Interpolant synthesis for quadratic polynomial inequalities and combination with *EUF*

Ting Gan[1,4], Liyun Dai[1], Bican Xia[1], Naijun Zhan[2], Deepak Kapur[3], and Mingshuai Chen[2]

[1] LMAM & School of Mathematical Sciences, Peking University
[2] State Key Lab. of Computer Science, Institute of Software, Chinese Academy of Sciences
[3] Department of Computer Science, University of New Mexico
[4] State key Lab. of Software Engineering, Wuhan University

Abstract. An algorithm for generating interpolants for formulas which are conjunctions of quadratic polynomial inequalities (both strict and nonstrict) is proposed. The algorithm is based on a key observation that quadratic polynomial inequalities can be linearized if they are concave. A generalization of Motzkin's transposition theorem is proved, which is used to generate an interpolant between two mutually contradictory conjunctions of polynomial inequalities, using semi-definite programming in time complexity $\mathcal{O}(n^3 + nm)$, where n is the number of variables and m is the number of inequalities[5]. Using the framework proposed in [22] for combining interpolants for a combination of quantifier-free theories which have their own interpolation algorithms, a combination algorithm is given for the combined theory of concave quadratic polynomial inequalities and the equality theory over uninterpreted functions (*EUF*).

Keywords: Program verification, interpolant, concave quadratic polynomial, Motzkin's theorem, SOS, semi-definite programming.

1 Introduction

It is well known that the bottleneck of existing verification techniques including theorem proving, model-checking, abstraction and so on is the scalability. Interpolation-based technique provide a powerful mechanism for local and modular reasoning, which provides an effective solution to this challenge. The study of interpolation was pioneered by Krajíček [14] and Pudlák [19] in connection with theorem proving, by McMillan in connection with model-checking [16], by Graf and Saïdi [9], McMillan [17] and Henzinger et al. [10] in connection with abstraction like CEGAR, by Wang et al. [11] in connection with machine-learning based invariant generation. Since then, developing efficient algorithms for generating interpolants for various theories and their use in verification have become an active research area [17,10,26,12,20,13,3,12,18,26]. In addition, D'Silva et al. [6] investigated strengths of various interpolants.

Methods have been developed for generating interpolants for Presburger arithmetic, decidable fragments of first-order logic, theory of equality over uninterpreted functions as well as their combination. However, in the literature, there is little work on how

[5] This complexity analysis assumes that despite the numerical nature of approximate SDP algorithms, they are able to generate correct answers in a fixed number of calls.

to synthesize non-linear interpolants, although nonlinear polynomials inequalities have been found useful to express invariants for software involving number theoretic functions as well as hybrid systems [28,27]. In [5], Dai et al. had a first try and gave an algorithm for generating interpolants for conjunctions of mutually contradictory nonlinear polynomial inequalities based on the existence of a witness guaranteed by Stengle's Positivstellensatz [23] that can be computed using semi-definite programming (SDP). Their algorithm is incomplete in general but if every variable ranges over a bounded interval (called Archimedean condition), then their algorithm is complete. A major limitation of their work is that two mutually contradictory formulas α, β must have the same set of variables.

We propose an algorithm to generate interpolants for quadratic polynomial inequalities (including strict inequalities). Based on the insight that for analyzing the solution space of concave quadratic polynomial inequalities, it suffices to linearize them. A generalization of Motzkin's transposition theorem is proved to be applicable for concave quadratic polynomial inequalities (both strict and nonstrict). Using this, we prove the existence of an interpolant for two mutually contradictory conjunctions α, β of concave quadratic polynomial inequalities. The proposed algorithm is recursive with the basis step of the algorithm relying on an additional condition (called the $\mathbf{NSC}$ condition). In this case, an interpolant output by the algorithm is a strict or a nonstrict inequality similar to the linear case. If $\mathbf{NSC}$ is not satisfied, then linear equalities on variables are derived resulting in simpler interpolation problems over fewer variables; the algorithm is recursively invoked on these smaller problem. The output of this recursive algorithm is in general an interpolant that is a disjunction of conjunction of polynomial inequalities. $\mathbf{NSC}$ can be checked in polynomial time by SDP algorithms; even though such algorithms are not exact and produce numerical errors, they often generate acceptable results in a few calls. It is proved that the interpolation algorithm is of polynomial time complexity in the number of variables and polynomial inequalities given that the time complexity of SDP algorithms is polynomial in the size of their input; this assumes that an SDP tool returns an approximate answer sufficient to generate a correct interpolant in a fixed number of calls.

Later, we develop a combination algorithm for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and equality theory over uninterpreted function symbols (*EUF*). We use the hierarchical calculus framework proposed in [22] and used in [20] for combining linear inequalities with *EUF*. We show that concavity condition on quadratic polynomials inequalities disallows derivation of nonlinear equalities of degree ≥ 2; further, under $\mathbf{NSC}$ on concave quadratic polynomial inequalities, only linear inequalities can be used to derive possible linear equalities. As a result, the algorithm for deducing equalities from linear inequalities in [20] as well as the `SEP` algorithm for separating terms expressed in common symbols in α, β can be used for interpolation generation for the combined theory of quadratic polynomial inequalities and *EUF*.

A prototypical implementation indicates the scalability and efficiency of the proposed approach.

The paper is organized as follows. After introducing some preliminaries in the next section, Section 3 discusses the linearization of concave quadratic polynomial. Section

4 presents an approach for computing an interpolant for two mutually contradictory conjunctions α, β of concave quadratic polynomial inequalities using SDP. Section 5 extends this algorithm to the combined theory of concave quadratic inequalities and *EUF*. Section 6 presents a preliminary implementation of the proposed algorithms and gives some comparison with related work. We draw a conclusion in Section 7. Because of space limit, we omit all proofs, please refer to the full version [8] for the details.

2 Preliminaries

Let $\mathbb{Q}$ and $\mathbb{R}$ be the set of rational and real numbers, respectively. Let $\mathbb{R}[\mathbf{x}]$ be the polynomial ring over $\mathbb{R}$ with variables $\mathbf{x} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$. An atomic polynomial formula φ is of the form $p(\mathbf{x}) \diamond 0$, where $p(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$, and $\diamond$ can be any of $>, \geq$. Let $\mathbf{PT}(\mathbb{R})$ be a first-order theory of polynomials with real coefficients. In this paper, we are focusing on quantifier-free fragment of $\mathbf{PT}(\mathbb{R})$. Later we discuss quantifier-free theory of equality of terms over uninterpreted function symbols and its combination with the quantifier-free fragment of $\mathbf{PT}(\mathbb{R})$. Let Σ be a set of (new) function symbols and $\mathbf{PT}(\mathbb{R})^{\Sigma}$ be the extension of the quantifier-free theory with uninterpreted function symbols in Σ.

For convenience, we use $\bot$ to stand for *false* and $\top$ for *true* in what follows.

Craig showed that given two formulas ϕ and ψ in a first-order logic $\mathcal{T}$ s.t. $\phi \models \psi$, there always exists an *interpolant* I over the common symbols of ϕ and ψ s.t. $\phi \models I, I \models \psi$. In the verification literature, this terminology has been abused following [17], where a *reverse interpolant* (coined by Kovács and Voronkov in [13]) I over the common symbols of ϕ and ψ is defined by

Definition 1. *Given ϕ and ψ in a theory $\mathcal{T}$ s.t. $\phi \wedge \psi \models_{\mathcal{T}} \bot$, a formula I is a* (reverse) interpolant *of ϕ and ψ if* (i) *$\phi \models_{\mathcal{T}} I$;* (ii) *$I \wedge \psi \models_{\mathcal{T}} \bot$; and* (iii) *$I$ only contains common symbols and free variables shared by ϕ and ψ.*

Clearly, $\phi \models_{\mathcal{T}} \psi$ iff $\phi \wedge \neg\psi \models_{\mathcal{T}} \bot$. Thus, I is an interpolant of ϕ and ψ iff I is a reverse interpolant of ϕ and $\neg\psi$. We abuse the terminology by calling reverse interpolants as interpolants.

2.1 Motzkin's transposition theorem

Motzkin's transposition theorem [21] is one of the fundamental results about linear inequalities; it also served as a basis of the interpolant generation algorithm for the quantifier-free theory of linear inequalities in [20].

Theorem 1 (Motzkin's transposition theorem [21]). *Let A and B be matrices and let $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ be column vectors. Then there exists a vector $\mathbf{x}$ with $A\mathbf{x} \geq \boldsymbol{\alpha}$ and $B\mathbf{x} > \boldsymbol{\beta}$, iff for all row vectors $\mathbf{y}, \mathbf{z} \geq 0$:*

$$(i) \text{ if } \mathbf{y}A + \mathbf{z}B = 0 \text{ then } \mathbf{y}\boldsymbol{\alpha} + \mathbf{z}\boldsymbol{\beta} \leq 0;$$
$$(ii) \text{ if } \mathbf{y}A + \mathbf{z}B = 0 \text{ and } \mathbf{z} \neq 0 \text{ then } \mathbf{y}\boldsymbol{\alpha} + \mathbf{z}\boldsymbol{\beta} < 0.$$

The following variant of Theorem 1 is used later.

Corollary 1. *Let $A \in \mathbb{R}^{r\times n}$ and $B \in \mathbb{R}^{s\times n}$ be matrices and $\boldsymbol{\alpha} \in \mathbb{R}^r$ and $\boldsymbol{\beta} \in \mathbb{R}^s$ be column vectors, where $A_i, i = 1,\ldots,r$ is the ith row of A and $B_j, j = 1,\ldots,s$ is the jth row of B. There does not exist a vector $\mathbf{x}$ with $A\mathbf{x} \geq \boldsymbol{\alpha}$ and $B\mathbf{x} > \boldsymbol{\beta}$, iff there exist real numbers $\lambda_1,\ldots,\lambda_r \geq 0$ and $\eta_0,\eta_1,\ldots,\eta_s \geq 0$ s.t.*

$$\sum_{i=1}^{r} \lambda_i(A_i\mathbf{x} - \alpha_i) + \sum_{j=1}^{s} \eta_j(B_j\mathbf{x} - \beta_j) + \eta_0 \equiv 0 \;\; \textit{with} \sum_{j=0}^{s} \eta_j > 0. \tag{1}$$

3 Concave quadratic polynomials and their linearization

Given $n \times n$-matrix A, we say A is *negative semi-definite*, written as $A \preceq 0$, if for every vector $\mathbf{x}$, $\mathbf{x}^{\mathrm{T}}A\mathbf{x} \leq 0$, and *positive semi-definite*, written as $A \succeq 0$, if for every vector $\mathbf{x}$, $\mathbf{x}^{\mathrm{T}}A\mathbf{x} \geq 0$. Let $A = (a_{ij})$ and $B = (b_{ij})$ be two matrices in $\mathbb{R}^{m\times n}$, the *inner product* of A and B, denoted by $\langle A, B\rangle$, is defined as $\langle A, B\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} \times b_{ij}$.

Definition 2 (Concave Quadratic). *A polynomial $f \in \mathbb{R}[\mathbf{x}]$ is called* concave quadratic (CQ) *if the following two conditions hold:*

(i) *f has total degree at most 2, i.e., it has the form $f = \mathbf{x}^{\mathrm{T}}A\mathbf{x} + 2\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{x} + a$, where A is a real symmetric matrix, $\boldsymbol{\alpha}$ is a column vector and $a \in \mathbb{R}$;*
(ii) *the matrix A is negative semi-definite.*

It is easy to see that if $f \in \mathbb{R}[\mathbf{x}]$ is linear, then f is CQ because its total degree is 1 and the corresponding A is 0 which is of course negative semi-definite.

A quadratic polynomial $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}A\mathbf{x} + 2\boldsymbol{\alpha}^{\mathrm{T}}\mathbf{x} + a$ can also be represented as an inner product of matrices, *i.e.*, $\left\langle P, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \mathbf{x}\mathbf{x}^{\mathrm{T}} \end{pmatrix} \right\rangle$, with P as $\begin{pmatrix} a & \alpha^{\mathrm{T}} \\ \alpha & A \end{pmatrix}$.

3.1 Linearization

Given a quadratic polynomial $f(\mathbf{x}) = \left\langle P, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \mathbf{x}\mathbf{x}^{\mathrm{T}} \end{pmatrix} \right\rangle$, its *linearization* is defined as $f(\mathbf{x}) = \left\langle P, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle$, where $\boldsymbol{X}$ is a symmetric matrix and $\begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \succeq 0$.

Let $\overline{\boldsymbol{X}} = (\boldsymbol{X}_{(1,1)}, \boldsymbol{X}_{(2,1)}, \boldsymbol{X}_{(2,2)}, \ldots, \boldsymbol{X}_{(k,1)}, \ldots, \boldsymbol{X}_{(k,k)}, \ldots, \boldsymbol{X}_{(n,1)}, \ldots, \boldsymbol{X}_{(n,n)})$ be the vector variable with dimension $\frac{n(n+1)}{2}$ corresponding to the matrix $\boldsymbol{X}$. Since $\boldsymbol{X}$ is a symmetric matrix, $\left\langle P, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle$ is a linear expression in $\mathbf{x}, \overline{\boldsymbol{X}}$.

Consider quadratic polynomials f_i and g_j ($i = 1,\ldots,r$, $j = 1,\ldots,s$),

$$f_i = \mathbf{x}^{\mathrm{T}}A_i\mathbf{x} + 2\boldsymbol{\alpha}_i^{\mathrm{T}}\mathbf{x} + a_i, \quad g_j = \mathbf{x}^{\mathrm{T}}B_j\mathbf{x} + 2\boldsymbol{\beta}_j^{\mathrm{T}}\mathbf{x} + b_j,$$

where A_i, B_j are symmetric $n \times n$ matrices, $\boldsymbol{\alpha}_i, \boldsymbol{\beta}_j \in \mathbb{R}^n$, and $a_i, b_j \in \mathbb{R}$. Then

$$f_i(\mathbf{x}) = \left\langle P_i, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \mathbf{x}\mathbf{x}^{\mathrm{T}} \end{pmatrix} \right\rangle, \;\; g_j(\mathbf{x}) = \left\langle Q_j, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \mathbf{x}\mathbf{x}^{\mathrm{T}} \end{pmatrix} \right\rangle,$$

where $P_i = \begin{pmatrix} a_i & \boldsymbol{\alpha}_i^{\mathrm{T}} \\ \boldsymbol{\alpha}_i & A_i \end{pmatrix}$, $Q_j = \begin{pmatrix} b_j & \boldsymbol{\beta}_j^{\mathrm{T}} \\ \boldsymbol{\beta}_j & B_j \end{pmatrix}$ are $(n+1) \times (n+1)$ matrices.

For CQ polynomials f_is and g_js, let

$$K \hat{=} \{\mathbf{x} \in \mathbb{R}^n \mid f_1(\mathbf{x}) \geq 0, \ldots, f_r(\mathbf{x}) \geq 0, g_1(\mathbf{x}) > 0, \ldots, g_s(\mathbf{x}) > 0\}, \tag{2}$$

$$K_1 \hat{=} \{\mathbf{x} \mid \exists \boldsymbol{X}. \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \succeq 0 \wedge \bigwedge_{i=1}^{r} \left\langle P_i, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle \geq 0 \wedge \bigwedge_{j=1}^{s} \left\langle Q_j, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle > 0\}. \tag{3}$$

In [7,15], when K and K_1 are defined only with f_is without g_js, *i.e.*, only with nonstrict inequalities, it is proved that $K = K_1$. By Theorem 2 below, we show that $K = K_1$ also holds even in the presence of strict inequalities when f_i and g_j are CQ. So, when f_is and g_js are CQ, the CQ polynomial inequalities can be transformed equivalently to a set of linear inequality constraints and a positive semi-definite constraint.

Theorem 2. *Let $f_1, \ldots, f_r, g_1, \ldots, g_s \in \mathbb{R}[\mathbf{x}]$ be CQ, K and K_1 as above, then $K = K_1$.*

3.2 Motzkin's theorem in matrix form

If $\left\langle P, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle$ is seen as a linear expression in $\mathbf{x}, \overline{\boldsymbol{X}}$, then Corollary 1 can be reformulated as:

Corollary 2. *Let $\mathbf{x}$ be a column vector variable with dimension n and $\boldsymbol{X}$ be an $n \times n$ symmetric matrix variable. Suppose $P_1, \ldots, P_r$ and $Q_1, \ldots, Q_s$ are $(n+1) \times (n+1)$ symmetric matrices. Let*

$$V \hat{=} \{(\mathbf{x}, \boldsymbol{X}) \mid \bigwedge_{i=1}^{r} \left\langle P_i, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle \geq 0, \bigwedge_{i=1}^{s} \left\langle Q_j, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle > 0\},$$

then $V = \emptyset$ iff there exist $\lambda_1, \ldots, \lambda_r \geq 0$ and $\eta_0, \eta_1, \ldots, \eta_s \geq 0$ such that

$$\sum_{i=1}^{r} \lambda_i \left\langle P_i, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle + \sum_{j=1}^{s} \eta_j \left\langle Q_j, \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} \\ \mathbf{x} & \boldsymbol{X} \end{pmatrix} \right\rangle + \eta_0 \equiv 0, \quad \eta_0 + \eta_1 + \ldots + \eta_s > 0.$$

4 Interpolants for concave quadratic polynomial inequalities

Problem 1: Given two formulas ϕ and ψ on n variables with $\phi \wedge \psi \models \bot$, where

$$\phi = f_1 \geq 0 \wedge \ldots \wedge f_{r_1} \geq 0 \wedge g_1 > 0 \wedge \ldots \wedge g_{s_1} > 0,$$
$$\psi = f_{r_1+1} \geq 0 \wedge \ldots \wedge f_r \geq 0 \wedge g_{s_1+1} > 0 \wedge \ldots \wedge g_s > 0,$$

in which $f_1, \ldots, f_r, g_1, \ldots, g_s$ are all CQ. Our goal is to develop an algorithm to generate a (reverse) Craig interpolant I for ϕ and ψ, on the common variables of ϕ and ψ, s.t. $\phi \models I$ and $I \wedge \psi \models \bot$. We use $\mathbf{x} = (x_1, \ldots, x_d)$ to stand for the common variables appearing in both ϕ and ψ, $\mathbf{y} = (y_1, \ldots, y_u)$ for the variables appearing only in ϕ and $\mathbf{z} = (z_1, \ldots, z_v)$ for the variables appearing only in ψ, where $d + u + v = n$. We call the conjunctive theory of CQ polynomial inequalities as ***CQI***.

The proposed Algorithm IG-CQI in Section 4.5 is recursive: the base case is when no sum of squares (SOS) polynomial can be generated by a nonpositive constant combination of the polynomials in nonstrict inequalities in $\phi \wedge \psi$. When this condition is not satisfied, then identify variables which can be eliminated by replacing them by linear expressions in terms of other variables and generate equisatisfiable problem with fewer variables on which the algorithm can be recursively invoked.

4.1 NSC condition and generalization of Motzkin's theorem

Definition 3. *Formulas ϕ and ψ in Problem 1 satisfy the non-existence of an SOS polynomial condition (*$\mathbf{NSC}$*) iff there do not exist $\delta_1 \geq 0, \ldots, \delta_r \geq 0$, s.t. $-(\delta_1 f_1 + \ldots + \delta_r f_r)$ is a non-zero SOS.*

Note that nonnegative quadratic polynomials are all SOS. So, the above condition implies that there is no nonnegative constant combination of nonstrict inequalities which is always *nonpositive*. If quadratic polynomials appearing in ϕ and ψ are linearized, then the above condition is equivalent to requiring that every nonnegative linear combination of the linearization of nonstrict inequalities in ϕ and ψ is *negative*.

The following theorem is a generalization of Motzkin's theorem to CQI and gives a method when $\mathbf{NSC}$ is satisfied, for generating an interpolant by considering linearization of ϕ, ψ in Problem 1 and using Corollary 2.

Theorem 3. *Let $f_1, \ldots, f_r, g_1, \ldots, g_s$ be CQ polynomials in Problem 1. If* $\mathbf{NSC}$ *holds, then there exist $\lambda_i \geq 0$ ($i = 1, \cdots, r$), $\eta_j \geq 0$ ($j = 0, 1, \cdots, s$) and a quadratic SOS polynomial $h \in \mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ such that*

$$\sum_{i=1}^{r} \lambda_i f_i + \sum_{j=1}^{s} \eta_j g_j + \eta_0 + h \equiv 0, \quad \eta_0 + \eta_1 + \ldots + \eta_s = 1. \tag{4}$$

4.2 Base case: generating interpolant when NSC is satisfied

Using Theorem 3, an interpolant for ϕ and ψ is generated from the SOS polynomial h by splitting it into two SOS polynomials as shown below.

Theorem 4. *Let ϕ and ψ be as in Problem 1 with $\phi \wedge \varphi \models \bot$, which satisfy* $\mathbf{NSC}$*. Then there exist $\lambda_i \geq 0$ ($i = 1, \cdots, r$), $\eta_j \geq 0$ ($j = 0, 1, \cdots, s$) and two quadratic SOS polynomial $h_1 \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$ and $h_2 \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$ such that*

$$\sum_{i=1}^{r} \lambda_i f_i + \sum_{j=1}^{s} \eta_j g_j + \eta_0 + h_1 + h_2 \equiv 0, \;\; \eta_0 + \eta_1 + \ldots + \eta_s = 1. \tag{5}$$

Let $I = \sum_{i=1}^{r_1} \lambda_i f_i + \sum_{j=1}^{s_1} \eta_j g_j + \eta_0 + h_1$. Then $I \in \mathbb{R}[\mathbf{x}]$, and if $\sum_{j=0}^{s_1} \eta_j > 0$, then $I > 0$ is an interpolant otherwise $I \geq 0$ is an interpolant.

Further, we can prove that h, h_1, h_2 have the following form:

$$\begin{aligned}(\mathrm{H}) \; : \;\; & h(\mathbf{x}, \mathbf{y}, \mathbf{z}) = a_1(y_1 - l_1(\mathbf{x}, y_2, \ldots, y_u))^2 + \cdots + a_u(y_u - l_u(\mathbf{x}))^2 + a_{u+1}(z_1 - \\ & l_{u+1}(\mathbf{x}, z_2, \ldots, z_v))^2 + \cdots + a_{u+v}(z_v - l_{u+v}(\mathbf{x}))^2 + a_{u+v+1}(x_1 - l_{u+v+1}(x_2, \ldots, x_d))^2 + \\ & \cdots + a_{u+v+d}(x_d - l_{u+v+d})^2 + a_{u+v+d+1},\end{aligned}$$

(H1) : $h_1(\mathbf{x},\mathbf{y}) = a_1(y_1 - l_1(\mathbf{x}, y_2, \ldots, y_u))^2 + \cdots + a_u(y_u - l_u(\mathbf{x}))^2 + \frac{a_{u+v+1}}{2}(x_1 - l_{u+v+1}(x_2, \ldots, x_d))^2 + \cdots + \frac{a_{u+v+d}}{2}(x_d - l_{u+v+d})^2 + \frac{a_{u+v+d+1}}{2}$,

(H2) : $h_2(\mathbf{x},\mathbf{z}) = a_{u+1}(z_1 - l_{u+1}(\mathbf{x}, z_2, \ldots, z_v))^2 + \cdots + a_{u+v}(z_v - l_{u+v}(\mathbf{x}))^2 + \frac{a_{u+v+1}}{2}(x_1 - l_{u+v+1}(x_2, \ldots, x_d))^2 + \cdots + \frac{a_{u+v+d}}{2}(x_d - l_{u+v+d})^2 + \frac{a_{u+v+d+1}}{2}$,

where $a_i \geq 0$ and l_j's are linear expressions. These forms of h_1, h_2 are used to generate equalities among variables later in the algorithm when **NSC** is not satisfied.

4.3 Computing interpolant using semi-definite programming

Let $W = \begin{pmatrix} 1 & \mathbf{x}^{\mathrm{T}} & \mathbf{y}^{\mathrm{T}} & \mathbf{z}^{\mathrm{T}} \\ \mathbf{x} & \mathbf{x}\mathbf{x}^{\mathrm{T}} & \mathbf{x}\mathbf{y}^{\mathrm{T}} & \mathbf{x}\mathbf{z}^{\mathrm{T}} \\ \mathbf{y} & \mathbf{y}\mathbf{x}^{\mathrm{T}} & \mathbf{y}\mathbf{y}^{\mathrm{T}} & \mathbf{y}\mathbf{z}^{\mathrm{T}} \\ \mathbf{z} & \mathbf{z}\mathbf{x}^{\mathrm{T}} & \mathbf{z}\mathbf{y}^{\mathrm{T}} & \mathbf{z}\mathbf{z}^{\mathrm{T}} \end{pmatrix}$, $f_i = \langle P_i, W\rangle$, $g_j = \langle Q_j, W\rangle$, where P_i and Q_j are $(n+1)\times(n+1)$ matrices, and $h_1 = \langle M, W\rangle$, $h_2 = \langle \hat{M}, W\rangle$, $M = (M_{ij})_{4\times 4}, \hat{M} = (\hat{M}_{ij})_{4\times 4}$ with appropriate dimensions, e.g., $M_{12} \in \mathbb{R}^{1\times d}$ and $\hat{M}_{34} \in \mathbb{R}^{u\times v}$. Then, with **NSC**, by Theorem 4, computing the interpolant is reduced to the following **SDP** feasibility problem.

Find: $\lambda_1, \ldots, \lambda_r, \eta_1, \ldots, \eta_s \in \mathbb{R}$ and symmetric matrices $M, \hat{M} \in \mathbb{R}^{(n+1)\times(n+1)}$ s.t.

$$\begin{cases} \sum_{i=1}^{r} \lambda_i P_i + \sum_{j=1}^{s} \eta_j Q_j + \eta_0 E_{(1,1)} + M + \hat{M} = 0, \sum_{j=1}^{s} \eta_j = 1, \\ M_{41} = (M_{14})^{\mathrm{T}} = 0, M_{42} = (M_{24})^{\mathrm{T}} = 0, M_{43} = (M_{34})^{\mathrm{T}} = 0, M_{44} = 0, \\ \hat{M}_{31} = (\hat{M}_{13})^{\mathrm{T}} = 0, \hat{M}_{32} = (\hat{M}_{23})^{\mathrm{T}} = 0, \hat{M}_{33} = 0, \hat{M}_{34} = (\hat{M}_{43})^{\mathrm{T}} = 0, \\ M \succeq 0, \hat{M} \succeq 0, \lambda_i \geq 0, \eta_j \geq 0, \text{ for } i = 1, \ldots, r, j = 1, \ldots, s, \end{cases}$$

where $E_{(1,1)}$ is a $(n+1)\times(n+1)$ matrix, whose all entries are 0 except for $(1,1) = 1$.

This standard **SDP** feasibility problem can be efficiently solved by **SDP** solvers such as CSDP [1], SDPT3 [24], etc. A major weakness of these algorithms is their incompleteness, however.

Approximate nature of SDP algorithms Even though known SDP algorithms are of polynomial complexity, they are numerical and are not guaranteed to produce exact answers; they are however able to generate results within a very small threshold in a fixed number of iterations. Such techniques are thus considerably more attractive than solving the Problem 1 using exact symbolic methods of high complexity. This is especially critical for scaling our approach. To guarantee the soundness of our approach, we check results produced by approximate numerical algorithms by symbolic checking [4] and numeric-symbolic method [25] to verify whether an interpolant so computed does indeed satisfy the conditions in Definition 1. If not, we can tone down the threshold of the **SDP** and repeat the above procedure.

4.4 General case

The case of *Var*(ϕ) $\subset$ *Var*(ψ) is easy: ϕ itself serves as an interpolant of ϕ and ψ. We thus assume that *Var*(ϕ) $\not\subseteq$ *Var*(ψ). If ϕ and ψ do not satisfy **NSC**, then an SOS

polynomial $h(\mathbf{x},\mathbf{y},\mathbf{z}) = -(\sum_{i=1}^{r} \lambda_i f_i)$ can be computed which can be split into two SOS polynomials $h_1(\mathbf{x},\mathbf{y})$ and $h_2(\mathbf{x},\mathbf{z})$ as discussed in Subsection 4.2. Then an SOS polynomial $f(\mathbf{x})$ s.t. $\phi \models f(\mathbf{x}) \geq 0$ and $\psi \models -f(\mathbf{x}) \geq 0$ can be constructed by setting $f(\mathbf{x}) = (\sum_{i=1}^{r_1} \delta_i f_i) + h_1 = -(\sum_{i=r_1+1}^{r} \delta_i f_i) - h_2, \delta_i \geq 0$. We show below how "simpler" interpolation subproblems ϕ', ψ' are constructed from ϕ and ψ using f.

Lemma 1. *If* **NSC** *is not satisfied, then there exists $f \in \mathbb{R}[\mathbf{x}]$ s.t. $\phi \Leftrightarrow \phi_1 \vee \phi_2$ and $\psi \Leftrightarrow \psi_1 \vee \psi_2$, where,*

$$\phi_1 = (f > 0 \wedge \phi),\ \phi_2 = (f = 0 \wedge \phi),\ \psi_1 = (-f > 0 \wedge \psi),\ \psi_2 = (f = 0 \wedge \psi). \quad (6)$$

It easily follows that an interpolant I for ϕ and ψ can be constructed from an interpolant $I_{2,2}$ for ϕ_2 and ψ_2.

Theorem 5. *Let ϕ, ψ, $\phi_1, \phi_2, \psi_1, \psi_2$ be same as in Lemma 1, $I_{2,2}$ be an interpolant for ϕ_2 and ψ_2, then $I := (f > 0) \vee (f \geq 0 \wedge I_{2,2})$ is an interpolant for ϕ and ψ.*

If h and hence h_1, h_2 have a positive constant $a_{u+v+d+1} > 0$, then f cannot be 0, implying that ϕ_2 and ψ_2 are $\bot$. We thus have

Theorem 6. *With ϕ, ψ, $\phi_1, \phi_2, \psi_1, \psi_2$ as in Lemma 1 and h has $a_{u+v+d+1} > 0$, then $f > 0$ is an interpolant for ϕ and ψ.*

In case h does not have a constant, *i.e.*, $a_{u+v+d+1} = 0$, from the fact that h_1 is an SOS and has the form (H1), each nonzero square term in h_1 is identically 0. This implies that some of the variables in $\mathbf{x}, \mathbf{y}$ can be linearly expressed in term of other variables; the same argument applies to h_2 as well. In particular, at least one variable is eliminated in both ϕ_2 and ψ_2, reducing the number of variables appearing in ϕ and ψ, which ensures the termination of the algorithm.

Theorem 7. *If h above does not have a constant, i.e., if $a_{u+v+d+1} = 0$, by eliminating (at least one) variables in ϕ and ψ in terms of other variables as derived from $h_1 = 0$, $h_2 = 0$, mutually contradictory formulas ϕ', ψ' with fewer variables are derived by*

$$\phi' = \textstyle\bigwedge_{i=1}^{r_1} \hat{f}_i \geq 0 \wedge \bigwedge_{j=1}^{s_1} \hat{g}_j > 0, \quad \psi' = \bigwedge_{i=r_1+1}^{r} \hat{f}_i \geq 0 \wedge \bigwedge_{j=s_1+1}^{s} \hat{g}_j > 0,$$

where $\hat{f}_i$s and $\hat{g}_j$s are derived from the respective f_i and g_i by replacing the eliminated variable(s) with the corresponding resulting expression(s).

The following simple example illustrates how the above construction works.

Example 1. Let $f_1 = x_1, f_2 = x_2, f_3 = -x_1^2 - x_2^2 - 2x_2 - z^2, g_1 = -x_1^2 + 2x_1 - x_2^2 + 2x_2 - y^2$. Two formulas $\phi := (f_1 \geq 0) \wedge (f_2 \geq 0) \wedge (g_1 > 0)$, $\psi := (f_3 \geq 0)$. $\phi \wedge \psi \models \bot$. **NSC** does not hold, since $h = -(0f_1 + 2f_2 + f_3) = x_1^2 + x_2^2 + z^2$ is an SOS; h is split into $h_1 = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2$, $h_2 = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 + z^2$. Thus $f = 0f_1 + 2f_2 + h_1 = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 + 2x_2$.

For the recursive call, we construct ϕ' from ϕ by adding $x_1 = 0, x_2 = 0$ derived from $h_1 = 0$; similarly ψ' is constructed from ψ by adding $x_1 = x_2 = 0, z = 0$ derived from $h_2 = 0$. That is, $\phi' = 0 \geq 0 \wedge 0 \geq 0 \wedge -y^2 > 0 = \bot$, $\psi' = 0 \geq 0 = \top$. Thus, $I(\phi', \psi') := (0 > 0) = \bot$ is an interpolant for (ϕ', ψ').

An interpolant for ϕ and ψ is thus $(f(x) > 0) \vee (f(x) = 0 \wedge I(\phi', \psi'))$, which is $\frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 + 2x_2 > 0$.

4.5 Algorithms

The above recursive approach is formally described as Algorithm 2. For the base case when ϕ, ψ satisfy **NSC**, it invokes Algorithm 1 using known SDP algorithms. For a predefined threshold, an **SDP** problem can be solved in polynomial time, say $g(k)$, where k is the input size [7]. Further its solution can be checked to determine whether a formula thus generated is indeed an interpolant; in case of failure, the process is repeated typically leading to convergence in a few iterations.

Algorithm 1: `Interpolation Generation for` **NSC** `Case (IG-NSC)`

input : ϕ and ψ satisfying **NSC**, and $\phi \wedge \psi \models \bot$, where
$\phi = f_1 \geq 0 \wedge \ldots \wedge f_{r_1} \geq 0 \wedge g_1 > 0 \wedge \ldots \wedge g_{s_1} > 0$,
$\psi = f_{r_1+1} \geq 0 \wedge \ldots \wedge f_r \geq 0 \wedge g_{s_1+1} > 0 \wedge \ldots \wedge g_s > 0$,
$f_1, \ldots, f_r, g_1, \ldots, g_s$ are all CQ polynomials,
$f_1, \ldots, f_{r_1}, g_1, \ldots, g_{s_1} \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$, $f_{r_1+1}, \ldots, f_r, g_{s_1+1}, \ldots, g_s \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$
output: A formula I to be an interpolant for ϕ and ψ

1 **Find** $\lambda_1, \ldots, \lambda_r \geq 0, , \eta_0, \eta_1, \ldots, \eta_s \geq 0, h_1 \in \mathbb{R}[\mathbf{x}, \mathbf{y}], h_2 \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$ by SDP s.t.

$$\sum_{i=1}^{r} \lambda_i f_i + \sum_{j=1}^{s} \eta_j g_j + \eta_0 + h_1 + h_2 \equiv 0, \;\; \eta_0 + \eta_1 + \ldots + \eta_s = 1,$$

where h_1, h_2 are SOS polynomials;

2 $f := \sum_{i=1}^{r_1} \lambda_i f_i + \sum_{j=1}^{s_1} \eta_j g_j + \eta_0 + h_1$;
3 **if** $\sum_{j=0}^{s_1} \eta_j > 0$ **then** $I := (f > 0)$; **else** $I := (f \geq 0)$;
4 **return** I

Algorithm 2: `Interpolation Generation for CQ Formulas (IG-CQI)`

input : ϕ and ψ with $\phi \wedge \psi \models \bot$, where
$\phi = f_1 \geq 0 \wedge \ldots \wedge f_{r_1} \geq 0 \wedge g_1 > 0 \wedge \ldots \wedge g_{s_1} > 0$,
$\psi = f_{r_1+1} \geq 0 \wedge \ldots \wedge f_r \geq 0 \wedge g_{s_1+1} > 0 \wedge \ldots \wedge g_s > 0$,
$f_1, \ldots, f_r, g_1, \ldots, g_s$ are all CQ polynomials,
$f_1, \ldots, f_{r_1}, g_1, \ldots, g_{s_1} \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$, and $f_{r_1+1}, \ldots, f_r, g_{s_1+1}, \ldots, g_s \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$
output: A formula I to be an interpolant for ϕ and ψ

1 **if** *Var*$(\phi) \subseteq$ *Var*(ψ) **then** $I := \phi$; **return** I;
2 **Find** $\delta_1, \ldots, \delta_r \geq 0, h \in \mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ by SDP s.t. $\sum_{i=1}^{r} \delta_i f_i + h \equiv 0$ and h is SOS;
`/* Check the condition` **NSC** `*/`
3 **if** *no solution* **then** $I :=$ IG-NSC(ϕ, ψ); **return** I;
`/*` **NSC** `holds */`
4 Construct $h_1 \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$ and $h_2 \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$ with the forms (H1) and (H2);
5 $f := \sum_{i=1}^{r_1} \delta_i f_i + h_1 = -\sum_{i=r_1+1}^{r} \delta_i f_i - h_2$;
6 Construct ϕ' and ψ' using Theorems 6 & 7 by eliminating variables due to $h_1 = h_2 = 0$;
7 $I' :=$ IG-CQI(ϕ', ψ') ;
8 $I := (f > 0) \vee (f \geq 0 \wedge I')$;
9 **return** I

Theorem 8 (Soundness and Completeness). *Algorithm 2 computes an interpolant I if it exists for any given ϕ and ψ with $\phi \wedge \psi \models \bot$.*

Theorem 9. *The complexity of* IG-NSC *and* IG-CQI *are $\mathcal{O}(g(r+s+n^2))$ [6], and $\mathcal{O}(ng(r+s+n^2))$, respectively, where r is the number of nonstrict inequalities, s is the number of strict inequalities, and n is the number of variables.*

5 Combination: *CQI* with *EUF*

This section combines the conjunctive theory of concave quadratic polynomial inequalities (*CQI*) with the theory of equality over uninterpreted function symbols (*EUF*). Algorithm 4 for generating interpolants for the combined theories is patterned after the algorithm $\text{INTER}_{LI(Q)^{\Sigma}}$ in Figure 3 in [20] following the hierarchical reasoning and interpolation generation framework in [22] with the following key differences[7]:

1. To generate interpolants for CQI, Algorithm 2 is called.
2. If **NSC** is satisfied by nonstrict polynomial inequalities, linear equalities are deduced only from the linear inequalities; it is thus possible to use $\text{INTER}_{LI(Q)^{\Sigma}}$ in Figure 3 in [20] for deducing equalities; separating terms for mixed equalities are computed in the same way as in the algorithm SEP in [20]. Further, it can be proved that a nonlinear polynomial equality of degree ≥ 2 cannot be generated from CQI.
3. If **NSC** is not satisfied, as in Algorithm 2, a polynomial $f(\mathbf{x})$ s.t. $\phi \models f(\mathbf{x}) \geq 0$ and $\psi \models -f(\mathbf{x}) \geq 0$ can be constructed by letting $f(\mathbf{x}) = (\sum_{i=1}^{r_1} \delta_i f_i) + h_1 = -(\sum_{i=r_1+1}^{r} \delta_i f_i) - h_2, \delta_i \geq 0$, as discussed in Section 4.4. Using Lemma 1, reduce the interpolation problem for ϕ and ψ to a simpler interpolation problem for ϕ' and ψ' with fewer variables.

5.1 Problem formulation

Let $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$ be a finite set of uninterpreted function symbols in *EUF*; further, denote $\Omega_1 \cup \Omega_2$ by Ω_{12} and $\Omega_1 \cup \Omega_3$ by Ω_{13}. Let $\mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]^{\Omega}$ be the extension of $\mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ in which polynomials can have terms built using function symbols in Ω and variables in $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Problem 2: Suppose two formulas ϕ and ψ with $\phi \wedge \psi \models \bot$, where $\phi = f_1 \geq 0 \wedge \ldots \wedge f_{r_1} \geq 0 \wedge g_1 > 0 \wedge \ldots \wedge g_{s_1} > 0$, $\psi = f_{r_1+1} \geq 0 \wedge \ldots \wedge f_r \geq 0 \wedge g_{s_1+1} > 0 \wedge \ldots \wedge g_s > 0$, in which $f_1, \ldots, f_r, g_1, \ldots, g_s$ are all CQ polynomials, $f_1, \ldots, f_{r_1}, g_1, \ldots, g_{s_1} \in \mathbb{R}[\mathbf{x}, \mathbf{y}]^{\Omega_{12}}$, $f_{r_1+1}, \ldots, f_r, g_{s_1+1}, \ldots, g_s \in \mathbb{R}[\mathbf{x}, \mathbf{z}]^{\Omega_{13}}$, the goal is to generate an interpolant I for ϕ and ψ, over the common symbols $\mathbf{x}, \Omega_1$, *i.e.*, I contains only polynomials in $\mathbb{R}[\mathbf{x}]^{\Omega_1}$.

Flatten and Purify: Flatten and purify ϕ and ψ by introducing fresh variables for each term starting with uninterpreted symbols as well as for the terms containing uninterpreted symbols. Keep track of new variables introduced exclusively for ϕ and ψ as well as new common variables.

[6] Under the assumption that SDP tool returns an approximate but correct answer in a fixed number of calls.

[7] The proposed algorithm and its way of handling of combined theories do not crucially depend upon using algorithms in [20]; however, adopting their approach makes proofs and presentation easier by focusing totally on *CQI*.

Let $\overline{\phi} \wedge \overline{\psi} \wedge \bigwedge D$ be obtained from $\phi \wedge \psi$ by flattening and purification where D consists of unit clauses of the form $\omega(c_1, \ldots, c_n) = c$, where $c_1, \ldots, c_n$ are variables and $\omega \in \Omega$. Following [22,20], using the axiom of an uninterpreted function symbol, a set N of Horn clauses are generated as follows, $N = \{\bigwedge_{k=1}^{n} c_k = b_k \rightarrow c = b \mid \omega(c_1, \ldots, c_n) = c \in D, \omega(b_1, \ldots, b_n) = b \in D\}$. The set N is partitioned into $N_\phi, N_\psi, N_{\text{mix}}$ with all symbols in N_ϕ, N_ψ appearing in $\overline{\phi}, \overline{\psi}$, respectively, and N_{mix} consisting of symbols from both $\overline{\phi}, \overline{\psi}$. It is easy to see that for every Horn clause in N_{mix}, each of equalities in the hypothesis as well as the conclusion is also mixed.

$$\phi \wedge \psi \models \bot \text{ iff } \overline{\phi} \wedge \overline{\psi} \wedge D \models \bot \text{ iff } (\overline{\phi} \wedge N_\phi) \wedge (\overline{\psi} \wedge N_\psi) \wedge N_{\text{mix}} \models \bot. \tag{7}$$

Notice that $(\overline{\phi} \wedge N_\phi) \wedge (\overline{\psi} \wedge N_\psi) \wedge N_{\text{mix}} \models \bot$ has no uninterpreted function symbols. If N_{mix} can be replaced by N_{sep}^{ϕ} and N_{sep}^{ψ} as in [20] using separating terms, then IG-CQI can be applied. An interpolant generated for this problem[8] can be used to generate an interpolant for ϕ, ψ after uniformly replacing all new symbols by their corresponding expressions from D.

5.2 Combination algorithm

If N_{mix} is empty, Algorithm 4 invokes Algorithm 2 (IG-CQI) on a finite set of subproblems generated from a disjunction of conjunction of polynomial inequalities by expanding Horn clauses in N_ϕ and N_ψ, and applying De Morgan's rules. The resulting interpolant is a disjunction of conjunction of the interpolants generated for each subproblem.

The case when N_{mix} is nonempty has the same structure as the algorithm $\text{INTER}_{LI(Q)^\Sigma}$ in [20]. The following lemma proves that if a conjunction of polynomial inequalities satisfies **NSC** and an equality on variables can be deduced from it, then it suffices to consider only linear inequalities in the conjunction. This property enables us to use Algorithm $\text{INTER}_{LI(Q)^\Sigma}$ in Figure 3 in [20] for deducing equalities; separating terms for the constants appearing in mixed equalities are computed in the same way as in Algorithm SEP in [20] (Lines 2 and 3 in Algorithm 3 where INTERp, a modified version of $\text{INTER}_{LI(Q)^\Sigma}$, is used solely to deduce equalities and separating terms and not interpolants, thus generating $N_{\text{sep}}^{\phi}, N_{\text{sep}}^{\psi}$). Then Algorithm 4 is called.

Lemma 2. *Let $\overline{\phi}$ and $\overline{\psi}$ be obtained as above satisfying* **NSC**. *If $\overline{\phi} \wedge \overline{\psi}$ is satisfiable, $\overline{\phi} \wedge \overline{\psi} \models c_k = b_k$, then LP($\overline{\phi}$) $\wedge$ LP($\overline{\psi}$) $\models c_k = b_k$, where LP(θ) is the conjunction of the linear constraints in θ.*

If **NSC** is not satisfied, then linear equalities from SOS polynomials h, h_1, h_2 and f as explained above and discussed in Section 4.4 (Lines 6–8 in Algorithm 3) are used to generate simpler subproblems $\overline{\phi'}$ and $\overline{\psi'}$ from $\overline{\phi}$ and $\overline{\psi}$, and Algorithm 3 is recursively called (Lines 9-10 in Algorithm 3).

Theorem 10 (Soundness and Completeness). IG-CQI-EUF *computes an interpolant I of mutually contradictory ϕ, ψ with CQ polynomial inequalities and EUF if it exists.*

[8] after properly handling N_{mix} since Horn clauses have symbols both from $\overline{\phi}$ and $\overline{\psi}$.

Algorithm 3: IG-CQI-EUF

input : $\overline{\phi}$ and $\overline{\psi}$ constructed respective from ϕ and ψ by flattening and purification,
D : definitions of fresh variables introduced during flattening and purifying ϕ, ψ,
N : instances of functionality axioms for functions in D,
$\overline{\phi} = f_1 \geq 0 \wedge \ldots \wedge f_{r_1} \geq 0 \wedge g_1 > 0 \wedge \ldots \wedge g_{s_1} > 0$,
$\overline{\psi} = f_{r_1+1} \geq 0 \wedge \ldots \wedge f_r \geq 0 \wedge g_{s_1+1} > 0 \wedge \ldots \wedge g_s > 0$,
where $\phi \wedge \psi \models \bot$, $f_1, \ldots, f_r, g_1, \ldots, g_s$ are all CQ polynomials,
$f_1, \ldots, f_{r_1}, g_1, \ldots, g_{s_1} \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$, and $f_{r_1+1}, \ldots, f_r, g_{s_1+1}, \ldots, g_s \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$

output: A formula I to be a Craig interpolant for ϕ and ψ

if **NSC** *holds* **then**
  $L_1 := \text{LP}(\overline{\phi})$; $L_2 := \text{LP}(\overline{\psi})$;
  INTERp$(L_1, L_2, N, \emptyset, \emptyset, D, \emptyset)$;
  /* INTERp is a modified version of the $\text{INTER}^{\Sigma}_{\text{LI}(\mathbb{Q})}$ algorithm given in Figure 3 in [20] which is used here to separate every mixed Horn clause in N of the form $\wedge_{i=1}^{n} c_i = d_i \Rightarrow c = d$ into $\wedge_{i=1}^{n} c_i = t_i^+ \Rightarrow c = f(t_1^+, \cdots, t_n^+)$, $\wedge_{i=1}^{n} d_i = t_i^+ \Rightarrow d = f(t_1^+, \cdots, t_n^+)$. It does not call $\text{INTER}_{\mathbf{LI}(Q)}$ to generate an interpolant (line 29 of $\text{INTER}^{\Sigma}_{\mathbf{LI}(\mathbb{Q})}$). When INTERp terminates N_{mix} with initial value N is separated into N_ϕ and N_ψ with entailed equalities in Δ. Because of space limitations, we are not reproducing lines 1-28 of the code in $\text{INTER}^{\Sigma}_{\mathbf{LI}(\mathbb{Q})}$. */
  $\overline{I} := \text{IG-NMIX}(\overline{\phi}, \overline{\psi}, N_\phi, N_\psi)$;
else
  Find $\delta_1, \ldots, \delta_r \geq 0$ and an SOS polynomial h by SDP s.t. $\sum_{i=1}^{r} \delta_i f_i + h \equiv 0$;
  Construct $h_1 \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$ and $h_2 \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$ with form (H1) and (H2);
  $f := \sum_{i=1}^{r_1} \delta_i f_i + h_1 = -\sum_{i=r_1+1}^{r} \delta_i f_i - h_2$;
  Construct $\overline{\phi'}$ and $\overline{\psi'}$ by Theorem 7 by eliminating variables from $h_1 = h_2 = 0$;
  $I' := \text{IG-CQI-EUF}(\overline{\phi'}, \overline{\psi'}, D, N_0)$; $\overline{I} := (f > 0) \vee (f \geq 0 \wedge I')$;
end
Obtain I from $\overline{I}$; **return** I

Algorithm 4: Invariant Generation without N_{mix} (IG-NMIX)

input : $\overline{\phi}$ and $\overline{\psi}$, constructed respectively from ϕ and ψ by flattening and purification,
N_ϕ : instances of functionality axioms for functions in D_ϕ,
N_ψ : instances of functionality axioms for functions in D_ψ,
where $\overline{\phi} \wedge \overline{\psi} \wedge N_\phi \wedge N_\psi \models \bot$

output: A formula I to be a Craig interpolant for ϕ and ψ

Transform $\overline{\phi} \wedge N_\phi$ to a DNF $\vee_i \phi_i$;
Transform $\overline{\psi} \wedge N_\psi$ to a DNF $\vee_j \psi_j$;
return $I := \vee_i \wedge_j \text{IG-CQI}(\phi_i, \psi_j)$

Example 2. Let $\phi := (f_1 = -(y_1 - x_1 + 1)^2 - x_1 + x_2 \geq 0) \wedge (y_2 = \alpha(y_1) + 1) \wedge (g_1 = -x_1^2 - x_2^2 - y_2^2 + 1 > 0)$, $\psi := (f_2 = -(z_1 - x_2 + 1)^2 + x_1 - x_2 \geq 0) \wedge (z_2 = \alpha(z_1) - 1) \wedge (g_2 =$

$-x_1^2-x_2^2-z_2^2+1>0$). Flattening and purification gives $\overline{\phi} := (f_1 \geq 0 \wedge y_2 = y+1 \wedge g_1 > 0)$, $\overline{\psi} := (f_2 \geq 0 \wedge z_2 = z-1 \wedge g_2 > 0)$, where $D = \{y = \alpha(y_1), z = \alpha(z_1)\}$, $N = (y_1 = z_1 \rightarrow y = z)$.

NSC is not satisfied, since $h = -f_1 - f_2 = (y_1 - x_1 + 1)^2 + (z_1 - x_2 + 1)^2$ is an SOS. We follow the steps given in Section 4.4 (Lines 6–8 of IG-CQI-EUF) and obtain $h_1 = (y_1 - x_1 + 1)^2$, $h_2 = (z_1 - x_2 + 1)^2$. This gives $f := f_1 + h_1 = -f_2 - h_2 = -x_1 + x_2$.

By Lemma 1, an interpolant for ϕ, ψ is an interpolant of $((\phi \wedge f > 0) \vee (\phi \wedge f = 0))$ and $((\psi \wedge -f > 0) \vee (\phi \wedge f = 0))$, that is $(f > 0) \vee (f \geq 0 \wedge I_2)$, where I_2 is an interpolant for $\phi \wedge f = 0$ and $\psi \wedge f = 0$. It is easy to see that $\phi \wedge f = 0 \models y_1 = x_1 - 1$, $\psi \wedge f = 0 \models z_1 = x_2 - 1$. Thus, it follows $\overline{\phi'} : -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1$, and $\overline{\psi'} : x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1$.

At Line 10, recursively call IG-CQI-EUF. Now **NSC** holds (Line 1); from linear inequalities in $\overline{\phi'}$ and $\overline{\psi'}$, $y_1 = z_1$ is deduced. Separating terms for y_1, z_1 are constructed by: $\overline{\phi'} \models x_1 - 1 \leq y_1 \leq x_2 - 1, \overline{\psi'} \models x_2 - 1 \leq z_1 \leq x_1 - 1$. Let $t = \alpha(x_2 - 1)$, then $y_1 = z_1 \rightarrow y = z$ is separated into two parts, *i.e.*, $y_1 = t^+ \rightarrow y = t$ and $t^+ = z_1 \rightarrow t = z$. Add them to $\overline{\phi'}$ and $\overline{\psi'}$ respectively, we have $\overline{\phi'}_1 = -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y_1 = x_2 - 1 \rightarrow y = t$, $\overline{\psi'}_1 = x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1 \wedge x_2 - 1 = z_1 \rightarrow t = z$. Then $\overline{\phi'}_1 = -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge (x_2 - 1 > y_1 \vee y_1 > x_2 - 1 \vee y = t)$, $\overline{\psi'}_1 = x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1 \wedge t = z$. Thus, $\overline{\phi'}_1 = \overline{\phi'}_2 \vee \overline{\phi'}_3 \vee \overline{\phi'}_4$, where $\overline{\phi'}_2 = -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge x_2 - 1 > y_1$, $\overline{\phi'}_3 = -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y_1 > x_2 - 1$, $\overline{\phi'}_4 = -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y = t$. Since $\overline{\phi'}_3 = \bot$, it follows $\overline{\phi'}_1 = \overline{\phi'}_2 \vee \overline{\phi'}_4$. Find interpolants $I(\overline{\phi'}_2, \overline{\psi'}_1)$ and $I(\overline{\phi'}_4, \overline{\psi'}_1)$, then $I(\overline{\phi'}_2, \overline{\psi'}_1) \vee I(\overline{\phi'}_4, \overline{\psi'}_1)$ is an interpolant.

6 Implementation and experimental results

We are currently developing a state of the art implementation of the above algorithms using C. In the meantime, for experimentation purposes, we have developed a prototype for putting together existing tools in *Mathematica*. An optimization library *AiSat* [5] built on *CSDP* [1] is used for solving SOS and SDP problems. We give some performance data about this prototype on some examples (see Table 1), which have been evaluated on a 64-bit Linux computer with a 2.93GHz Intel Core-i7 processor and 4GB of RAM.

The performance of the prototype is compared on the same platform to those of three publicly available interpolation procedures for linear-arithmetic cases, *i.e.* Rybalchenko's tool CLP-PROVER in [20], McMillan's procedure FOCI in [17], and Beyer's tool CSISAT in [2]. As Table 2 shows, our approach can successfully solve all these examples rather efficiently. It is especially the completeness and generality that makes the approach competitive for synthesizing interpolants. In particular, the prototype performs, in linear cases, with the same complexity as CSISAT and even better than CLP-PROVER and FOCI. Whilst in nonlinear cases, the method developed in [5] is limited and incomplete even though it works for nonlinear polynomials (using SDP) since it requires bounds on variables as well as uncommon variables are not allowed.

Ex.	Type	Problem	Synthesized Interpolant
5	NLA	$\phi : -y_1 + x_1 - 2 \geq 0 \wedge 2x_2 - x_1 - 1 > 0$ $\wedge -y_1^2 - x_1^2 + 2x_1y_1 - 2y_1 + 2x_1 \geq 0$ $\wedge -y_2^2 - y_1^2 - x_2^2 - 4y_1 + 2x_2 - 4 \geq 0$ $\psi : -z_1 + 2x_2 + 1 \geq 0 \wedge 2x_1 - x_2 - 1 > 0$ $\wedge -z_1^2 - 4x_2^2 + 4x_2z_1 + 3z_1 - 6x_2 - 2 \geq 0$ $\wedge -z_2^2 - x_1^2 - x_2^2 + 2x_1 + z_1 - 2x_2 - 1 \geq 0$	$-x_1 + x_2 > 0$
6	NLA	$\phi : 4 - x^2 - y^2 \geq 0 \wedge y \geq 0 \wedge x + y - 1 > 0$ $\psi : x \geq 0 \wedge 1 - x^2 - (y+1)^2 \geq 0$	$\frac{1}{2}(x^2 + y^2 + 4y) > 0$
7	LA	$\phi : z - x \geq 0 \wedge x - y \geq 0 \wedge -z > 0$ $\psi : x + y \geq 0 \wedge -y \geq 0$	$-0.8x - 0.2y > 0$
8	LA+ *EUF*	$\phi : f(x) \geq 0 \wedge x - y \geq 0 \wedge y - x \geq 0$ $\psi : -f(y) > 0$	$f(y) \geq 0$
9	Ellipsoid	$\phi : -{x_1}^2 + 4x_1 + x_2 - 4 \geq 0$ $\wedge -x_1 - x_2 + 3 - y^2 > 0$ $\psi : -3{x_1}^2 - {x_2}^2 + 1 \geq 0 \wedge x_2 - z^2 \geq 0$	$-3 + 2x_1 + {x_1}^2 + \frac{1}{2}{x_2}^2 > 0$
10	Ellipsoid	$\phi : 4 - (x-1)^2 - 4y^2 \geq 0 \wedge y - \frac{1}{2} \geq 0$ $\psi : 4 - (x+1)^2 - 4y^2 \geq 0 \wedge x + 2y \geq 0$	$-15.93 + 19.30x - 9.65x^2$ $+91.76y - 38.60y^2 > 0$
11	Octagon	$\phi : -3 \leq x \leq 1 \wedge -2 \leq y \leq 2 \wedge -4 \leq x - y \leq 2$ $\wedge -4 \leq x + y \leq 2 \wedge x + 2y + 1 \leq 0$ $\psi : -1 \leq x \leq 3 \wedge -2 \leq y \leq 2 \wedge -2 \leq x - y \leq 4$ $\wedge -2 \leq x + y \leq 4 \wedge 2x - 5y + 6 \leq 0$	$-88.08 - 649.94x$ $-1432.44y > 0$
12	Octagon	$\phi : 2 \leq x \leq 7 \wedge 0 \leq y \leq 3 \wedge 0 \leq x - y \leq 6$ $\wedge 3 \leq x + y \leq 9 \wedge 23 - 3x - 8y \leq 0$ $\psi : 0 \leq x \leq 5 \wedge 2 \leq y \leq 5 \wedge -4 \leq x - y \leq 2$ $\wedge 3 \leq x + y \leq 9 \wedge y - 3x - 2 \leq 0$	$562.10 + 1244.11x$ $-869.83y > 0$

Table 1. The output formulas in the last column have been verified using the approach given in [4] to be the true interpolants w.r.t. their corresponding problems in the third column.

Example	Type	Time (sec)			
		CLP-PROVER	FOCI	CSISAT	Our Approach
Example 1	NLA	–	–	–	0.003
Example 2	NLA+*EUF*	–	–	–	0.036
Example 5	NLA	–	–	–	0.014
Example 6	NLA	–	–	–	0.003
Example 7	LA	0.023	×	0.003	0.003
Example 8	LA+*EUF*	0.025	0.006	0.007	0.003
Example 9	Ellipsoid	–	–	–	0.002
Example 10	Ellipsoid	–	–	–	0.002
Example 11	Octagon	0.059	×	0.004	0.004
Example 12	Octagon	0.065	×	0.004	0.004

– means interpolant generation fails, and × specifies particularly wrong answers (satisfiable).

Table 2. Evaluation results of the presented examples

7 Conclusion

The paper proposes a polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals. Under a technical condition that if no nonpositive constant combination of nonstrict inequalities is a sum of squares polynomials, then such an interpolant can be generated essentially using the linearization of concave quadratic polynomials. Otherwise, if this condition is not satisfied, then the algorithm is recursively called on smaller problems after deducing linear equalities relating variables. The resulting interpolant is a disjunction of conjunction of polynomial inequalities.

Using the hierarchical calculus framework proposed in [22], we give an interpolation algorithm for the combined quantifier-free theory of concave quadratic polynomial inequalities and equality over uninterpreted function symbols. The combination algorithm is patterned after a combination algorithm for the combined theory of linear inequalities and equality over uninterpreted function symbols.

A prototype has been built, and experimental results indicate our approach is applicable to all existing abstract interpretation domains widely used in verification for programs and hybrid systems like *octagon*, *polyhedra*, *ellipsoid* and so on, which is encouraging for using this approach in the state of the art of verification techniques based on interpolation[9].

Acknowledgement

The first three authors are supported partly by NSFC under grants 11290141, 11271034 and 61532019; the fourth and sixth authors are supported partly by "973 Program" under grant No. 2014CB340701, by NSFC under grant 91418204, by CDZ project CAP (GZ 1023), and by the CAS/SAFEA International Partnership Program for Creative Research Teams; the fifth author is supported partly by NSF under grant DMS-1217054 and by the CAS/SAFEA International Partnership Program for Creative Research Teams.

References

1. *CSDP*. `http://projects.coin-or.org/Csdp/`.
2. D. Beyer, D. Zufferey, and R. Majumdar. Interpolation for LA+EUF. In *CAV 2008*, volume 5123 of *Lecture Notes in Computer Science*, pages 304–308, 2008.
3. A. Cimatti, A. Griggio, and R. Sebastiani. Efficient interpolation generation in satisfiability modulo theories. In *TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 397–412, 2008.
4. L. Dai, T. Gan, B. Xia, and N. Zhan. Barrier certificate revisited. *To appear J. of Symbolic Computation*, 2016.
5. L. Dai, B. Xia, and N. Zhan. Generating non-linear interpolants by semidefinite programming. In *CAV 2013*, volume 8044 of *Lecture Notes in Computer Science*, pages 364–380, 2013.

[9] The tool and all case studies can be found at `http://lcs.ios.ac.cn/~chenms/tools/InterCQI_v1.1.tar.bz2`.

6. V. D'Silva, M. Purandare, G. Weissenbacher, and D. Kroening. Interpolant strength. In *VMCAI 2010*, volume 5944 of *Lecture Notes in Computer Science*, pages 129–145, 1997.
7. T. Fujie and M. Kojima. Semidefinite programming relaxation for nonconvex quadratic programs. *J. of Global Optimization*, 10(4):367–380, 1997.
8. T. Gan, L. Dai, Xia B, N. Zhan, D. Kapur, and M. Chen. Interpolation synthesis for quadratic polynomial inequalities and combination with *EUF*. *CoRR*, abs/1601.04802, 2016.
9. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *CAV 1997*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83, 1997.
10. T. Henzinger, R. Jhala, R. Majumdar, and K. McMillan. Abstractions from proofs. In *POPL 2004*, pages 232–244, 2004.
11. Y. Jung, W. Lee, B. Wang, and K. Yi. Predicate generation for learning-based quantifier-free loop invariant inference. In *TACAS 2011*, volume 6605 of *Lecture Notes in Computer Science*, pages 205–219, 2011.
12. D. Kapur, R. Majumdar, and C. Zarba. Interpolation for data structures. In *FSE 2006*, pages 105–116, 2006.
13. L. Kovács and A. Voronkov. Interpolation and symbol eleimination. In *CADE 2009*, volume 5663 of *Lecture Notes in Computer Science*, pages 199–213, 2009.
14. J. Krajíček. Interpolation theorems, lower bounds for proof systems, and independence results for bounded arithmetic. *J. of Symbolic Logic*, 62(2):457–486, 1997.
15. M. Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging Applications of Algebraic Geometry*, volume 149 of *The IMA Volumes in Mathematics and its Applications*, pages 157–270. 2009.
16. K. McMillan. Interpolation and sat-based model checking. In *CAV 2003*, volume 3920 of *Lecture Notes in Computer Science*, pages 1–13, 2003.
17. K. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.
18. K. McMillan. Quantified invariant generation using interpolation saturation prover. In *TACAS 2008*, volume 4963 of *Lecture Notes in Computer Science*, pages 413–427, 2008.
19. P. Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *J. of Symbolic Logic*, 62(3):981–998, 1997.
20. A. Rybalchenko and V. Sofronie-Stokkermans. Constraint solving for interpolation. *J. Symb. Comput.*, 45(11):1212–1233, 2010.
21. A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.
22. V. Sofronie-Stokkermans. Interpolation in local theory extensions. *Logical Methods in Computer Science*, 4(4), 2008.
23. G. Stengle. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Ann. Math.*, 207:87–97, 1974.
24. R. H. Tütüncü, K. C. Toh, and M. J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *J. of Mathematical programming*, 95(2):189–217, 2003.
25. Z. Yang, W. Lin, and M. Wu. Exact safety verification of hybrid systems based on bilinear SOS representation. *ACM Trans. Embedded Comput. Syst.*, 14(1):16:1–16:19, 2015.
26. G. Yorsh and M. Musuvathi. A combination method for generating interpolants. In *CADE'05*, volume 3632 of *LNCS*, pages 353–368.
27. H. Zhao, N. Zhan, and D. Kapur. Synthesizing switching controllers for hybrid systems by generating invariants. In *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, volume 8051 of *Lecture Notes in Computer Science*, pages 354–373, 2013.
28. H. Zhao, N. Zhan, D. Kapur, and K. Larsen. A "hybrid" approach for synthesizing optimal controllers of hybrid systems: A case study of the oil pump industrial example. In *FM 2012*, volume 7436 of *Lecture Notes in Computer Science*, pages 471–485. 2012.