Interpolating Quantifier-Free Presburger Arithmetic

Daniel Kroening[1], Jérôme Leroux[2], and Philipp Rümmer[1]

[1] Oxford University Computing Laboratory, United Kingdom
[2] Laboratoire Bordelais de Recherche en Informatique, France

Abstract. Craig interpolation has become a key ingredient in many symbolic model checkers, serving as an approximative replacement for expensive quantifier elimination. In this paper, we focus on an interpolating decision procedure for the full quantifier-free fragment of *Presburger Arithmetic*, i.e., linear arithmetic over the integers, a theory which is a good fit for the analysis of software systems. In contrast to earlier procedures based on quantifier elimination and the Omega test, our approach uses integer linear programming techniques: relaxation of interpolation problems to the rationals, and a complete branch-and-bound rule tailored to efficient interpolation. Equations are handled via a dedicated polynomial-time sub-procedure. We have fully implemented our procedure on top of the SMT-solver OpenSMT and present an extensive experimental evaluation.

1 Introduction

Craig interpolation has become a key ingredient in many symbolic model checkers, serving as an approximative replacement for expensive quantifier elimination [10]. The application of Craig interpolants in lieu of quantifier elimination relies on the availability of an effective *interpolating decision procedure*. In this paper, we focus on an interpolating decision procedure for the *quantifier-free fragment of Presburger Arithmetic (QFPA for short)*, that is linear arithmetic over the integers, a theory which is a good fit for the analysis of software systems. An interpolant ψ for a pair (ϕ_A, ϕ_B) of Presburger formulas is a Presburger formula such that free variables in ψ occur both in ϕ_A and ϕ_B, and such that ϕ_A entails ψ and ϕ_B entails $\neg\psi$.

Interpolating decision procedures typically derive the interpolant from a proof of inconsistency of ϕ_A and ϕ_B, which in turn is computed by a decision procedure for the underlying logic. Decision problems arising in software analysis are often large, and call for a scalable algorithm. The most efficient decision procedures for the quantifier-free fragment of the Presburger arithmetic known today use the Simplex algorithm in combination with a variant of the *branch-and-bound technique*. The Simplex algorithm is used to solve the *relaxed problem*, in which the variables are permitted to take fractional values. In case a variable x obtains the fractional value r, branch-and-bound will consider the two sub-problems in which $x \leq \lfloor r \rfloor$ or $x \geq \lceil r \rceil$, respectively. The original problem has an integer solution iff one of the two sub-problems has a solution. Branch-and-bound is incomplete by itself, and usually augmented by a *cutting-plane* technique, e.g., *Gomory's cutting planes*. An instance of an efficient implementation of these techniques is the SMT-solver Z3 [6].

In principle, any cut-based decision procedure for Presburger can be used for the computation of interpolants. The primary problem is computational cost: for the most common cut rules (in particular for Gomory's cutting planes) it is possible to construct cases where the derivation of interpolants from proofs has exponential complexity. This high complexity is caused by *mixed cuts*, which involve rounding (rational) constant terms of inequalities that are derived from both ϕ_A and ϕ_B. Intuitively, interpolating calculi rely on identifying which parts of ϕ_A and ϕ_B are contributing to an intermediate argument; additional effort is required when rounding intermediate arguments derived from both ϕ_A and ϕ_B.

The contribution of this paper is a novel interpolating decision procedure for the full QFPA fragment. Our algorithm computes in polynomial time interpolants for two classes of constraints (i) conjunctions of inequality constraints unsatisfiable over the rationals, and (ii) conjunctions of equality and divisibility constraints unsatisfiable over the integers. For the full QFPA fragment, the algorithm is exponential in the worst case. This complexity is proved tight since we exhibit formulas such that every interpolant is exponentially large. Moreover the algorithm improves the doubly exponential upper bound complexity known for the computation of interpolants based on the elimination of blocks of quantifiers [17]. Our general procedure integrates efficient reasoning and interpolation for equalities by means of a transformation of matrices into Smith Normal Form, which resembles a known procedure for interpolating linear diophantine equations [7]. For reasoning about inequalities, our procedure uses a complete version of the branch-and-cut principle that avoids mixed cuts and therefore allows interpolant extraction from proofs in polynomial time. Since the proof size is exponentially large in the worst case, we deduce an exponential upper bound for the runtime of the algorithm.

Related Work. Interpolation procedures have been proposed for various fragments of linear integer arithmetic. McMillan considers the logic of difference-bound constraints [12]. This logic, a fragment of QFPA, is decidable by reduction to rational arithmetic. As an extension, Cimatti et al. [5] present an interpolation procedure for the unit two variable per inequality (UTVPI) fragment of linear integer arithmetic. Both fragments allow efficient reasoning and interpolation, but are not sufficient to express many typical program constructs, such as integer division. In [7], interpolation procedures for QFPA restricted to conjunctions of integer linear (dis)equalities, and for QFPA restricted to conjunctions of divisibility constraints are given. The combination of both fragments with integer linear inequalities is not supported, however. Our work closes this gap, as it permits predicates involving all types of constraints.

Lynch et al. [9] define an interpolation procedure for linear rational arithmetic, and extend it to integer arithmetic by means of Gomory cuts. For integer arithmetic, however, interpolation in [9] can produce formulas that violate the vocabulary condition (i.e., can contain variables that are not common to ϕ_A and ϕ_B), and are therefore not true interpolants. The problem is that Gomory cuts used in [9] do not prevent mixed cuts, for which no efficient interpolation is possible in QFPA.

Brillout et al. [2] define a complete interpolating sequent calculus for QFPA. The calculus contains a rule strengthen that is general enough to simulate arbitrary (possibly mixed) Gomory cuts, but in general causes exponential complexity of interpolant

extraction from proofs. In contrast, our cut rule (which is embedded in an effective decision procedure) enables extraction with polynomial complexity.

The recent SMT-solver *SmtInterpol* decides and interpolates problems in linear integer arithmetic, apparently using an architecture similar to the one in [11]. To the best of our knowledge, the precise design and calculus of *SmtInterpol* has not been documented in publications yet (see Sect. 9 for an empirical comparison with our approach).

Interpolation for rational arithmetic is a well-explored field. McMillan presents an interpolating theorem prover for linear rational arithmetic and uninterpreted functions [11]; an interpolating SMT-solver for the same logic has been developed by Beyer et al. [1]. Rybalchenko et al. introduce an algorithm for interpolating rational arithmetic with uninterpreted functions without the need for explicit proofs [15].

2 Interpolation For Quantifier-Free Presburger Formulas

Naturally, if there exists an interpolant ψ for (ϕ_A, ϕ_B) then $\phi_A \wedge \phi_B$ is unsatisfiable. Conversely, if $\phi_A \wedge \phi_B$ is unsatisfiable, interpolants for (ϕ_A, ϕ_B) can be obtained by introducing the sets X_A, X_B of free variables of respectively ϕ_A and ϕ_B, and the following Presburger formulas:

$$\begin{array}{lcll} \psi_\bot & = & (\exists x)_{x \in X_A \setminus X_B} & \phi_A \\ \psi_\top & = & \neg\,(\exists x)_{x \in X_B \setminus X_A} & \phi_B \end{array}$$

Since $\phi_A \wedge \phi_B$ is unsatisfiable we observe that $\psi_\bot$ and $\psi_\top$ are two interpolants for (ϕ_A, ϕ_B). The formulas $\psi_\bot$ and $\psi_\top$ are respectively called the *strongest interpolant* and the *weakest interpolant* since $\psi_\bot$ entails ψ and ψ entails $\psi_\top$ for any interpolant ψ.

We are interested in computing quantifier-free Presburger interpolants for pairs of quantifier-free Presburger formulas. Formulas in this logic are defined by fixing a countable set X of variables. Quantifier-free Presburger formulas are formulas in the following grammar where $x \in X$, $\alpha \in \mathbb{Z}$, $\beta \in \mathbb{Z}$ and $m \in \mathbb{N}_{\geq 2}$:

$$\begin{array}{lcl} \phi & ::= & p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \\ p & ::= & l \neq \beta \mid l = \beta \mid l \leq \beta \mid l \in \beta + m\mathbb{Z} \\ l & ::= & 0 \mid \alpha x \mid l + l \end{array}$$

The category l denotes *linear terms*. The category p denotes *predicates of linear arithmetic*. For simplicity reason, we only allow constants β as right-hand side of the predicates. Predicates $l \neq \beta$, $l = \beta$ and $l \leq \beta$ are respectively *disequality predicates*, *equality predicates*, and *inequality predicates*. Predicates $l \in \beta + m\mathbb{Z}$ are *divisibility predicates*, which are short-hand notation for $\exists x\ l - mx = \beta$. These predicates are included to allow quantifier-free interpolation. In fact, let us consider the pair $(x - 2y = 0,\ x - 2z = 1)$ of quantifier-free Presburger formulas. Note that x is the unique free variable that occurs in both formulas. The even divisibility predicate $x \in 2\mathbb{Z}$ is an interpolant; any interpolant requires at least one divisibility predicate.

The semantics of Presburger formulas is defined as is common over the domain $\mathbb{Z}$ of integers. We write $\phi \models \psi$ to express that ϕ *entails* ψ, i.e., ψ holds whenever ϕ holds.

Since Presburger formulas are effectively equivalent to quantifier-free Presburger formulas, we can compute two quantifier-free Presburger formulas $\psi'_\bot$ and $\psi'_\top$ equivalent to $\psi_\bot$ and $\psi_\top$ respectively. In particular if $\phi_A \wedge \phi_B$ is unsatisfiable, we deduce that $\psi'_\bot$ and $\psi'_\top$ are two quantifier-free interpolants for (ϕ_A, ϕ_B). However, the computation of $\psi'_\bot$ or $\psi'_\top$ requires a lot of useless computational efforts. For instance if ϕ_A is a formula of the form $(x = 0) \wedge \phi'_A$ and ϕ_B is a formula of the form $(x = 1) \wedge \phi'_B$ where ϕ'_A and ϕ'_B are very complex Presburger formulas, it is sufficient to consider $\psi = (x = 0)$ to obtain an interpolant for (ϕ_A, ϕ_B); eliminating variables for computing $\psi'_\bot$ and $\psi'_\top$ can be very difficult. From a theoretical point of view, up to our knowledge the best known upper-bound complexity for eliminating blocks of existential quantifiers is double-exponential [17].

In this paper we provide an algorithm computing interpolants for the QFPA fragment in exponential time in the worst case. We first show that this result is tight. For this purpose, consider the following families of formulas (where $n \in \mathbb{N}_{>1}$):

$$\phi^n_A \;=\; -n < y + 2nx \leq 0, \qquad \phi^n_B \;=\; 0 < y + 2nz \leq n\,.$$

We can observe that ϕ^n_A and ϕ^n_B are inconsistent, and that the only interpolant for the interpolation problem (ϕ^n_A, ϕ^n_B) is the following formula ψ (up to equivalence):

$$\psi \;=\; \Big(y \in -n + 1 + 2n\mathbb{N}\Big) \vee \Big(y \in -n + 2 + 2n\mathbb{N}\Big) \vee \cdots \vee \Big(y \in 2n\mathbb{N}\Big)$$

The size of ψ is linear in n, and therefore exponential in the size of (ϕ^n_A, ϕ^n_B); the same holds for all equivalent quantifier-free formulas in Presburger arithmetic.

Using a SAT approach [11] we reduce the interpolation computation problem to conjunctions of *literals* (predicates or negation of predicates) extracted from ϕ_A and ϕ_B. In particular, w.l.o.g. we can assume that ϕ_A, ϕ_B are conjunctions of *literals*. By introducing fresh variables, we can assume that explicit divisibility predicates do not appear. In fact, let us consider the formulas ϕ'_A and ϕ'_B obtained from ϕ_A and ϕ_B by replacing $l \in \beta + m\mathbb{Z}$ and $\neg(l \in \beta + m\mathbb{Z})$ by respectively $l - mx = \beta$ and $l - mx - y = \beta \wedge -y \leq -1 \wedge y \leq m - 1$ where x, y are two fresh variables distinct for each replaced predicate. Since introduced variables are local to either ϕ'_A or ϕ'_B we deduce that any formula is an interpolant for (ϕ_A, ϕ_B) if and only if it is an interpolant for (ϕ'_A, ϕ'_B). Thus, we can assume without loss of generality that ϕ_A and ϕ_B do not contain divisibility predicates.

Finally, since the negations of the predicates $l \neq \beta$, $l = \beta$, $l \leq \beta$ are equivalent to the predicates $l = \beta$, $l \neq \beta$, $-l \leq -\beta - 1$, we can assume that the literals of ϕ_A and ϕ_B are predicates (without negation). We have reduced our problem to the computation of interpolants for formulas ϕ_A, ϕ_B that are conjunctions of disequality, equality and inequality predicates.

3 Overview of the Interpolation Procedure

We assume the vocabulary $X = \{x_1, \ldots, x_n\}$, using an arbitrary but fixed enumeration of the variables, and denote the vector of all variables by $x = (x_1, \ldots, x_n)^t$. We identify

a linear term l with the matrix product $l = u^t x$ where $u = (\alpha_1, \ldots, \alpha_n)^t \in \mathbb{Z}^n$ denotes coefficients of x in l, i.e. $l = \alpha_1 x_1 + \cdots + \alpha_n x_n$. We associate to a predicate p the vector $u_p \in \mathbb{Z}^n$, the relation $\#_p \in \{\neq, =, \leq\}$, and the integer $\beta_p \in \mathbb{Z}$ such that p is denoted by $u_p^t x \#_p \beta_p$. *Valuations* of X are identified with vectors $v = (v_1, \ldots, v_n)^t \in \mathbb{Z}^n$ such that v *satisfies* a predicate p if $u_p^t v \#_p \beta_p$ holds. We introduce the *ith elementary vector* $e_{i,n}$ *of* $\mathbb{Z}^n$ (simply denoted by e_i when n is unambiguous) defined by:

$$e_{i,n} = (\underbrace{0, \ldots, 0}_{i-1 \text{ zeroes}}, 1, 0, \ldots, 0)^t \in \mathbb{Z}^n$$

Predicates are strengthened with *interval* labels. The ordered set $(\mathbb{Z}, \leq)$ is extended into $(\mathbb{Z}_\infty, \leq)$ where $\mathbb{Z}_\infty = \mathbb{Z} \cup \{-\infty, \infty\}$ and $\leq$ satisfies $-\infty \leq \delta \leq \infty$ for every $\delta \in \mathbb{Z}_\infty$. An *(integral) interval* is a a set of the form $[\![\delta_-, \delta_+]\!] = \{\delta \in \mathbb{Z} \mid \delta_- \leq \delta \leq \delta_+\}$ where $\delta_-, \delta_+ \in \mathbb{Z}_\infty$. The interval $[\![\delta, \delta]\!]$ where $\delta \in \mathbb{Z}$ is simply denoted by $\{\delta\}$. In the sequel, a predicate p labelled with an interval I is denoted by $(p)_I$. Semantically, a labelled predicate $(p)_I$ is satisfied by a valuation v if v satisfies p and $u_p^t v \in I$. In order to simplify the presentation, we assume that $I \subseteq \{\beta_p\}$ if p is an equality and $I \subseteq [\![-\infty, \beta_p]\!]$ if p is an inequality. The label of a disequality can be any interval. Observe that any unlabeled formula ϕ is equivalent to a labelled one satisfying the previous labeling conventions. Given a conjunction ϕ of labelled predicates, we denote by $\bar{\phi}$ the formula obtained from ϕ by unlabeling the predicates.

We first show on the following example how the unsatisfiability of a conjunction $\phi = \phi_A \wedge \phi_B$ can be discovered by analyzing *systems of inequalities over the rational numbers* and *systems of equalities over the integers*. We consider the following formulas:

$$\begin{aligned} \phi_A &= (x - 2y \leq 0)_{[\![-\infty,0]\!]} \wedge \\ &\quad (2y - x \leq 0)_{[\![-\infty,0]\!]} \end{aligned} \qquad \begin{aligned} \phi_B &= (x - 2z \leq 1)_{[\![-\infty,1]\!]} \wedge \\ &\quad (2z - x \leq -1)_{[\![-\infty,-1]\!]} \end{aligned}$$

The label of $(2z - x \leq -1)_{[\![-\infty,-1]\!]}$ is first partitioned into $J_1 \cup J_2$ where $J_1 = [\![-\infty, -2]\!]$ and $J_2 = \{-1\}$. We observe that ϕ is unsatisfiable if and only if both the formulas $\phi_1 = \phi_{A,1} \wedge \phi_{B,1}$ and $\phi_2 = \phi_{A,2} \wedge \phi_{B,2}$ are unsatisfiable, where $\phi_{A,1} = \phi_A$, $\phi_{A,2} = \phi_A$, and:

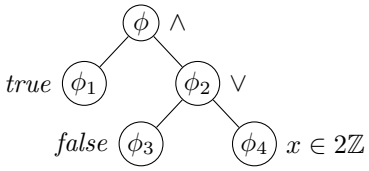
$$\begin{aligned} \phi_{B,1} &= (x - 2z \leq 1)_{[\![-\infty,1]\!]} \wedge \\ &\quad (2z - x \leq -1)_{J_1} \end{aligned} \qquad \begin{aligned} \phi_{B,2} &= (x - 2z \leq 1)_{[\![-\infty,1]\!]} \wedge \\ &\quad (2z - x \leq -1)_{J_2} \end{aligned}$$

We introduce the system of *interval predicates* extracted from ϕ_1 labels, i.e. $-\infty \leq x - 2y \leq 0 \wedge -\infty \leq 2y - x \leq 0 \wedge -\infty \leq x - 2z \leq 1 \wedge -\infty \leq 2z - x \leq -2$. An LP-solver decides in polynomial time its unsatisfiability over the rational numbers. In particular we deduce that ϕ_1 is unsatisfiable over the integers. The unsatisfiability of ϕ_2 is obtained by partitioning the label of $(x - 2y \leq 0)_{[\![-\infty,0]\!]}$ into $J_3 \cup J_4$ where $J_3 = [\![-\infty, -1]\!]$ and $J_4 = \{0\}$. We observe that ϕ_2 is unsatisfiable if and only if both the following formulas $\phi_3 = \phi_{A,3} \wedge \phi_{B,3}$ and $\phi_4 = \phi_{A,4} \wedge \phi_{B,4}$ are unsatisfiable, where $\phi_{B,3} = \phi_{B,2}$, $\phi_{B,4} = \phi_{B,2}$, and:

$$\begin{aligned} \phi_{A,3} &= (x - 2y \leq 0)_{J_3} \wedge \\ &\quad (2y - x \leq 0)_{[\![-\infty,0]\!]} \end{aligned} \qquad \begin{aligned} \phi_{A,4} &= (x - 2y \leq 0)_{J_4} \wedge \\ &\quad (2y - x \leq 0)_{[\![-\infty,0]\!]} \end{aligned}$$

From the system of interval predicates extracted from ϕ_3 labels, an LP-solver shows that ϕ_3 is unsatisfiable. Finally, let us consider the system of equalities extracted from the ϕ_4 labels, i.e. $x - 2y = 0 \wedge 2z - x = -1$. Since this system is unsatisfiable over the integers, we deduce that ϕ_4 is unsatisfiable. We have proved that ϕ is unsatisfiable by strengthening predicates until either a system of inequalities becomes unsatisfiable over the rational numbers, or a system of equalities becomes unsatisfiable over the integers.

Now, we exhibit a way for computing an interpolant ψ for (ϕ_A, ϕ_B). From the system of inequalities proving that ϕ_1 is unsatisfiable over the rational numbers, we deduce in Sect. 5 that $\psi_1 = (true)$ is an interpolant for $(\phi_{A,1}, \phi_{B,1})$. The same approach shows that $\psi_3 = (false)$ is an interpolant for $(\phi_{A,3}, \phi_{B,3})$. From the system of equalities proving that ϕ_4 is unsatisfiable, we deduce in Sect. 4 that $\psi_4 = (x \in 2\mathbb{Z})$ is an interpolant for $(\phi_{A,4}, \phi_{B,4})$. Finally, we show in Sect. 7 that an interpolant for (ϕ_A, ϕ_B) can be obtained from ψ_1, ψ_3 and ψ_4 by considering the following tree where the leaves ϕ_1, ϕ_3 and ϕ_4 are respectively labelled by the interpolants ψ_1, ψ_3 and ψ_4, where the node ϕ is labelled by $\wedge$ since the partitioned label of ϕ comes from its B part, and where the node ϕ_2 is labelled by $\vee$ since the partitioned label of ϕ_2 comes from its A part. This tree provides the interpolant $\psi = true \wedge (false \vee x \in 2\mathbb{Z})$ for (ϕ_A, ϕ_B):



Our general algorithm follows this approach. Now, let us assume that ϕ_A and ϕ_B are any conjunctions of labelled predicates. Interpolants for (ϕ_A, ϕ_B) or valuations w satisfying $\bar{\phi}_A \wedge \bar{\phi}_B$ are computed using algorithm interpolant(ϕ_A , ϕ_B).

```
1 interpolant ( φ_A , φ_B )
2   if check_equality( φ_A , φ_B ) returns a formula ψ return ψ
3   if check_inequality( φ_A , φ_B ) returns a formula ψ return ψ
4   if check_unsatpred( φ_A , φ_B ) returns a formula ψ return ψ
5   return strengthening( φ_A , φ_B )
```

This algorithm first executes three sub-algorithms check_equality, check_inequality and check_unsatpred respectively presented in Sect. 4, Sect. 5 and Sect. 6:

- check_equality returns in polynomial time an interpolant if a system of equalities extracted from ϕ_A and ϕ_B labels is unsatisfiable over the integers.
- check_inequality returns in polynomial time an interpolant if a system of inequalities extracted from ϕ_A and ϕ_B labels is unsatisfiable over the rational numbers.
- check_unsatpred returns in linear time an interpolant if an unsatisfiable labelled predicate occurs in ϕ_A or ϕ_B. This sub-algorithm is required for the termination when disequalities occur in ϕ_A or ϕ_B.

When these sub-algorithms fail in computing an interpolant, the sub-algorithm strengthening is executed. It tries to compute a valuation satisfying $\bar{\phi}_A \wedge \bar{\phi}_B$. If it fails, the label of a predicate is partitioned and algorithm interpolant is recursively called on each element of the partition. This last sub-algorithm is presented in Sect. 7.

4 Unsatisfiable Equalities Over The Integers

This section describes interpolation in the case that the inconsistency of $\phi_A \wedge \phi_B$ is caused by equations. To this end, we extract a system $U_A x = d_A$ of equations from ϕ_A, where $U_A \in \mathbb{Z}^{m \times n}$ is an integer matrix and $d_A \in \mathbb{Z}^m$ is an integer vector. The system $U_A x = d_A$ consists of all equations $u_p^t x = \delta$ such that ϕ_A contains a predicate p labelled with a singleton $J = \{\delta\}$. The same is done for ϕ_B by introducing $U_B \in \mathbb{Z}^{l \times n}$ and $d_B \in \mathbb{Z}^l$. We also introduce the formulas ϕ'_A and ϕ'_B obtained from ϕ_A and ϕ_B by keeping the other labelled predicates $(p)_I$ with I not reduced to a singleton. The conjunctions ϕ_A, ϕ_B can then be represented in the form

$$\phi_A \;=\; U_A x = d_A \wedge \phi'_A, \qquad \phi_B \;=\; U_B x = d_B \wedge \phi'_B$$

In order to examine the satisfiability of the two systems $U_A x = d_A$, $U_B x = d_B$ of equations, we combine them to

$$Ux = d, \qquad U = \begin{pmatrix} U_A \\ U_B \end{pmatrix} \in \mathbb{Z}^{(l+m)\times n},\; d = \begin{pmatrix} d_A \\ d_B \end{pmatrix} \in \mathbb{Z}^{l+m}$$

and solve them by transforming the matrix U into *Smith Normal Form* (SNF):

Lemma 4.1 (Smith Normal Form of integer matrices). *Suppose $U \in \mathbb{Z}^{k\times n}$ is an integer matrix. U can be represented as $U = LSR$, such that $L \in \mathbb{Z}^{k\times k}$ and $R \in \mathbb{Z}^{n\times n}$ are invertible (in the respective rings of integer matrices), and $S \in \mathbb{Z}^{k\times n}$ is in* Smith Normal Form*:*

$$S = \begin{pmatrix} \alpha_1 & 0 & \cdots & & & \cdots & 0 \\ 0 & \alpha_2 & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \\ & & \ddots & \alpha_r & 0 & & \\ \vdots & & & 0 & 0 & \cdots & \vdots \\ 0 & \cdots & & & & \cdots & 0 \end{pmatrix}$$

where $r \leq \min\{k, n\}$ and $\alpha_1, \ldots, \alpha_r$ are positive integers such that $\alpha_{i+1} \in \alpha_i \mathbb{Z}$ for all $i \in \{1, \ldots, r-1\}$. The matrices L, S, R can effectively be computed from U in polynomial time [8].

Given the decomposition $U = LSR$, the satisfiability of the system $Ux = d \Leftrightarrow SRx = L^{-1}d$ can directly be determined: a solution to the equations exists if and only if (i) each element α_i of S divides the ith component of $L^{-1}d$, and (ii) for each $r < i \leq k$ the ith component of $L^{-1}d$ is zero.

We first consider the case that the system $Ux = d$ is unsatisfiable (satisfiable systems are discussed in Sect. 7). In this case, an interpolant can be computed from the

equations without involving the inequalities or disequalities in ϕ'_A, ϕ'_B. An interpolation procedure for equations has been described in [7] (using transformation of matrices to Hermite Normal Form) and can easily be carried over to our context of matrices in SNF.

If $Ux = d$ is unsatisfiable, then the equivalent system $S(Rx) = L^{-1}d$ contains an unsatisfiable equation

$$e_i^t S(Rx) = e_i^t L^{-1} d$$

such that the right-hand side $e_i^t L^{-1} d$ cannot be represented as an integral linear combination of the left-hand side coefficients $e_i^t S$. This equation can be obtained as a linear combination of the equations in $Ux = d$ by left-multiplying with the row vector $s^t = e_i^t L^{-1}$. Restricting this linear combination to the equations from ϕ_A and eliminating variables that only occur in ϕ_A (the variables $X_A \backslash X_B$) yields an interpolant:

$$\psi \;=\; (\exists x_j)_{x_j \in X_A \backslash X_B}\; s^t \begin{pmatrix} U_A \\ 0 \end{pmatrix} x = s^t \begin{pmatrix} d_A \\ 0 \end{pmatrix}$$

Note that a quantifier-free interpolant can trivially be obtained by rewriting the existential quantifiers to a divisibility constraint: a formula like $\exists y_1, \ldots, y_u.\ \beta_1 y_1 + \cdots + \beta_u y_u + l = \beta$ is equivalent to the constraint $l \in \beta + \gcd(\beta_1, \ldots, \beta_u)\mathbb{Z}$.

To see that ψ is indeed an interpolant for (ϕ_A, ϕ_B), we can first observe that the following entailments hold:

$$\phi_A \;\models\; U_A x = d_A \;\models\; s^t \begin{pmatrix} U_A \\ 0 \end{pmatrix} x = s^t \begin{pmatrix} d_A \\ 0 \end{pmatrix} \;\models\; \psi$$

Vice versa, because $s^t U x = s^t d$ is unsatisfiable and the variables $X_A \backslash X_B$ do not occur in ϕ_B, it is also the case that ϕ_B and ψ are inconsistent:

$$\phi_B \;\models\; s^t \begin{pmatrix} 0 \\ U_B \end{pmatrix} x = s^t \begin{pmatrix} 0 \\ d_B \end{pmatrix} \;\models\; \neg\psi$$

The following algorithm summarizes the equality interpolation procedure:

---

1 check_equality(ϕ_A , ϕ_B)
2 extract equality systems $U_A x = d_A$ and $U_B x = d_B$ from ϕ_A and ϕ_B
3 `let` L, S, R be the Smith Normal Form decomposition of U
4 `if` there exists i such that $e_i^t S R x = e_i^t L^{-1} d$ is unsatisfiable
5 `let` $s^t = e_i^t L^{-1}$
6 `return` a divisibility predicate equivalent to:
7 $(\exists x)_{x \in X_A \backslash X_B}\; s^t \binom{U_A}{0} x = s^t \binom{d_A}{0}$

Proposition 4.2. *Algorithm* check_equality*(ϕ_A , ϕ_B) returns in polynomial time an interpolant for (ϕ_A, ϕ_B) if the system of equalities $Ux = d$ is not satisfiable over the integers.*

5 Unsatisfiable Inequalities Over The Rationals

Interpolation procedures for linear inequalities over the rationals have been described in [13, 11], and are in the following paragraphs adapted to our setting. In order to examine the satisfiability of $\phi_A \wedge \phi_B$ over the rationals, we extract systems of inequalities $C_A x \leq c_A$ and $C_B x \leq c_B$ (with $C_A \in \mathbb{Z}^{m' \times n}, C_B \in \mathbb{Z}^{l' \times n}, c_A \in \mathbb{Z}^{m'}$, and $c_B \in \mathbb{Z}^{l'}$) from the labelled predicates in ϕ_A, ϕ_B. More precisely, whenever ϕ_A contains a predicate $(p)_I$ such that $I = [\![\delta_-, \delta_+]\!]$ then $C_A x \leq c_A$ contains the inequalities $-u_p^t x \leq -\delta_-$ and $u_p^t x \leq \delta_+$ if $\delta_-, \delta_+ \in \mathbb{Z}$. Predicates labelled with an interval I such that $\delta_- = -\infty$ or $\delta_+ = \infty$ are in the same way translated to single inequalities.

The system $C_B x \leq c_B$ is constructed in the same manner from ϕ_B. As in Sect. 4, we then combine both systems into one:

$$Cx \leq c, \qquad C = \begin{pmatrix} C_A \\ C_B \end{pmatrix} \in \mathbb{Z}^{(l'+m') \times n},\ c = \begin{pmatrix} c_A \\ c_B \end{pmatrix} \in \mathbb{Z}^{l'+m'}$$

A complete criterion for the solvability of $Cx \leq c$ is given by *Farkas' lemma* [16]:

Lemma 5.1 (Farkas). *Suppose $C \in \mathbb{Q}^{k \times n}$ is a rational matrix and $c \in \mathbb{Q}^k$ is a vector. Exactly one of the following statements is true:*

- *The system $Cx \leq c$ is satisfiable: there is a vector $v \in \mathbb{Q}^n$ such that $Cv \leq c$.*
- *There is a non-negative vector $w \in \mathbb{Q}^k$ such that $w^t C = 0$ and $w^t c < 0$.*

We can decide in polynomial time which case holds, and simultaneously compute the corresponding vector v or w.

For the rest of this section, let us assume that the second case holds, and that we have computed a non-negative vector $w \in \mathbb{Q}^{l'+m'}$ as in the lemma (the first case is discussed in the next section). Without loss of generality, we assume that w is integral, because w can be multiplied with any possibly occurring denominators. The following inequality is an interpolant for (ϕ_A, ϕ_B):

$$\psi \ = \ w^t \begin{pmatrix} C_A \\ 0 \end{pmatrix} x \leq w^t \begin{pmatrix} c_A \\ 0 \end{pmatrix}$$

To see that ψ is an interpolant, first recall that $w^t C = 0$, which implies that the term $w^t \binom{C_A}{0} x$ only contains variables that also occur in $w^t \binom{0}{C_B} x$. This means that all free variables in ψ occur both in ϕ_A and ϕ_B.

Furthermore, the entailment $\phi_A \models \psi$ holds:

$$\phi_A \ \models \ C_A x \leq c_A \ \models \ \begin{pmatrix} C_A \\ 0 \end{pmatrix} x \leq \begin{pmatrix} c_A \\ 0 \end{pmatrix} \ \models \ \psi$$

We can, vice versa, derive a formula from ϕ_B that contradicts ψ, because the combined inequality $w^t C x \leq w^t c$ is unsatisfiable by construction:

$$\phi_B \ \models \ C_B x \leq c_B \ \models \ \begin{pmatrix} 0 \\ C_B \end{pmatrix} x \leq \begin{pmatrix} 0 \\ c_B \end{pmatrix} \ \models \ w^t \begin{pmatrix} 0 \\ C_B \end{pmatrix} x \leq w^t \begin{pmatrix} 0 \\ c_B \end{pmatrix} \ \models \ \neg\psi$$

Altogether, we have proved that ψ is an interpolant for (ϕ_A, ϕ_B). The following algorithm summarizes the inequality interpolation procedure:

check_inequality(ϕ_A , ϕ_B)
  extract inequality systems $C_A x \leq c_A$ and $C_B x \leq c_B$ from ϕ_A and ϕ_B
  `if` there exists $w \in \mathbb{Z}^k$ such that $w^t C = 0$ and $w^t c < 0$
    `return` the inequality predicate:
      $w^t \binom{C_A}{0} x \leq w^t \binom{c_A}{0}$

Proposition 5.2. *Algorithm* check_inequality*(ϕ_A , ϕ_B) returns in polynomial time an interpolant for (ϕ_A, ϕ_B) if the system of inequalities $Cx \leq c$ is not satisfiable over the rationals.*

6 Unsatisfiable Predicates

We observe that *false* or *true* are trivial interpolants for (ϕ_A, ϕ_B) if an unsatisfiable predicate $(p)_I$ occurs in ϕ_A or ϕ_B. Algorithm check_unsatpred implements this idea. This algorithm is important for the termination of algorithm interpolant. In fact, an alternative version of algorithm interpolant without check_unsatpred never terminates on (ϕ_A, ϕ_B) with $\phi_A = (x = 0)_{\{0\}}$ and $\phi_B = (x \neq 0)_{\mathbb{Z}}$.

check_unsatpred(ϕ_A , ϕ_B)
  `if` an unsatisfiable predicate $(p)_I$ occurs in ϕ_A `return` *false*
  `if` an unsatisfiable predicate $(p)_I$ occurs in ϕ_B `return` *true*

Proposition 6.1. *Algorithm* check_unsatpred*(ϕ_A , ϕ_B) returns in linear time an interpolant for (ϕ_A, ϕ_B) if an unsatisfiable predicate $(p)_I$ occurs in ϕ_A or ϕ_B.*

7 When Strengthening is Necessary

We assume that (i) the system of equalities $Ux = d$ introduced in Sect. 4 admits an integral solution, and (ii) the system of inequalities $Cx \leq c$ introduced in Sect. 5 admits a rational solution.

Farkas' lemma provides in polynomial time a vector $v \in \mathbb{Q}^n$ such that $Cv \leq c$. This vector is rounded up to a vector $w \in \mathbb{Z}^n$ satisfying the system of equalities $Ux = d$ by using the Smith Normal Form decomposition LSR of U (see Sect. 4):

$$w = R^{-1}[Rv]$$

where $[Rv]$ is the *integral part* of Rv, i.e. the unique vector in $\mathbb{Z}^n$ such that there exists a vector $\epsilon \in \mathbb{Q}^n$ satisfying $Rv = [Rv] + \epsilon$ and $-\frac{1}{2} < \epsilon_i \leq \frac{1}{2}$ for every i.

Lemma 7.1. *Vector w satisfies the system of equalities $Ux = d$.*

Intuitively w is "not so far" from v since $v = w + R^{-1}\epsilon$, and since v satisfies the system of inequalities $Cx \leq c$ it is quite possible that w also satisfies this system. Hence this vector is a good candidate for a valuation satisfying $\phi_A \wedge \phi_B$. Note that if

w does not satisfy this conjunction but it satisfies the more relaxed formula $\bar{\phi}_A \wedge \bar{\phi}_B$ obtained from $\phi_A \wedge \phi_B$ by removing the labels, we have discovered a solution to our original problem (labels are just used to prove the unsatisfiability). So let us assume that w is not a solution of $\bar{\phi}_A \wedge \bar{\phi}_B$. In this case, there exists a labelled predicate $(p)_I$ that occurs in $\phi_A \wedge \phi_B$ such that w does not satisfy p. We introduce the *pivot value* $\mu = u_p^t v$ for partitioning I into the following three disjoint intervals $I_\mu^<$, $I_\mu^=$, and $I_\mu^>$ where $I_\mu^\# = \{\delta \in I \mid \delta \# \mu\}$. We select the rational value μ for partitioning I since $\mu \in I$ (recall that v satisfies the system $Cx \leq c$). Note that the integral value $u_p^t w$ is not a good choice for partitioning I since in general this value is not in I. In particular w is just used to select a predicate p and its value is no longer used in the sequel.

The decomposition of I into $(I_\mu^<, I_\mu^=, I_\mu^>)$ should not be replaced by the partitions $(I_\mu^<, I_\mu^\geq)$ or $(I_\mu^\leq, I_\mu^>)$ since the termination of the algorithm is no longer guaranteed with these partitions. In fact the partition $(I_\mu^<, I_\mu^\geq)$ degenerates to $(\emptyset, I)$ if μ is the lower bound of I and the partition $(I_\mu^\leq, I_\mu^>)$ degenerates to $(I, \emptyset)$ if μ is the upper bound of I. Intuitively in these two cases the predicate $(p)_I$ is not really strengthened.

An interpolant ψ for (ϕ_A, ϕ_B) is deduced from interpolants $\psi^\#$ of $(\phi_A^\#, \phi_B^\#)$ for each $\# \in \{<, =, >\}$ by introducing the following formula:

$$\psi \;=\; \begin{cases} \psi^< \vee \psi^= \vee \psi^> & \text{if } (p)_I \text{ occurs in } \phi_A \\ \psi^< \wedge \psi^= \wedge \psi^> & \text{if } (p)_I \text{ occurs in } \phi_B \end{cases}$$

---

1 strengthening(ϕ_A , ϕ_B)
2 `let` $v \in \mathbb{Q}^n$ such that $Cv \leq c$
3 `let` $w = R^{-1}\lfloor Rv \rfloor$
4 `if` w satisfies $\bar{\phi}_A \wedge \bar{\phi}_B$ `return` w
5 `let` $(p)_I$ be a labelled predicate of $\phi_A \wedge \phi_B$ such that w does not satisfy p
6 `let` $\mu = u_p^t v$
7 `foreach` $\# \in \{<, =, >\}$
8 `let` $(\phi_A^\#, \phi_B^\#)$ obtained from (ϕ_A, ϕ_B) by replacing I by $I_\mu^\#$
9 `let` $\psi^\#$=interpolant($\phi_A^\#$, $\phi_B^\#$)
10 `if` the previous function returns a valuation w `return` w
11 `return` $\begin{cases} \psi^< \vee \psi^= \vee \psi^> & \texttt{if } (p)_I \text{ occurs in } \phi_A \\ \psi^< \wedge \psi^= \wedge \psi^> & \texttt{if } (p)_I \text{ occurs in } \phi_B \end{cases}$

Proposition 7.2. *When algorithm* interpolant*(ϕ_A , ϕ_B) terminates, it returns either an interpolant for (ϕ_A, ϕ_B) or a valuation $w \in \mathbb{Z}^n$ satisfying $\bar{\phi}_A \wedge \bar{\phi}_B$.*

8 Termination And Complexity

The exponential worst case execution time of interpolant is proved using a rooted tree that logs the algorithm execution. As expected a node N denotes a recursive sub-call of interpolant with input (ϕ_A^N, ϕ_B^N). Internal nodes N have three children denoted by $N_\#$ with $\# \in \{<, =, >\}$.

We first examine sub-algorithm strengthening(ϕ_A , ϕ_B) when the computed vector $v \in \mathbb{Q}^n$ is rounded up into an integer vector $w \in \mathbb{Z}^n$ that is not a solution of $\bar{\phi}_A \wedge \bar{\phi}_B$. We denote by $(p)_I$ a labelled predicate that occurs in ϕ_A or ϕ_B such that w does not satisfy p.

Lemma 8.1. *The set I contains at least two distinct integers.*

Recall that p is a predicate of the form $u_p^t x \#_p \beta_p$. The *distance* of the pivot value μ to β_p is bounded by the following lemma where $||z||_1 = |z_1| + \cdots + |z_n|$ for any vector $z = (z_1, \ldots, z_n)^t \in \mathbb{Z}^n$.

Lemma 8.2. *We have $|\mu - \beta_p| \leq \frac{1}{2} \left|\left|u_p^t R^{-1}\right|\right|_1$.*

We introduce an integer s denoting the size of the input problem, i.e. the number of bits to denote (ϕ_A, ϕ_B) with integral coefficients encoded in binary. Since the lines of the computed matrices U are vectors u_p for some predicates p, we deduce that the size of the matrix U is bounded by s. As the Smith Normal Form of a matrix U is obtained with a polynomial time algorithm, we deduce that there exists a polynomial P such that $\frac{1}{2} \left|\left|u_p^t R^{-1}\right|\right|_1 < 2^{P(s)}$ at any step of the computation. From the previous Lemma 8.2 we deduce that every pivot value μ satisfies $|\mu - \beta_p| < 2^{P(s)}$. Let us recall that the pivot value μ is used by sub-algorithm strengthening to partition I into three intervals $I_\mu^<$, $I_\mu^=$ and $I_\mu^>$. An immediate induction shows that every predicate p is labelled by an interval with integral bounds in $[\![\beta_p - 2^{P(s)}, \beta_p + 2^{P(s)}]\!]$. In particular the number of possible intervals I that label a predicate p is bounded by $(2 + 2^{P(s)+1})^2$. Let k denote the number of predicates. We have proved that the number of possible labelings is bounded by $(2 + 2^{P(s)+1})^{2k}$.

Lemma 8.3. *Intervals $I_\mu^<$, $I_\mu^=$ and $I_\mu^>$ are strictly included in I.*

Lemma 8.4. *Two distinct internal nodes have distinct labels.*

From the previous lemma we deduce that the number of internal nodes N is bounded by $(2 + 2^{P(s)+1})^{2k}$. As an internal node has at most three leaf children, we deduce that the number of nodes is bounded by $4(2 + 2^{P(s)+1})^{2k} = O(4^{Q(s)})$ where Q is the polynomial $Q(s) = 2s(P(s) + 1)$. We have proved the following theorem.

Theorem 8.5. *In exponential time in the worst case, algorithm* interpolant*(ϕ_A , ϕ_B) returns either a valuation satisfying $\bar{\phi}_A \wedge \bar{\phi}_B$ or an interpolant for (ϕ_A, ϕ_B).*

9 Experimental Evaluation

We have created a prototypical implementation of our interpolating decision procedure and integrated it as a theory solver into the SMT-solver OpenSMT [3], with the long-term goal of creating an interpolating SMT-solver to be used in model checkers. The prototype was developed on top of a recent development version of OpenSMT that already provided an interpolation procedure for propositional logic. In order to implement the algorithm check_inequality, we internally invoke the LP solver present in OpenSMT, which realizes the algorithm from [6].

To the best of our knowledge, the following tools and algorithms are the only ones available for comparison (also see Sect. 1):

		OpenSMT	SmtInterpol	iPrincess	Omega QE
Averest	10/9	10/1/31.75/ 90/221	8/4/97.02/ 72/149	0/0/–/ –/–	–/–/203.89/ 8/132639
CIRC/multiplier	16/1	5/1/48.94/ 45/2357	5/1/24.40/ 45/48827	6/1/130.46/ 35/12764	–/–/108.71/ 125/15392
CIRC/simplebitadder	17/0	7/0/102.81/ 63/23362	5/0/8.58/ 45/41077	6/0/412.82/ 49/47218	–/–/97.83/ 129/93181
check	4/1	4/1/0.77/ 36/1.7	2/1/0.17/ 18/2.3	4/1/36.65/ 33/485	–/–/0.26/ 30/0.67
nec-smt/small	17/18	1/0/251.95/ 9/36	7/0/259.86/ 63/1728	0/0/–/ –/–	–/–/134.88/ 66/15867
mathsat	100/21	74/15/52.96/ 666/2020	65/13/45.74/ 585/126705	11/11/61.78/ 99/13745	–/–/168.81/ 612/101088
rings	294/0	9/0/59.93/ 81/4611	0/0/–/ –/–	54/0/108.01/ 62/3470	–/–/227.25/ 1474/55307
wisa	2/3	0/0/–/ –/–	1/2/394.22/ 9/1039	0/0/–/ –/–	–/–/67.01/ 14/23709
	unsat/sat	*unsat / sat / average time / #interpolants / average int. size*			

Table 1. Results of applying the four compared tools to SMT-LIB benchmarks (times in seconds). Experiments were done on an Intel Xeon X5667 4-core machine with 3.07GHz, heap-space limited to 12GB, running Linux, with a timeout of 900s.

- the theorem prover *iPrincess* [2], which implements an interpolating decision procedure for QFPA based on a sequent calculus,
- the SMT-solver *SmtInterpol*,[3] a recently released interpolating decision procedure for linear integer arithmetic that uses an architecture similar to the one in *Foci* [11],
- *quantifier elimination (QE) procedures*, which can be used to generate interpolants as illustrated in Sect. 2; for our experiments, we use the implementation of the Omega test [14] available in *iPrincess*.

The benchmarks for our experiments are derived from different families of the SMT-LIB category QF-LIA. Some of the selected families (e.g., *rings*) are specifically designed to test integer reasoning capabilities, and contain problems satisfiable over rationals. Because SMT-LIB benchmarks are usually conjunctions at the outermost level, we partitioned them into $A \wedge B$ by choosing the first $\frac{k}{10} \cdot n$ of the benchmark conjuncts as A, the rest as B (where n is the total number of conjuncts, and $k \in \{1, \ldots, 9\}$). This yields 9 interpolation problems for each SMT-LIB benchmark.

Our experimental results are summarized in Table 1:[4]

- the number unsatisfiable/satisfiable problems tested, and the number of unsat/sat results that the tools were able to derive; in the remaining cases, either a timeout

[3] `http://swt.informatik.uni-freiburg.de/research/tools/smtinterpol`
[4] `http://www.philipp.ruemmer.org/interpolating-opensmt.shtml`

or a memory-out occurred. No figures are given for QE, which does not decide satisfiability of interpolation problems.
- the average time (in seconds) required to solve each benchmark, including the time for computing the 9 interpolants for a benchmarks. For QE, this is simply the average time to compute 9 interpolants.
- the total number of interpolants that could be computed. For OpenSMT and SmtInterpol, which compute interpolants on-the-fly while solving a problem, this is always $9\times$ the number of unsat results. iPrincess first constructs a proof for a problem, and afterwards extracts interpolants, which means that sometimes fewer than 9 interpolants can be computed (interpolant extraction has exponential complexity).
- the average size of generated interpolants, in terms of the number of equations, inequalities, and occurrences of propositional variables in the interpolant.[5]

Discussion. The experimental results show that our implementation in OpenSMT is competitive with all compared interpolation procedures: in 4 of the 8 families, it is able to prove the largest of problems unsatisfiable (and to compute interpolants for them); in all families but *CIRC/simplebitadder*, the runtime is smaller or comparable with the other tools; in 4 families, the generated interpolants are significantly smaller (on average) than the interpolants computed by the other tools.

QE is able to generate a large number of interpolants in the families *CIRC/multiplier*, *CIRC/simplebitadder*, and *rings*, albeit the generation is slow (on average) and the interpolants are large. It can be observed that our construction of interpolation problems by choosing arbitrary partitionings of SMT-LIB problems tends to generate many trivial interpolation problems, in the sense that the partition ϕ_A does not contain any local variables (or only few). On such interpolation problems, QE naturally performs very well; with an increasing number of local symbols, the performance of QE quickly degrades (also see [2] for a discussion of this phenomenon).

The complexity of interpolant extraction in iPrincess (which can be exponential due to mixed cuts) becomes visible in *rings*, where the prover can solve many more problems than the other systems, but can only produce a small number of interpolants.

Conclusion. We have presented an algorithm computing interpolants in the quantifier-free fragment of Presburger arithmetic in exponential time in the worst case. This algorithm combines the one presented in [7] that computes interpolants in polynomial time for systems of equalities over the integers and the one presented in [11] that computes interpolant in polynomial time for systems of inequalities over the rational numbers, without any overhead. In fact, sub-algorithm strengthening is called only if sub-algorithms check_equality and check_inequality fail in computing an interpolant.

Even though we limit the presentation to conjunctions of literals, following [11] the algorithm can be applied to any formula of the QFPA fragment. In the worst case this extended algorithm calls the presented algorithm for each conjunction of literals

[5] OpenSMT generates interpolants that use the SMT-LIB `flet` operator to achieve a more compact representation, as a result of how propositional interpolants are computed. Eliminating `flet`s can sometimes significantly increase the size of interpolants, but is practically not necessary for further processing, which is why `flet`s have been kept for our comparison.

extracted from ϕ_A and ϕ_B. In particular the worst case complexity is still exponential (we call an exponential number of times an exponential algorithm and $2^n 2^n = 4^n$). In particular our algorithm matches the exponential lower bound complexity.

We have created a prototypical implementation of our interpolating decision procedure. The experimental results show that our implementation is competitive with all compared interpolation procedures; work on further optimizations and further benchmarks is in progress. We are interested in applying interpolation to the verification of *safety properties* for *counter-systems*, a class of automata equipped with a finite set of counters (applications of these automata are given in [4]). More precisely, we plan to implement the combination of the *lazy-interpolation framework* [12] with the *acceleration framework* presented in [4] that requires an efficient interpolator for QFPA.

Acknowledgements. We would like to thank the OpenSMT team and Gérald Point for help with the implementation, Thomas Wahl for discussions, and the anonymous referees for helpful comments.

References

[1] Beyer, D., Zufferey, D., Majumdar, R.: CSIsat: Interpolation for LA+EUF. In: CAV. LNCS, vol. 5123, pp. 304–308. Springer (2008)
[2] Brillout, A., Kroening, D., Rümmer, P., Wahl, T.: An interpolating sequent calculus for quantifier-free Presburger arithmetic. In: IJCAR. LNCS, vol. 6173. Springer (2010)
[3] Bruttomesso, R., Pek, E., Sharygina, N., Tsitovich, A.: The OpenSMT solver. In: Esparza, J., Majumdar, R. (eds.) TACAS. LNCS, vol. 6015, pp. 150–153. Springer (2010)
[4] Caniart, N., Fleury, E., Leroux, J., Zeitoun, M.: Accelerating interpolation-based model-checking. In: TACAS. LNCS, vol. 4963, pp. 428–442. Springer (2008)
[5] Cimatti, A., Griggio, A., Sebastiani, R.: Interpolant generation for UTVPI. In: Schmidt, R.A. (ed.) CADE, LNCS, vol. 5663, pp. 167–182. Springer (2009)
[6] Dutertre, B., de Moura, L.M.: A fast linear-arithmetic solver for DPLL(T). In: CAV. LNCS, vol. 4144, pp. 81–94. Springer (2006)
[7] Jain, H., Clarke, E.M., Grumberg, O.: Efficient Craig interpolation for linear diophantine (dis)equations and linear modular equations. In: CAV. LNCS, Springer (2008)
[8] Kannan, R., Bachem, A.: Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. SIAM J. Comput. 8(4), 499–507 (1979)
[9] Lynch, C., Tang, Y.: Interpolants for linear arithmetic in SMT. In: ATVA. LNCS, Springer (2008)
[10] McMillan, K.L.: Applications of Craig interpolants in model checking. In: TACAS. LNCS, vol. 3440, pp. 1–12. Springer (2005)
[11] McMillan, K.L.: An interpolating theorem prover. Theor. Comput. Sci. 345(1) (2005)
[12] McMillan, K.L.: Lazy abstraction with interpolants. In: CAV. LNCS, Springer (2006)
[13] Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. J. Symb. Log. 62(3), 981–998 (1997)
[14] Pugh, W.: The Omega test: a fast and practical integer programming algorithm for dependence analysis. Communications of the ACM 8, 102–114 (1992)
[15] Rybalchenko, A., Sofronie-Stokkermans, V.: Constraint solving for interpolation. In: VMCAI. LNCS, vol. 4349, pp. 346–362. Springer (2007)
[16] Schrijver, A.: Theory of Linear and Integer Programming. Wiley (1986)
[17] Weispfenning, V.: Complexity and uniformity of elimination in Presburger arithmetic. In: ISSAC. pp. 48–53 (1997)