})$
7 else return $(\mathbf{v})$
```

Algorithm 1. Algorithm for decomposition of Farkas coefficients

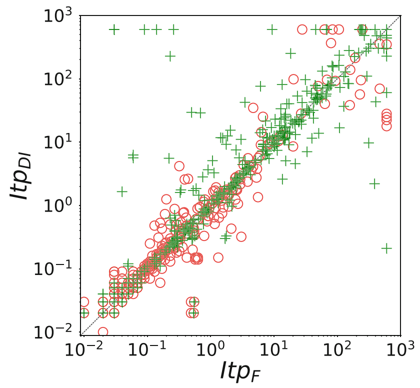
Algebra, see e.g. [5]. If any component of the linear combination is negative, the combination cannot be used and we fall back to the trivial decomposition leading to the original Farkas interpolant. As a basis of a vector space is not unique, the implementation of `KernelBasis` may return an unsuitable basis even if a suitable one exists. This happened even in simple cases, so we implemented a strategy to replace unsuitable elements by a suitable sum of elements, if possible. Our preliminary results using this strategy are promising.

6 Experiments

We have implemented our algorithm in our SMT solver OpenSMT [21], which had already provided a variety of interpolation algorithms for propositional logic [22,33], theory of uninterpreted functions [4] and theory of linear real arithmetic [3]. We implemented both primal and dual versions of decomposed interpolation algorithm, which return the finest decomposition they can find.

We evaluated the effect of decomposed interpolants in a model-checking scenario using the model checker sally relying on OpenSMT for interpolation.[3] The pdkind engine of sally was used, relying on Yices [15] for satisfiability queries and OpenSMT for interpolation queries. We experimented with four LRA interpolation algorithms: the original interpolation algorithms based on Farkas' lemma, Itp_F and Itp'_F, and the interpolation algorithm computing decomposed interpolants, Itp_{DI} and Itp'_{DI}. In each case, we used McMillan's interpolation rules [28] for the Boolean part. For comparison, we ran also a version of sally using MathSAT in its default settings as an interpolation engine instead of OpenSMT. Since OpenSMT does not support the combination of incrementality and interpolation, MathSAT was also used in non-incremental mode in this setting. The results are summarised in Figs. 1 and 2, and Table 1. The result of a portfolio is the virtual best of the results of individual algorithms

[3] Detailed description of the set-up and specifications of the experiments, together with all the results, can be found at http://verify.inf.usi.ch/content/decomposed-interpolants.



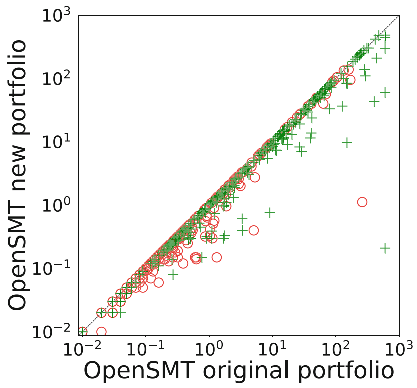


Fig. 1. Evaluation of the decomposed interpolants in model checking scenario. On the left, comparison of performance of SALLY using OPENSMT with different interpolation procedures, Itp_F and Itp_{DI}. On the right, the benefit of adding Itp_{DI} and Itp'_{DI} to the portfolio of interpolation procedures.

in the portfolio. The original portfolio of OPENSMT consists of Itp_F and Itp'_F, while in the new portfolio Itp_{DI} and Itp'_{DI} are added.

We used the same benchmarks as in [36]. They consist of several problem sets related to fault-tolerant algorithms (**om**, **ttesynchro**, **ttastartup**, **unifapprox**, **azadmanesh**, **approxagree**, **hacms**, **misc**), benchmarks from software model checking (**cav12**, **ctigar**), benchmark suite of KIND model checker (**lustre**), simple concurrent programs (**conc**), and problems modeling a lock-free hash table (**lfht**). Each benchmark is a transition system with formulas characterizing initial states, a transition relation and a property that should hold. SALLY can finish with two possible answers (or run out of resources without an answer): *valid* means the property holds and an invariant implying the property has been found; *invalid* means the property does not hold and a counterexample leading to a state where the property does not hold has been found. In the plots, we denote the answers as + and ∘, respectively. The benchmarks were run on Linux machines with Intel E5-2650 v3 processor (2.3 GHz) with 64 GB of memory. Each benchmark was restricted to 600 s of running time and to 4 GB of memory.

Figure 1 illustrates the benefit of adding Itp_{DI} and Itp'_{DI} to the portfolio of OPENSMT interpolation algorithms. The direct comparison of Itp_F and Itp_{DI} clearly shows that in many cases the use of decomposed interpolants outperforms the original procedure, sometimes by an order of magnitude. The comparison of the old and the new portfolio shows that the importance of decomposition is still significant even after taking the capabilities of dual versions into account.

Figure 2 shows the benefit of the new portfolio by comparing the model checker performance to one using a different SMT solver. As far as we know, MATHSAT also computes interpolants from the proof of unsatisfiability and uses interpolation algorithm based on Farkas' lemma for LRA conflicts. Comparing to OPENSMT's Itp_F, we see that the version of SALLY using MATHSAT is superior, most probably

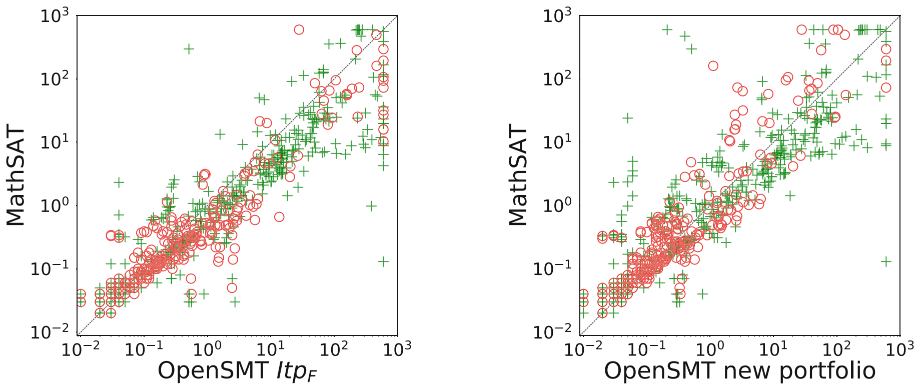


Fig. 2. Comparison of performance for the use of the interpolation procedure of MathSAT and OpenSMT—original Itp_F and the whole portfolio, respectively.

due to the differences in the implementation of the SMT solver. However, using the portfolio of interpolation procedures available in OpenSMT bridges the gap and allows sally to solve more benchmarks as can be seen in Table 1. This also shows a potential improvement for MathSAT if it would offer the same portfolio of interpolation procedures as OpenSMT does.

Table 1 demonstrates the gain in the performance of the model checker from adding Itp_{DI} and Itp'_{DI} to the interpolation portfolio. The results are summarised *by category* with the name of the category and its number of benchmarks in the first column. The two columns per interpolation engine show the number of benchmarks successfully solved (validated/invalidated) within the limits and the total running time for *solved* benchmarks. Not only does the model checker with the extended portfolio solve *more* instances, but it also does so in *less* time.

Table 2 answers the question how often the new interpolation procedure manages to decompose Farkas coefficients, thus returning a different interpolant than the original procedure would. The statistics differ for Itp_{DI} and Itp'_{DI} due to the special nature of the interpolation problems in this model checking algorithm, as B-part *always* contains *only* shared symbols. Theoretically, this means Itp'_{DI} cannot discover any non-trivial elements of decomposition as there are no B-local variables. On the other hand, the decomposition to trivial elements is always possible, as all B-inequalities contain only shared variables. In our implementation, however, we consider the locality of a variable not from the global point of the whole interpolation problem, but from the local point of the current theory conflict. Consequently, even if a variable is shared in the whole problem, it can be local for the current theory conflict and the interpolant is not decomposed even if, from a global point, it could have been.

For Itp_{DI}, the first column reports the number of benchmarks with at least a *single* decomposition (any; with at least one trivial element; with at least one non-trivial element). The second column ("#non-triv. LRA itps") reports the total number of interpolation problems for theory conflict, not counting those without

Table 1. Performance of SALLY with old and new OPENSMT interpolation capabilities. Comparison with MATHSAT with its default interpolation included.

	OPENSMT portfolio Itp_F, Itp'_F		OPENSMT portfolio Itp_F, Itp'_F, Itp_{DI}, Itp'_{DI}		MATHSAT	
Problem set	solved (V/I)	$\sum$ time(s)	solved (V/I)	$\sum$ time(s)	solved (V/I)	$\sum$ time(s)
approxagree (9)	9 (8/1)	101	9 (8/1)	72	9 (8/1)	173
azadmanesh (20)	16 (13/3)	74	16 (13/3)	69	19 (16/3)	102
cav12 (99)	63 (46/17)	3,427	63 (46/17)	2,960	64 (47/17)	796
conc (6)	3 (3/0)	48	4 (4/0)	347	3 (3/0)	38
ctigar (110)	70 (51/19)	1,812	72 (53/19)	1,493	75 (55/20)	1,803
hacms (5)	1 (1/0)	147	1 (1/0)	84	1 (1/0)	55
lfht (27)	17 (17/0)	502	17 (17/0)	502	16 (16/0)	518
lustre (790)	757 (423/334)	5,122	759 (425/334)	4,903	752 (420/332)	5,610
misc (10)	7 (6/1)	80	7 (6/1)	80	7 (6/1)	36
om (9)	9 (7/2)	7	9 (7/2)	7	9 (7/2)	6
ttastartup (3)	1 (1/0)	2	1 (1/0)	2	1 (1/0)	13
ttesynchro (6)	6 (3/3)	11	6 (3/3)	10	6 (3/3)	6
unifapprox (11)	10 (7/3)	21	10 (7/3)	20	11 (8/3)	125
	969 (586/383)	11,354	974 (591/383)	10,549	973 (591/382)	9,281

even theoretical possibility for decomposition. These include the problems where all inequalities were from one part of the problem (resulting in trivial interpolants, either $\top$ or $\bot$) and the problems with a single inequality in the A-part (trivially yielding an interpolant equal to that inequality). The last column reports the number of successfully decomposed interpolants (with at least one trivial element; with at least one non-trivial element). Note that it can happen that a successful decomposition contains both trivial and non-trivial elements. For Itp'_{DI}, statistics regarding decompositions with non-trivial elements are left out as these decompositions were extremely rare. We see that at least one decomposition was possible in only roughly half of all the benchmarks. This explains why there are many points on the diagonal in Fig. 1. On the other hand, it shows that the test for the *possibility* of decomposition is very cheap and does not present a significant overhead. Another conclusion we can draw is that when the structure of the benchmark allows decomposition, the decomposition can often be discovered in many theory conflicts that appear during the solving.

During the evaluation we noticed that a small change in the solver sometimes had a huge effect on the performance of the model checker for a particular benchmark. It made previously unsolved instance easily solvable (or the other way around). To confirm that on some benchmarks Itp_{DI} is really better than Itp_F, we ran the model checker 100 times on chosen benchmarks, each time with a different random seed for the interpolating solver. For the benchmark **dillig03.c.mcmt** from category **ctigar** the model checker using Itp_F did *not* converge (in all runs) while Itp_{DI} ensured convergence in 0.2 s (in all runs). Itp_F also did not solve **fib_bench_safe_v1.mcmt** from category **conc**

Table 2. Interpolation statistics – pwd stands for "Number of problems with at least one decomposition". The numbers in parentheses denote "Decompositions with trivial and with non-trivial elements" (trivial/non-trivial).

	Itp_{DI}			Itp'_{DI}		
Problem set	pwd	#non-triv. LRA itps	#decomp. itps	pwd	#non-triv. LRA itps	#decomp. itps
approxagree (9)	1 (1/0)	7	7 (7/0)	1	18	18
azadmanesh (20)	4 (0/4)	4,831	266 (0/266)	4	4,353	4,353
cav12 (99)	31 (25/15)	1,368,187	7,399 (1,690/5,738)	45	204,036	57,127
conc (6)	3 (3/3)	424,145	215,376 (1,256 /214,120)	3	13	13
ctigar (110)	73 (56/70)	2,982,559	826,621 (29,378 /797,871)	77	152,613	152,612
hacms (5)	5 (5/5)	363,265	15,282 (532/14,750)	5	58,416	58,416
lfht (27)	13 (12/13)	838,094	12,785 (169/12,616)	14	111,060	111,060
lustre (790)	356 (356/192)	2,571,091	1,851,213 (855,958 /1,054,516)	500	1,833,310	1,833,310
misc (10)	5 (4/5)	195,819	62,865 (8,700 /55,042)	6	35,131	35,108
om (9)	4 (4/3)	1,150	236 (206/30)	3	168	168
ttastartup (3)	2 (2/2)	69,699	924 (16/908)	3	11,528	11,528
ttesynchro (6)	4 (4/0)	64	38 (38/0)	5	310	310
unifapprox (11)	0 (0/0)	0	0 (0/0)	2	25	25

and **large_const_c.mcmt** from category **ctigar**, while Itp_{DI} solved them in 42 runs on average in 377 s, and in 80 runs on average in 97 s, respectively. Finally, the benchmark **DRAGON_13.mcmt** from **lustre** was solved by Itp_F in 5 runs on average in 539 s, while it was solved by Itp_{DI} in 23 runs on average in 441 s.

7 Conclusion

In this paper, we have presented a new interpolation algorithm for linear real arithmetic that generalizes the interpolation algorithm based on Farkas' lemma used in modern SMT solvers. We showed that the algorithm is able to compute interpolants in the form of a *conjunction* of inequalities that are logically stronger than the single inequality returned by the original approach. This is useful in the IC3-style model-checking algorithms where Farkas interpolants have been shown to be a source of incompleteness. In our experiments, we have demonstrated that the opportunity to decompose Farkas interpolants occurs frequently in practice and that the decomposition often leads to (i) shortening of solving time and, in some cases, to (ii) solving a problem not solvable by the previous approach.

As the next steps, we plan to investigate how to automatically determine what kind of interpolant would be more useful for the current interpolation query in IC3-style model-checking algorithms. We also plan to investigate other uses of interpolation in model checking where stronger (or weaker) interpolants are desirable [34].

Acknowledgements. We would like to thank Dejan Jovanović for providing the benchmarks and for the help with integrating OpenSMT into sally. This work was supported by the Czech Science Foundation project 17-12465S and by the Swiss National Science Foundation (SNSF) grant 200020_166288.

References

1. Albarghouthi, A., McMillan, K.L.: Beautiful Interpolants. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 313–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_22
2. Alt, L., Fedyukovich, G., Hyvärinen, A.E.J., Sharygina, N.: A proof-sensitive approach for small propositional interpolants. In: Gurfinkel, A., Seshia, S.A. (eds.) VSTTE 2015. LNCS, vol. 9593, pp. 1–18. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29613-5_1
3. Alt, L., Hyvärinen, A.E.J., Sharygina, N.: LRA interpolants from no man's land. In: Strichman, O., Tzoref-Brill, R. (eds.) HVC 2017. LNCS, vol. 10629, pp. 195–210. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70389-3_13
4. Alt, L., Hyvärinen, A.E.J., Asadi, S., Sharygina, N.: Duality-based interpolation for quantifier-free equalities and uninterpreted functions. In: Stewart, D., Weissenbacher, G. (eds.) FMCAD 2017, pp. 39–46. IEEE (2017)
5. Andrilli, S., Hecker, D.: Elementary Linear Algebra, 5th edn. Academic Press, Cambridge (2016). https://doi.org/10.1016/C2013-0-19116-7
6. Barrett, C., de Moura, L., Ranise, S., Stump, A., Tinelli, C.: The SMT-LIB initiative and the rise of SMT. In: Barner, S., Harris, I., Kroening, D., Raz, O. (eds.) HVC 2010. LNCS, vol. 6504, p. 3. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19583-9_2
7. Barrett, C., Sebastiani, R., Seshia, S., Tinelli, C.: Satisfiability modulo theories. Frontiers in Artificial Intelligence and Applications, 1 edn., vol. 185, pp. 825–885 (2009)
8. Bradley, A.R.: SAT-based model checking without unrolling. In: Jhala, R., Schmidt, D. (eds.) VMCAI 2011. LNCS, vol. 6538, pp. 70–87. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-18275-4_7
9. Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: The MathSAT5 SMT solver. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 93–107. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_7
10. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. J. Symbolic Logic **22**(3), 269–285 (1957)
11. Davis, M., Logemann, G., Loveland, D.W.: A machine program for theorem-proving. Commun. ACM **5**(7), 394–397 (1962)
12. Davis, M., Putnam, H.: A computing procedure for quantification theory. J. ACM **7**(3), 201–215 (1960)
13. Detlefs, D., Nelson, G., Saxe, J.B.: Simplify: a theorem prover for program checking. J. ACM **52**(3), 365–473 (2005)
14. D'Silva, V., Kroening, D., Purandare, M., Weissenbacher, G.: Interpolant strength. In: Barthe, G., Hermenegildo, M. (eds.) VMCAI 2010. LNCS, vol. 5944, pp. 129–145. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11319-2_12

15. Dutertre, B.: Yices 2.2. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 737–744. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_49
16. Dutertre, B., de Moura, L.: A fast linear-arithmetic solver for DPLL(T). In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 81–94. Springer, Heidelberg (2006). https://doi.org/10.1007/11817963_11
17. Farkas, G.: A Fourier-féle mechanikai elv alkalmazásai (Hungarian) (On the applications of the mechanical principle of Fourier) (1894)
18. Gurfinkel, A., Kahsai, T., Komuravelli, A., Navas, J.A.: The SeaHorn verification framework. In: Kroening, D., Păsăreanu, C.S. (eds.) CAV 2015. LNCS, vol. 9206, pp. 343–361. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-21690-4_20
19. Gurfinkel, A., Rollini, S.F., Sharygina, N.: Interpolation properties and SAT-based model checking. In: Van Hung, D., Ogawa, M. (eds.) ATVA 2013. LNCS, vol. 8172, pp. 255–271. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-02444-8_19
20. Huberman, B.A., Lukose, R.M., Hogg, T.: An economics approach to hard computational problems. Science **275**(5296), 51–54 (1997)
21. Hyvärinen, A.E.J., Marescotti, M., Alt, L., Sharygina, N.: OpenSMT2: An SMT solver for multi-core and cloud computing. In: Creignou, N., Le Berre, D. (eds.) SAT 2016. LNCS, vol. 9710, pp. 547–553. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-40970-2_35
22. Jančík, P., Alt, L., Fedyukovich, G., Hyvärinen, A.E.J., Kofroň, J., Sharygina, N.: PVAIR: Partial Variable Assignment InterpolatoR. In: Stevens, P., Wąsowski, A. (eds.) FASE 2016. LNCS, vol. 9633, pp. 419–434. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49665-7_25
23. Jančík, P., Kofroň, J., Rollini, S.F., Sharygina, N.: On interpolants and variable assignments. In: FMCAD 2014, pp. 123–130. IEEE (2014)
24. Jovanović, D., Dutertre, B.: Property-directed k-induction. In: FMCAD 2016, pp. 85–92. IEEE (2016)
25. Komuravelli, A., Gurfinkel, A., Chaki, S.: SMT-based model checking for recursive programs. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 17–34. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_2
26. Korovin, K., Tsiskaridze, N., Voronkov, A.: Conflict resolution. In: Gent, I.P. (ed.) CP 2009. LNCS, vol. 5732, pp. 509–523. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04244-7_41
27. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems: Safety. Springer, Heidelberg (1995). https://doi.org/10.1007/978-1-4612-4222-2
28. McMillan, K.L.: Interpolation and SAT-based model checking. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 1–13. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_1
29. McMillan, K.L.: An interpolating theorem prover. Theoret. Comput. Sci. **345**(1), 101–121 (2005)
30. de Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
31. Nieuwenhuis, R., Oliveras, A.: Proof-producing congruence closure. In: Giesl, J. (ed.) RTA 2005. LNCS, vol. 3467, pp. 453–468. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-32033-3_33
32. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. J. Symbolic Logic **62**(3), 981–998 (1997)

33. Rollini, S.F., Alt, L., Fedyukovich, G., Hyvärinen, A.E.J., Sharygina, N.: PeRIPLO: a framework for producing effective interpolants in SAT-based software verification. In: McMillan, K., Middeldorp, A., Voronkov, A. (eds.) LPAR 2013. LNCS, vol. 8312, pp. 683–693. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-45221-5_45
34. Rollini, S.F., Sery, O., Sharygina, N.: Leveraging interpolant strength in model checking. In: Madhusudan, P., Seshia, S.A. (eds.) CAV 2012. LNCS, vol. 7358, pp. 193–209. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31424-7_18
35. Rybalchenko, A., Sofronie-Stokkermans, V.: Constraint solving for interpolation. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 346–362. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-69738-1_25
36. Schindler, T., Jovanović, D.: Selfless interpolation for infinite-state model checking. In: Dillig, I., Palsberg, J. (eds.) VMCAI 2018. LNCS, vol. 10747, pp. 495–515. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73721-8_23
37. Scholl, C., Pigorsch, F., Disch, S., Althaus, E.: Simple interpolants for linear arithmetic. In: DATE 2014, pp. 1–6. IEEE (2014)
38. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (1998)
39. Sheeran, M., Singh, S., Stålmarck, G.: Checking safety properties using induction and a SAT-solver. In: Hunt, W.A., Johnson, S.D. (eds.) FMCAD 2000. LNCS, vol. 1954, pp. 127–144. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-40922-X_8
40. Silva, J.P.M., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Trans. Comput. **48**(5), 506–521 (1999)