Craig Interpolation for Quantifier-Free Presburger Arithmetic*

Angelo Brillout[1], Daniel Kroening[2], and Thomas Wahl[1]

[1] Computer Systems Institute, ETH Zurich
[2] Computing Laboratory, Oxford University

Abstract. *Craig interpolation* has become a versatile algorithmic tool for improving software verification. Interpolants can, for instance, accelerate the convergence of fixpoint computations for infinite-state systems. They also help improve the refinement of iteratively computed lazy abstractions. Efficient interpolation procedures have been presented only for a few theories. In this paper, we introduce a complete interpolation method for the full range of *quantifier-free Presburger arithmetic* formulas. We propose a novel convex variable projection for integer inequalities and a technique to combine them with equalities. The derivation of the interpolant has complexity low-degree polynomial in the size of the refutation proof and is typically fast in practice.

1 Introduction

A *Craig interpolant*, or simply *interpolant*, for an inconsistent pair of formulas A and B is a formula I that is implied by A, inconsistent with B, and contains only variables occurring in both A and B [1]. In other words, a Craig interpolant is weaker than A, but still strong enough to be inconsistent with B, and therefore provides an "explanation" of the inconsistency in terms of the common variables. In his original theorem, Craig showed that an interpolant exists for any two inconsistent first-order formulas A and B.

Craig interpolants have proven to be useful in many areas. McMillan suggested to use them in an over-approximating image operator [2], which has led to a considerable advance in SAT-based model checking. For infinite-state systems, interpolants can significantly improve the refinement step in lazy predicate abstraction [3]. Methods to efficiently compute interpolants are known for propositional logic and linear arithmetic over the reals with uninterpreted functions [4, 5]. For these theories, an interpolant can be derived in linear time from a deductive proof of inconsistency of A and B.

Presburger arithmetic is a popular theory for modeling computer systems, for example to describe the behavior of infinite-state programs [6]. It was shown to be decidable by quantifier elimination [7], which is, however, of double-exponential complexity. Fortunately, formulas arising in system specification and verification

* This research is supported by an award from IBM Research and by the European Union as part of the FP7-STREP MOGENTES project.

are mainly quantifier-free [8, 9]. In this paper we therefore focus on *quantifier-free Presburger arithmetic* (QFP). An interpolant between two inconsistent QFP formulas A and B can be computed by existentially quantifying the variables that occur only in A, followed by quantifier elimination. This approach is, however, prohibitively expensive.

Contribution In this paper, we propose an algorithm that extracts an interpolant directly from a proof of inconsistency of A and B. Our algorithm extends the framework of Pugh's *Omega test* [10]. We present suitable deduction steps in the form of inference rules. Following a suggestion by McMillan [5], the rules are augmented with *partial interpolants*—expressions that are transformed step by step to yield an interpolant of the initial formulas A and B once a contradiction has been reached. We present our algorithm for conjunctions of equalities and inequalities; interpolants for an arbitrary Boolean skeleton can be obtained using the framework described in [5].

For conjunctions of equalities, our algorithm exploits the fact that *exact* variable projection is efficient for certain fragments of QFP. We therefore treat equalities separately in the first part of the paper and describe such a projection procedure. Our procedure supports *stride constraints*, i.e., quantified equalities expressing divisibility relationships. For conjunctions of inequalities, we show that deriving an interpolant requires the *strongest convex* projection (which may be inexact) and give an efficient algorithm for computing this projection. Finally, we describe the first interpolation method that combines conjunctions of integer equality and inequality constraints.

Related work For propositional logic, several interpolation methods have been presented [4, 2, 11]. In addition to the work by McMillan [5], Rybalchenko et al. propose an algorithm for linear arithmetic over the reals with uninterpreted functions that circumvents the need for an explicit proof [12]. For integer arithmetic, McMillan considers the logic of difference-bound constraints [3]. This logic, a fragment of QFP, is decidable by reducing it to arithmetic over the reals. Difference-bound constraints are, however, not sufficient to express many typical program constructs, such as integer divisibility [8].

For interpolating SMT (satisfiability modulo theory) solvers, which involve calls to theory-specific provers, combination frameworks have been presented in [13, 14]. In [15], an SMT solver is used to derive interpolants for rational linear arithmetic with uninterpreted functions. In [16], separate interpolation procedures for two theories are presented, namely (i) QFP restricted to conjunctions of integer linear (dis)equalities and (ii) QFP restricted to conjunctions of stride constraints. The combination of both fragments with integer linear inequalities is, however, not supported. Our work closes this gap, as it permits predicates involving all types of constraints. Such predicates arise naturally for instance in inductive invariant discovery, as argued in [16].

Outline This paper is organized as follows. Section 2 contains background and terminology. In section 3, we present the rules for computing interpolants of inconsistent equality and stride constraints. Section 4 does the same for inequality constraints and for combinations of both. In section 5.2, we discuss the time complexity of our algorithm.

2 Preliminaries

2.1 Craig Interpolants

Two QFP formulas are *inconsistent* if their conjunction is unsatisfiable. We define $\mathcal{V}(\phi)$ to be the set of variables occurring in a (quantifier-free) formula ϕ. For any two formulas A and B, we write $\mathcal{L}_A$ for the set of variables *local* to A, i.e., $\mathcal{L}_A = \mathcal{V}(A) \setminus \mathcal{V}(B)$. Analogously, we write $\mathcal{G}$ for the *global* (common) variables of A and B, i.e., $\mathcal{G} = \mathcal{V}(A) \cap \mathcal{V}(B)$. The quantifier-free formulas A and B are *equisatisfiable*, denoted $A \stackrel{\circ}{\equiv} B$, if existentially quantifying their respective local variables produces two logically equivalent formulas, i.e., $\exists \mathcal{L}_A.A \equiv \exists \mathcal{L}_B.B$. Let $\perp$ and $\top$ represent the Boolean values *false* and *true*, respectively.

Definition 1. *A* (Craig) Interpolant *for two inconsistent quantifier-free formulas* (A, B) *is a formula* I *such that:*

(1) $A \models I$*,*
(2) $(B, I) \models \perp$*, and*
(3) $\mathcal{V}(I) \subseteq \mathcal{G}$*.*

As an example, let A and B be the (inconsistent) formulas $x = y + 1 \wedge z = y$ and $x = y$, respectively. An example of an interpolant I for A and B is $x = y + 1$.

2.2 Quantifier-free Presburger Arithmetic

Presburger arithmetic is the first-order theory defined by the structure $\langle \mathbb{Z}, =, \leq, + \rangle$, i.e., quantified linear *integer* arithmetic with arbitrary Boolean connectives. In 1929, M. Presburger presented a quantifier elimination procedure for this logic, which gives rise to a decision procedure [7].

We consider in this paper *quantifier-free* Presburger arithmetic with stride predicates, denoted QFP. Atoms, henceforth called *constraints*, are of the form

$$t \bowtie 0 \quad (\bowtie \in \{=, \leq\}) \quad \text{or} \quad d \mid t \quad (d \in \mathbb{N}_{\geq 2}),$$

where t is a term of the form $\sum_{j \in J} a_j x_j + c$. We call these atoms *equality*, *inequality* and *stride* constraints.

The stride predicates $d \mid t$ specify divisibility properties of a term t, e.g., $2 \mid x$ denotes that x is even. We refer to d as the *periodicity* of a stride constraint. To motivate the need for stride predicates, consider the equalities $x - 2y = 0$ and $x - 2z - 1 = 0$, whose only quantifier-free interpolant is $2 \mid x$ [5].

We say that two constraints $\sum_{j \in J} a_j x_j + c \bowtie 0$ and $\sum_{j \in J} b_j x_j + d \bowtie 0$ are *parallel* if for every $j \in J$, $a_j = b_j$ or for every $j \in J$, $a_j = -b_j$. A *unit coefficient* is a coefficient a_j with $|a_j| = 1$.

QFP formulas are constructed using the usual Boolean connectives. We adopt the method in [5] to reduce reasoning over arbitrary Boolean combinations of constraints to reasoning over conjunctions. Despite the stipulation of being quantifier-free, we permit a restricted form of quantification in QFP, namely over *finite* sets of integers. Formulas containing such quantifications are semantically quantifier-free since they can be rewritten using a finite disjunction.

2.3 Equisatisfiability-Preserving Manipulations

Tightening of inequalities Let $g := gcd(\{|a_j| : j \in J\})$ be the greatest common divisor of the coefficients in the term $t = \sum_{j \in J} a_j x_j + c$ of an inequality $t \leq 0$. We say the inequality is *tight* if g divides c. Every inequality can be transformed into an equivalent tight form by replacing c with $g\lceil \frac{c}{g} \rceil$. We refer to $\mathcal{T}(f)$ as the tight form of an inequality f. (Note that an equality constraint $t = 0$ is unsatisfiable if g does not divide c.)

Homogenization Let $Q(x)$ be a formula over x. We *homogenize* $Q(x)$ by computing an equisatisfiable formula $F(\sigma)$ over a new variable σ (but without x) such that all coefficients of σ are unit coefficients. This is achieved as follows:

1. Compute the least common multiple $l := lcm\{|a| : \ a \text{ is a coefficient of } x \text{ in some constraint}\}$.
2. Multiply each constraint over a term containing a multiple ax of x by $\frac{l}{|a|}$; for a stride constraint $d \mid t$ this means to multiply both d and t by $\frac{l}{|a|}$. The result is a formula $Q'(x)$ equivalent to $Q(x)$ where all coefficients of x are either l or $-l$.
3. Replace every occurrence of lx in $Q'(x)$ with a new variable σ and conjoin the result with the new constraint $l \mid \sigma$.

The obtained formula $F(\sigma)$ and the original $Q(x)$ are equisatisfiable, with σ having unit coefficients everywhere, as shown by Cooper [17]. A formula is called *σ-homogenized* if all occurrences of σ have unit coefficients.

Exact projection We define a projection method that is based on [17]; our method is simpler since it assumes an x-homogenized conjunction $Q(x)$ of constraints containing at most one inequality. *Exact projection* amounts to eliminating x from $Q(x)$, resulting in an equisatisfiable formula. We distinguish two cases:

- If there is at least one equality containing x in $Q(x)$, let eq be any such equality. Since every occurrence of x has a unit coefficient, eq can be rewritten as $x = t$. Now obtain a new, equisatisfiable formula $Q'(t)$ by dropping the conjunct eq from $Q(x)$ and replacing x by t everywhere else.
- Otherwise, let $l := lcm\{d : d \text{ is a periodicity of some stride constraint containing } x\}$. Remove any inequality over x from $Q(x)$ resulting in a $Q'(x)$. Eliminate x by replacing $Q'(x)$ with $\exists i \in \{0, \ldots, l\}.Q'(i)$. The result is equisatisfiable to $Q(x)$.

We denote by $proj(Q(x), x)$ a procedure that first x-homogenizes $Q(x)$ and then returns an equisatisfiable formula by exact projection. We extend this procedure to act on a formula Q and a *set* of variables V, denoted $proj(Q, V)$, by applying $proj$ to $(Q(x), x)$ for all $x \in V$ in any order.

3 Equality and Stride Constraints

In this section, we present an algorithm for deriving an interpolant for two inconsistent formulas A and B that are conjunctions of stride and equality constraints. The algorithm is based on an elimination procedure for equality and stride constraints (section 3.1). The procedure is refined in section 3.2 by annotating its steps with partial interpolants.

3.1 Eliminating Equality and Stride Constraints

We use an algorithm proposed by Pugh [10] for eliminating the equalities from the system of constraints. For this purpose, we need a slightly modified "centered" modulus function $\widehat{\text{mod}}$, defined as $a \mathbin{\widehat{\text{mod}}} b := a - b\lfloor \frac{a}{b} + \frac{1}{2} \rfloor$. We write $t \mathbin{\widehat{\text{mod}}} b$ to denote $\sum_{i \in J}(a_i \mathbin{\widehat{\text{mod}}} b)x_i + (c \mathbin{\widehat{\text{mod}}} b)$ for a term t of the form $\sum_{i \in J} a_i x_i + c$. This follows from distributivity of $\widehat{\text{mod}}$.

The elimination algorithm first replaces each stride constraint $d \mid t$ by the equisatisfiable equality $d\sigma + t = 0$, where σ is a fresh variable. What remains is a system of equalities. Consider the following equality involving variable x:

$$ax + t = 0\,. \tag{1}$$

If x has a unit coefficient, we can eliminate the equality by deleting it from the system and replacing every occurrence of x by $-at$. Otherwise, by applying the $\widehat{\text{mod}}$ operator to both sides of equality (1) and introducing a fresh variable σ, we obtain the new constraint

$$\left(a \mathbin{\widehat{\text{mod}}} m\right) x + \left(t \mathbin{\widehat{\text{mod}}} m\right) = m\sigma \tag{2}$$

where $m = |a| + 1$. Since $a \mathbin{\widehat{\text{mod}}} m = -sign(a)$, variable x in (2) has a unit coefficient. Thus, we can eliminate x in (1) and in all other constraints involving x. As shown in [10], the absolute values of the coefficients in the new equality resulting from (1) have decreased, eventually resulting in an equality with a unit coefficient. This equality can be eliminated without applying the $\widehat{\text{mod}}$ operator.

We call the original constraint (1), which is used to derive a constraint with a unit coefficient, the *pivot equality*, denoted eq_p. Let ϕ be a conjunction of equalities. We denote by $elim(\phi)$ the procedure that eliminates all equalities in ϕ using pivot equalities eq_p chosen according to some heuristics – we refer the reader to [10] for such a heuristic. Note that each elimination of an equality leaves the remaining system equisatisfiable to the original one. Therefore, if the procedure ever encounters an unsatisfiable equality, it immediately returns $\bot$, indicating inconsistency of the original constraints. Otherwise, the original system is eventually reduced to an equality of the form $c = c$ for some constant c; the procedure returns $\top$. Note that, since we assume A and B to be inconsistent, *elim* never returns $\top$ unless we consider combinations of equalities and inequalities and the inconsistency is due to the inequalities (section 5).

3.2 Interpolation for Equality and Stride Constraints

The first part of our contribution follows. We introduce rules in order to derive an interpolant from a proof of inconsistency of the linear equality formulas A and B. To do so, we borrow the notion of a *partial interpolant* from [5].

Definition 2. *A* partial equality interpolant for (A, B) *is a conjunction of linear equalities* ϕ^A *such that:*

(1) $A \models \phi^A$*, and*
(2) $(B, \phi^A) \models \phi$*, and*
(3) *if* ϕ *contains an unsatisfiable equality, then* $\mathcal{V}(\phi^A) \subseteq \mathcal{G}$*.*

where A, B and ϕ *are conjunctions of equalities. We write* $(A, B) \vdash \phi\ [\phi^A]$ *if we can* derive *the partial interpolant* ϕ^A *from* (A, B)*.*

Observe that if $\phi \equiv \bot$, definitions 1 and 2 coincide, with ϕ^A as the interpolant.

Consider now a proof of inconsistency of the two conjunctions A and B of equalities. The proof consists of a sequence of proof rule applications. We extend these rules to apply to partial interpolants that are attached to antecedent and consequent of the rules. The partial interpolants are transformed to eventually result in an interpolant for (A, B). We first present a rule to introduce hypotheses and the corresponding partial interpolant for (A, B) in the proof tree.

$$\textsc{HypEq}\frac{}{(A, B) \vdash A \wedge B\ [A]}$$

The partial interpolant is simply A. Note that HypEq introduces all equalities simultaneously. The soundness proof for this rule, showing that the derived partial equality interpolant conforms to the three conditions of definition 2, is straightforward.

The next rule eliminates the equality constraints as mentioned in section 3.1. The rule results in a partial interpolant where A is projected by elimination of the variables local to A:

$$\textsc{ElimEq}\frac{(A, B) \vdash \qquad A \wedge B \quad [A]}{(A, B) \vdash elim(A \wedge B)\ [proj(A, \mathcal{L}_A)]}$$

If function $elim(A \wedge B)$ returns $\bot$, the (final) interpolant is $proj(A, \mathcal{L}_A)$. Note that, in this interpolant, every variable local to A has been eliminated by *proj* and that no new variable has been introduced.

Soundness (of ElimEq). To show the soundness of the rule, we argue that the rule *preserves* the three conditions of definition 2. Regarding the first condition, the fact that $A \models proj(A, \mathcal{L}_A)$ follows immediately from the soundness of Cooper's projection procedure. Since $A \wedge B \stackrel{\circ}{\equiv} B \wedge proj(A, \mathcal{L}_A) \stackrel{\circ}{\equiv} elim(A \wedge B)$, we know that $(B, proj(A, \mathcal{L}_A)) \models elim(A \wedge B)$. This shows condition 2. The *proj* procedure eliminates every local variable to A and thus $\mathcal{V}(proj(A, \mathcal{L}_A)) \subseteq \mathcal{G}$. This shows condition 3. □

Example 1. We would like to find an interpolant for $A \equiv (6 \mid 3z - 2y - 2)$ and $B \equiv (6x - y = 0)$. Using the HypEq rule, we introduce both constraints and the partial interpolant. We apply the ElimEq rule to the result:

$$\textsc{ElimEq}\frac{(A, B) \vdash 6\sigma + 3z - 2y - 2 = 0 \wedge 6x - y = 0\ [6 \mid 3z - 2y - 2]}{(A, B) \vdash 6\sigma - 12x - 2 = 0\ [\exists i \in \{0 \ldots 6\}.\ (6 \mid i - 2y - 2) \wedge (3 \mid i)]}$$

We eliminate y by applying *elim*, since y has a unit coefficient in $6x - y = 0$. However, the substitution of $6x$ for y produces a contradiction since $gcd(6, 12)$ does not divide 2. We project the partial interpolant by eliminating the only local variable $z \in \mathcal{L}_A$. To do so, *proj* z-homogenizes the partial interpolant, resulting in $(6 \mid \sigma - 2y - 2) \wedge (3 \mid \sigma)$ and finally in the interpolant $\exists i \in \{0, \ldots, 6\}.\ (6 \mid i - 2y - 2) \wedge (3 \mid i)$. □

4 Inequality Constraints

This section presents a method for deriving an interpolant for two inconsistent formulas A and B that are conjunctions of inequalities. We first review the variable elimination procedure used in the Omega test (Section 4.1). We then introduce the notion of strongest convex projection (Section 4.2), which is necessary to refine the procedure with partial interpolants (Section 4.3).

4.1 Fourier-Motzkin variable elimination for QFP

W. Pugh adapted the *Fourier-Motzkin* (FM) variable elimination method to QFP [10]. This section briefly reviews this method. In the following, t_1 and t_2 are two terms not containing the variable x, and a, b are positive integers. Consider the two inequalities

$$ax + t_1 \leq 0 \quad \text{and} \quad -bx + t_2 \leq 0\ . \tag{3}$$

These inequalities are upper (left constraint) and lower (right constraint) bounds on x. Equivalently, we get

$$at_2 \leq abx \leq -bt_1 \tag{4}$$

by multiplying the upper and lower bounds by b and a, respectively. The FM method eliminates variable x by deducing the following inequality from (4):

$$\mathcal{T}(at_2 + bt_1 \leq 0) \tag{5}$$

where $\mathcal{T}(at_2 + bt_1 \leq 0)$ denotes the tight form of $at_2 + bt_1 \leq 0$. Inequality (5) is a projection of (4) that eliminates x. Note that (4) implies (5), but not generally vice versa: the two are not equisatisfiable. We therefore speak of an *inexact* projection. If the distance between the upper and the lower bound is less than ab, there may or may not be a solution to the following equation:

$$\mathcal{T}(-ab + 1 \leq at_2 + bt_1 \leq 0)\,. \tag{6}$$

Note that in (6), the strict inequality $-ab < at_2 + bt_1$ has been replaced by the equivalent inequality $-ab + 1 \leq at_2 + bt_1$. In geometrical terms, (6) describes the "thin" part of the polyhedron (4). If no inconsistency is found by (inexact) projection of all inequalities, i.e., only inequalities of the form $-p \leq 0$, $p \in \mathbb{N}_{\geq 1}$ remain, one must check for solutions in this "thin" part.

For this purpose, Pugh introduced *splinters*. Given are the bounds (3) leading to inexact projection. An equality $-bx + t_2 + i = 0$ is added to the original set of inequalities. This equality is eliminated as explained in section 3.1 and the FM algorithm is called recursively. This is done for each $i \in \{0, \ldots, s\}$ where $s = \lfloor (|nb| - |n| - b)/|n| \rfloor$ and n is the negative coefficient of x with the largest absolute value in any inequality. If all splinters of all inexact projections produce an inconsistency, then the original system of inequalities is unsatisfiable. We refer the reader to [10] for further details.

4.2 Strongest Convex Projection

Consider the case that an inconsistency is reached without the need for splinters, i.e., inexact projection is sufficient to show inconsistency of (A, B). Since the inexact projection (5), being a single inequality, describes a *convex* region, there is also a convex interpolant. In order to compute it, we introduce the notion of *strongest convex* projection, i.e., the *strongest* projection expressible with *one* inequality. Formally, we introduce:

Definition 3. *For lower and upper bounds $ax + t_1 \leq 0$ and $-bx + t_2 \leq 0$, let $t' \leq 0$ be the tight form of $at_2 + bt_1 \leq 0$, and let $m \geq 0$. Inequality $t' + m \leq 0$ is the* strongest convex projection *of these bounds if there is no integer i such that:*

$$(at_2 \leq abx \leq -bt_1) \models (t' + i \leq 0) \models (t' + m \leq 0)\,.$$

We now present a new method to compute the strongest convex projection of a lower and an upper bound; see algorithm 1. The bounds are converted into the inequality $-ab + 1 \leq at_2 + bt_1 \leq 0$. Tightening this inequality results in a constraint of the form $-c' \leq t' \leq 0$, which can equivalently be expressed as the quantifier-free formula $\exists i \in \{-c', \ldots, 0\}.t' = i$. This is our pivot equality (line 1). This equality, conjoined with the lower and upper bounds, can be checked for satisfiability, thus revealing which integers i are feasible in the "thin" part of the polyhedron (4). We perform this check in line 2 using an elimination procedure modified from section 3.1: we find an equality with a unit coefficient, rewrite it into the form $y = t_y$ and replace every occurrence of y in the inequality and in

eq_p by t_y. This is repeated until the pivot equality eq_p is reduced to $\top$, resulting in the bounds of the form as given by f_1 and f_2 in line 2. Note that the constants c and d depend on i.

Algorithm 1 Strongest convex projection

Input: lower bound $ax + t_1 \leq 0$, upper bound $-bx + t_2 \leq 0$
Output: strongest convex projection of these bounds
1: **let** $eq_p \;=\; (\exists i \in \{-c', \ldots, 0\}.\, t' = i)$ // tight form of $-ab + 1 \leq at_2 + bt_1 \leq 0$
2: **let** $f_1 = (t'_1 + c(i) \leq 0)$ and $f_2 = (t'_2 + d(i) \leq 0)$ be the tight inequalities resulting from reducing eq_p to $\top$
3: **if** $t'_1 = t'_2$ or f_1 and f_2 are not parallel **then**
4:     **return** $t' \leq 0$
5: **else** // $t'_1 = -t'_2$
6:     **let** $A = \{i : -c' \leq i \leq 0 \;\wedge\; c(i) + d(i) \leq 0\}$
7:     **if** $A \neq \emptyset$ **then**
8:         **return** $t' - (\min A) \leq 0$
9:     **else**
10:         **return** $t' - c' + 1 \leq 0$

We demonstrate some aspects of algorithm 1 with the following example.

Example 2. Suppose the following bounds are given:

$$x + 3y - 2 \leq 0 \;\wedge\; x - 3y + 1 \leq 0\,. \tag{7}$$

In line 1, the algorithm tightens the "thin" part of the projection which is $-8 \leq 6x - 3 \leq 0$. The result is $6x = 0$, i.e., here $c' = 0$ and $i = 0$. In line 2, this equality is substituted into (7); tightening produces two inequalities $3y \leq 0$ and $-3y + 3 \leq 0$. These are parallel with unequal terms (case $t'_1 = -t'_2$, line 5). Since $A = \emptyset$, the strongest convex projection $6x + 1 \leq 0$ is returned in line 10. □

In the following section, we continue example 2 and demonstrate why the notion of strongest convex projection is necessary for deriving partial interpolants.

4.3 Interpolation for Inequality Constraints

The notion of partial interpolants for inequalities is defined as follows.

Definition 4. *A* partial inequality interpolant for (A, B) *is an inequality* $t^A \leq 0$ *such that:*

(1) $A \models t^A \leq 0$,
(2) $B \models t - t^A \leq 0$, *and*
(3) $\mathcal{V}(t^A \leq 0) \subseteq \mathcal{V}(A)$ *and* $\mathcal{V}(t - t^A) \subseteq \mathcal{V}(B)$.

where A, B are conjunctions of inequalities and t, t^A terms. We write $(A, B) \vdash t \leq 0\ [t^A \leq 0]$ *if we can* derive *the partial interpolant $t^A \leq 0$ from (A, B).*

Observe that if t is a positive constant, $t - t^A \leq 0$ is a contradiction and $t^A \leq 0$ is an interpolant for (A, B).

We now present rules that implement the FM elimination procedure for QFP. As with equalities, these rules also compute partial interpolants that preserve the properties of definition 4. When introducing a hypothesis in the proof, the partial interpolant depends on the origin of the hypotheses:

$$\textsc{HypIn}\frac{}{(A,B) \vdash t \leq 0\ [\chi_A(t \leq 0)]}\ (t \leq 0) \in (A, B)$$

where $\chi_A(t \leq 0)$ is defined to be $t \leq 0$ if $t \leq 0 \in A$ and $0 \leq 0$ otherwise.

The next rule projects inequalities. When combing two inequalities to achieve projection, the same linear combination is applied to the partial interpolants.

$$\textsc{Proj}\frac{\begin{array}{ll}(A,B) \vdash & ax + t_1 \leq 0\ \ [t_1^A \leq 0] \\ (A,B) \vdash & -bx + t_2 \leq 0\ \ [t_2^A \leq 0]\end{array}}{(A,B) \vdash \mathcal{T}(at_2 + bt_1 \leq 0)\ [\mathcal{T}(at_2^A + bt_1^A + m \leq 0)]}\, a, b \in \mathbb{N}_{\geq 1}$$

where m is a constant such that if inexact projection occurs, $\mathcal{T}(t_2^A + t_1^A + m \leq 0)$ is the strongest convex projection, and otherwise $m = 0$.

Soundness (of Proj*).* We check if the conditions of definition 4 are preserved. Condition 1 is straightforward. From the premises, we know that $B \models ax + t_1 - t_1^A \leq 0$ and $B \models -bx + t_2 - t_2^A \leq 0$ and thus $B \models (at_2 + bt_1) - (at_2^A + bt_1^A) \leq 0$, which is convex. Tightening a constraint only increases the constant c of the corresponding inequality and since the projection (if any) of the partial interpolant is the *strongest*, we conclude $B \models \mathcal{T}(at_2 + bt_1) - \mathcal{T}(at_2^A + bt_1^A + m)$. This proves condition 2. The fact that $\mathcal{T}$ does not change the coefficients in a constraint emphasizes the similarity with the linear arithmetic method described in [5]. As in this work, projection by eliminating a variable x also eliminates x in the partial interpolant. Since every variable has to be eliminated to obtain an inconsistency, the interpolant does not contain any local variable. This shows condition 3. □

Example 3. We show how to derive an interpolant for $A \equiv x+3y-2 \leq 0 \wedge x-3y+1 \leq 0$ and $B \equiv -x \leq 0$. We write $t \leq 0\ [t^A \leq 0]$ instead of $(A, B) \vdash t \leq 0\ [t^A \leq 0]$ and we do not show how to introduce hypotheses to save space. First, we project the two inequalities from A by eliminating x:

$$\textsc{Proj}\ \frac{\begin{array}{r}-x \leq 0\ [0 \leq 0] \\ x + 3y - 2 \leq 0\ [x + 3y - 2 \leq 0]\end{array}}{3y \leq 0\ [x + 3y - 2 \leq 0]} \qquad \textsc{Proj}\ \frac{\begin{array}{r}-x \leq 0\ [0 \leq 0] \\ x - 3y + 1 \leq 0\ [x - 3y + 1 \leq 0]\end{array}}{-3y + 1 \leq 0\ [x - 3y + 1 \leq 0]}$$

We can now derive a contradiction by eliminating y:

$$\textsc{Proj}\ \frac{\begin{array}{r}3y \leq 0\ [x + 3y + 1 \leq 0] \\ -3y + 1 \leq 0\ [x - 3y - 2 \leq 0]\end{array}}{3 \leq 0\ [6x + 1 \leq 0]}$$

Note that the interpolant $6x + 1 \leq 0$ is the strongest convex projection of $x+3y+1 \leq 0$ and $x-3y-2 \leq 0$, which was computed in the example at the end of section 4.2. Observe that the standard projection of the partial interpolant according to equation (5) is $6x \leq 0$, which does not interpolate (A, B). □

Splinters If the FM procedure for QFP introduces splinters as described in section 4.1, the Omega test is called recursively for each splinter. More precisely, in case of an inexact projection when applying the PROJ rule, the interpolation algorithm is called upon each pair $(A, B)_i$ and $(A, B)_{\geq s+1}$ defined as $(A \wedge t_2^A + i = 0, B \wedge t_2 - t_2^A = 0)$ and $(A \wedge t_2^A + s + 1 \leq 0, B \wedge t_2 - t_2^A \leq 0)$, respectively.

If all splinters produce an inconsistency, i.e., all pairs $(A, B)_i$ and $(A, B)_{\geq s+1}$ are inconsistent, the original system is unsatisfiable. We can construct an interpolant for (A, B) from the respective interpolants I_i and $I_{\geq s+1}$ for $(A, B)_i$ and $(A, B)_{\geq s+1}$ as follows:

$$\text{SPLIN}\frac{\begin{array}{ll}(A,B)_{\geq s+1} & \vdash \bot\ [I_{\geq s+1}] \\ (A,B)_i & \vdash \bot\ [I_i]\end{array}}{\begin{array}{ll}(A,B) & \vdash \bot\ [\vee_i^s\ I_i \vee I_{\geq s+1}]\end{array}} \quad \text{for all } i \in \{0, \ldots, s\}$$

Soundness (of SPLIN*).* We show that the derived interpolant conforms to definition 1. We denote by A_i, B_i, $A_{\geq s+1}$ and $B_{\geq s+1}$ the respective components of the pairs $(A, B)_i$ and $(A, B)_{\geq s+1}$. Condition 1 follows from $A_i \models I_i$ and $\vee_i^s A_i \vee A_{\geq s+1} \equiv A$ follows condition 1, i.e., $A \models \vee_i A_i \vee A_{\geq s+1}$. Condition 2 follows from the unsatisfiability of $\vee_i^s A_i \wedge B_i \wedge A_{\geq s+1} \wedge B_{\geq s+1}$ and $\vee_i B_i \vee B_{\geq s+1} \equiv B$. Finally, since $\mathcal{V}(A_i) = \mathcal{V}(A)$ and $\mathcal{V}(A_{\geq s+1}) = \mathcal{V}(A)$ condition 3 also holds.

Example 4. In example 3 we first eliminate x and then y by using PROJ. Note that if we reverse this order, no inconsistency is reached by using PROJ only. This is due to the inexact projection of $x + 3y - 2 \leq 0$ and $x - 3y + 1 \leq 0$ by elimination of y. In this case, the number of splinters that must be derived is given by $s = (3 * 3 - 3 - 3)/3 = 1$.

The Omega test is then called recursively for each pair $(A, B)_i$, $i \in \{0, 1\}$ and $(A, B)_{\geq 2}$ given by $(A \wedge x - 3y + 1 + i = 0, B)$ and $(A \wedge x - 3y + 1 + 3 \leq 0, B)$, respectively. The pairs $(A, B)_i$ contain both inequalities and equalities. In the next section, we show how to derive an interpolant for $(A, B)_1$. Once each pair $(A, B)_i$ and $(A, B)_{\geq 2}$ has been proved inconsistent, we combine their respective interpolants with the SPLIN rule:

$$\text{SPLIN}\ \frac{\begin{array}{ll}(A,B)_0 & \vdash \bot\ [2x \leq 0 \wedge 3 \mid (x+1)] \\ (A,B)_1 & \vdash \bot\ [2x \leq 0 \wedge 3 \mid (x+2)] \\ (A,B)_{\geq 2} & \vdash \bot\ [6x+1 \leq 0]\end{array}}{(A,B) \vdash \bot\ [(6x+1 \leq 0) \vee (2x \leq 0 \wedge 3 \mid x+2) \vee (2x \leq 0 \wedge 3 \mid x+1)]}$$

Note that the result is indeed an interpolant for (A, B). □

5 Putting it all together

5.1 Combining equality and stride constraints with inequalities

We now turn to the most challenging part of this work, namely deriving an interpolant for two inconsistent formulas A and B that are conjunctions of equality, inequality and stride constraints. In this section, we denote by E_A and E_B the conjunction of equality and stride constraints of A and B, respectively. In order to detect any inconsistency, the Omega test begins by eliminating stride and equality constraints from the system, i.e., the HypEq and ElimEq rules are applied to the pair (E_A, E_B). We distinguish two cases:

(i) Suppose an unsatisfiable equality was found during the elimination. In this case, $E_I := proj(E_A, \mathcal{L}(E_A))$ is an interpolant for (E_A, E_B). From the validity of $A \models E_A$, $B \models E_B$ and $\mathcal{L}(E_A) \subseteq \mathcal{L}(A)$, it follows that E_I also an interpolant for (A, B). That is, we derive an interpolant for (A, B) using the HypEq and ElimEq rules only, *without* considering the inequalities at all.

(ii) Otherwise, all equality and stride constraints are successfully eliminated. In this case, for each $x = t_u$ derived with the $\widehat{\text{mod}}$ operator, the Omega test replaces each occurrence of x in *every* constraint of A and B, not only in the equalities. Eventually, a new pair (A', B') consisting only of inequalities remains. The formula $A' \wedge B'$ is then equisatisifibale to $A \wedge B$.

To formalize the second case, we denote by $\phi\{x \leftarrow t_u\}$ the result of substituting the term t_u for every occurrence of variable x in ϕ. By $\phi\{\boldsymbol{x} \leftarrow \boldsymbol{t_u}\}$ we denote the *sequence* of substitutions performed, in this order, during the equality elimination process. The formulas A' and B' are then given by $A\{\boldsymbol{x} \leftarrow \boldsymbol{t_u}\}$ and $B\{\boldsymbol{x} \leftarrow \boldsymbol{t_u}\}$, respectively.

We can now derive new partial interpolants for (A', B') using the HypIn, Proj and SplIn rules, with A' and B' in place of A and B. Once a contradiction is reached, the obtained interpolant will be valid for (A', B'), but not for (A, B). More precisely, since the terms $\boldsymbol{t}_u$ may contain new variables, the generated interpolant may also contain a variable not occurring in (A, B). The problem is to map an interpolant for (A', B') to an interpolant for (A, B).

We address this problem as follows. Let $t^{A'} \leq 0$ be an interpolant for (A', B'). We show below how to compute a partial interpolant $t^A \leq 0$ for (A, B) such that $t^{A'} = t^A\{\boldsymbol{x} \leftarrow \boldsymbol{t}_u\}$. We then demonstrate that $proj(t^A \leq 0 \wedge E_A, \mathcal{L}_A)$ is an interpolant for (A, B). This is formalized using the following rule:

$$\text{Comb}\ \frac{(A', B') \vdash \bot\ [t^{A'} \leq 0]}{(A\ , B\ ) \vdash \bot\ [proj(t^A \leq 0 \wedge E_A, \mathcal{L}_A)]} \quad \begin{array}{l} t^{A'} = t^A\,\{\boldsymbol{x} \leftarrow \boldsymbol{t_u}\}, \\ (A, B) \vdash t \leq 0\,[t^A \leq 0] \end{array}$$

The partial interpolant $t^A \leq 0$ that is needed to apply this rule is computed by "postponing" the substitutions. That is, after applying the Proj rule, the partial interpolant is kept in the form $at_1^A + bt_2^A\{\boldsymbol{x} \leftarrow \boldsymbol{t}_u\} \leq 0$ instead of $at_1^A\{\boldsymbol{x} \leftarrow\ \boldsymbol{t_u}\} + bt_2^A\{\boldsymbol{x} \leftarrow \boldsymbol{t}_u\} \leq 0$.

Soundness (of COMB). We show that the derived interpolant satisfies definition 1. First, we observe that applying a substitution before a projection is equivalent to applying it after the projection, i.e., $at_1\{\boldsymbol{x} \leftarrow \boldsymbol{t_u}\} + bt_2\{\boldsymbol{x} \leftarrow \boldsymbol{t}_u\} \leq 0 \equiv at_1 + bt_2\{\boldsymbol{x} \leftarrow \boldsymbol{t}_u\} \leq 0$, where the substitutions are naturally extended to terms. Thus, there is no immediate need to apply the substitutions $\{\boldsymbol{x} \leftarrow \boldsymbol{t}_u\}$ before projecting using the PROJ rule. More precisely, we can always derive a partial interpolant such that $(A', B') \vdash t \leq 0\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}[t^A \leq 0\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}]$. Subsequently, we know that $t_A \leq 0$ and $t \leq 0$ are linear combinations of inequalities in A and that, if projection occurs, $t^A \leq 0$ is the strongest convex projection. Thus, $t_A \leq 0$ is a partial interpolant for (A, B).

If an inconsistency is reached, we have derived a partial interpolant for (A, B) such that $t\{\boldsymbol{x} \leftarrow \boldsymbol{t}\} = c$ for some positive constant c. Since $t^A \leq 0$ is a partial interpolant we conclude $A \models \mathit{proj}(t^A \leq 0 \wedge E_A)$, which proves condition 1. To prove condition 2, we first note that $B \wedge \mathit{proj}(t^A \leq 0 \wedge E_A)$ and $(B \wedge t^A \leq 0 \wedge E_A)\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}$ are equisatisifiable. We know $B \models t - t^A \leq 0$ and, thus, conclude $(B \wedge t \leq 0 \wedge E_A)\{\boldsymbol{x} \leftarrow \boldsymbol{t}\}$. This contradicts $t\{\boldsymbol{x} \leftarrow \boldsymbol{t}\} = c$, proving condition 2. Condition 3 follows since *proj* eliminates all variables local to A. □

Example 5. Consider the pair $(A, B)_1$ given by $(A \wedge x - 3y + 2 = 0, B)$, where A and B are from example 3. There is only one equality to eliminate. Since it has a unit coefficient, the only substitution is $\{x \leftarrow (3y - 2)\}$. The two partial interpolants resulting in an inconsistency are:

$$\text{PROJ}\ \frac{\begin{array}{l}(A,B)_1 \vdash \quad\ \ 6y \leq 0\ [\ x - 3y + 1 \leq 0\ \{x \leftarrow (3y-2)\}] \\ (A,B)_1 \vdash -3y + 2 \leq 0\ [\qquad\qquad\ 0 \leq 0\ \{x \leftarrow (3y-2)\}]\end{array}}{(A,B)_1 \vdash \qquad\qquad \bot \quad [\ x - 3y + 1 \leq 0\ \{x \leftarrow (3y-2)\}]}$$

Note that the subsitutions were not applied to the partial interpolant in order to determine the final interpolant with the COMB rule:

$$\text{COMB}\ \frac{(A,B)_1 \vdash \bot\ [\ x - 3y + 1 \leq 0\ \{x \leftarrow (3y-2)\}]}{(A,B)_1 \vdash \bot\ [2x \leq 0 \wedge 3 \mid (x+2)]}$$

The resulting interpolant has been obtained by applying *proj* to $x - 3y + 1 \leq 0 \wedge x - 3y + 2 = 0$. □

Summary Fig. 1 shows the Omega test extended by our deduction rules in order to construct (partial) interpolants. Since the Omega test is complete for conjunctions of equalities, inequalities and stride constraints and we provide a deduction rule for each of its steps, the extended algorithm is complete as well.

In practice, we decouple the search for an inconsistency from the computation of an interpolant. That is, our implementation of OMEGA-INTERPOLATE takes A, B and an inconsistency proof as input and annotates this proof with partial interpolants. This allows many optimizations. Substitutions are not performed if all equalities encountered during step ① are satisfiable, and no partial interpolant will be computed for projections that do not lead to an inconsistency. Throughout the algorithm, arithmetic normalizations, such as replacing $3y \leq 0$ by $y \leq 0$, prevent the coefficients from growing unnecessarily.

OMEGA-INTERPOLATE (A, B)

① Introduce equalities and stride constraints from (A, B) using HYPEQ.
While eliminating equalities and stride constraints using ELIMEQ:
- If unsatisfiable constraint found: return "UNSAT" + interpolant generated by ELIMEQ.

If (A, B) has no inequalities: return "SAT".

② Introduce inequalities from (A, B), with substitutions applied, using HYPIN.
While projecting all inequalities using PROJ:
- If unsatisfiable constraint found: return "UNSAT" + interpolant generated by COMB.

If inexact projection occurred:
- Recursively call OMEGA-INTERPOLATE for each pair $(A', B')_i$ and $(A', B')_{\geq s+1}$, $i \in \{0, \dots, s\}$.
- If every pair is "UNSAT", compute the interpolant for (A', B') using SPLIN and return "UNSAT" + interpolant for (A, B) using COMB.

Otherwise: return "SAT".

Fig. 1. The Omega-Test with Interpolation

5.2 Time Complexity

We discuss the worst-case time complexity of our interpolation algorithm. Let a be the maximum absolute value of any coefficient and any periodicity occurring in the partial interpolants across the entire proof. In the original set of constraints, let w denote the maximum number of variables per constraint in A and e the number of equality and stride constraints in A. The cost of eliminating one variable x using *proj* is $O(e(w + \log^2 a))$. Let v be the number of local variables that occur in equalities. Procedure *proj* is applied v times in order to eliminate all local variables. In the worst case, the number of constraints e increases by one after each projection due to homogenization. In case of an inconsistency in the equalities, the worst-case time complexity of deriving an interpolant is therefore $O((w + \log^2 a)(ve + v^2))$.

For inequalities, the PROJ rule, including computing the strongest convex projection, has a complexity of $O(w \log a)$. If PROJ is applied p times, the overall interpolation complexity is therefore $O(pw \log a + (w + \log^2 a)(ve + v^2))$. We observed the run-time to be much smaller in practice, owing to many unit or small coefficients in the original pair (A, B) (also confirmed by [8]).

6 Conclusion

We have presented an interpolation method for *quantifier-free Presburger arithmetic* (QFP). Our method first eliminates equalities and stride constraints from the system and then projects inequalities using an extension of the Fourier-Motzkin variable elimination. These steps are formalized as proof rules that, as a side effect, transform *partial interpolants* to full interpolants for the given system of constraints. Our method is the first to enable efficient interpolation for quantifier-free linear integer arithmetic. In contrast to previous work, it permits combinations of equalities, inequalities and divisibility properties.

The results presented in this paper are expected to improve model checking based on counterexample-guided abstraction refinement (CEGAR). As shown in [16], program verification often requires computing inductive invariants involving constraints over integers. If a candidate invariant fails, interpolation can aid the discovery of new candidates. Our work permits the computation of interpolants for formulas given as combinations of the above-mentioned constraints.

A preliminary implementation of our algorithm shows that for QFP formulas occurring in practice, the run-time of the algorithm is much better than the estimated worst-case performance. We contribute this efficiency to small variable coefficients and a small number of variables per constraint.

References

1. Craig, W.: Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. J. Symb. Log. **22** (1957)
2. McMillan, K.L.: Interpolation and SAT-based model checking. In: CAV. (2003)
3. McMillan, K.L.: Lazy abstraction with interpolants. In: CAV. (2006)
4. Pudlák, P.: Lower bounds for resolution and cutting plane proofs and monotone computations. J. Symb. Log. **62** (1997)
5. McMillan, K.L.: An interpolating theorem prover. Theor. Comput. Sci. **345** (2005)
6. Bultan, T., Gerber, R., Pugh, W.: Symbolic model checking of infinite state systems using Presburger arithmetic. In: CAV. (1997)
7. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du Ier congrès de Mathématiciens des Pays Slaves. (1929)
8. Seshia, S.A., Bryant, R.E.: Deciding quantifier-free Presburger formulas using parameterized solution bounds. Logical Methods in Computer Science **1** (2005)
9. Ganesh, V., Berezin, S., Dill, D.L.: Deciding Presburger arithmetic by model checking and comparisons with other methods. In: FMCAD. (2002)
10. Pugh, W.: The Omega test: a fast and practical integer programming algorithm for dependence analysis. In: SC. (1991)
11. Krajícek, J.: Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. J. Symb. Log. **62** (1997)
12. Rybalchenko, A., Sofronie-Stokkermans, V.: Constraint solving for interpolation. In: VMCAI. (2007)
13. Cimatti, A., Griggio, A., Sebastiani, R.: Efficient interpolant generation in satisfiability modulo theories. In: TACAS. (2008) To appear.
14. Yorsh, G., Musuvathi, M.: A combination method for generating interpolants. In: CADE. (2005)
15. Beyer, D., Zufferey, D., Majumdar, R.: CSIsat: Interpolation for LA+EUF. In: CAV. (2008)
16. Jain, H., Clarke, E., Grumberg, O.: Efficient interpolation for linear diophantine (dis)equations and linear modular equations. In: CAV. (2008)
17. Cooper, D.C.: Theorem proving in arithmetic without multiplication. In: Machine Intelligence 7. (1972)