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]
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

IJCAR 2016, June 27-30, 2016

Outline

1. An overview of our approach
   - Motivation
   - Generalization of Motzkin's transposition theorem
   - Concave quadratic polynomials
2. Generating interpolants for concave quadratic polynomial inequalities
   - **NSC** condition: generalized Motzkin's theorem applies
   - **SOS** (**NSC** not satisifed): equalities from expressions in a sum of squares being equal to 0.
3. Combination with equality logic with uninterpreted functions (EUF)
   - similar to the linear case
4. Concluding remarks

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).
  - ...

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).
  - ...
- Craig interpolant synthesis plays a central role in interpolation-based techniques, but most of existing work concerns decidable fragments of first-order logic, linear arithmetic, EUF and their combinations.

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).
  - ...
- Craig interpolant synthesis plays a central role in interpolation-based techniques, but most of existing work concerns decidable fragments of first-order logic, linear arithmetic, EUF and their combinations.
- Non-linear constraints are very common in the verification of programs and hybrid systems, but little work

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).
  - ...
- Craig interpolant synthesis plays a central role in interpolation-based techniques, but most of existing work concerns decidable fragments of first-order logic, linear arithmetic, EUF and their combinations.
- Non-linear constraints are very common in the verification of programs and hybrid systems, but little work
  - Dai et al. [Dai,Xia & Zhan 13] proposed an approach to constructing non-linear interpolants based on **Nullstellensatz theorem** using *SDP*.

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).
  - ...
- Craig interpolant synthesis plays a central role in interpolation-based techniques, but most of existing work concerns decidable fragments of first-order logic, linear arithmetic, EUF and their combinations.
- Non-linear constraints are very common in the verification of programs and hybrid systems, but little work
  - Dai et al. [Dai,Xia & Zhan 13] proposed an approach to constructing non-linear interpolants based on **Nullstellensatz theorem** using SDP.
  - Too expensive on dealing with uncommon variables by quantifier elimination.

Motivation

- Interpolation-based techniques can scale up existing verification techniques of programs like model-checking, theorem proving, ...
- Related work
  - By Krajíček [Krajíček 97] and Pudlák [Pudlák 97] in connection with theorem proving.
  - By McMillan [McMillan 03] in connection with model-checking.
  - By Graf and Saïdi [Graf& Saïdi 97], McMillan [McMillan 05] and Henzinger et al. [Henzinger et al. 04] in connection with CEGAR.
  - By Rybalchenko et al. [Rybalchenko & Sofronie-Stokkermans 10] on the combined theory of linear arithmetic and equality logic with uninterpreted functions (EUF).
  - ...
- Craig interpolant synthesis plays a central role in interpolation-based techniques, but most of existing work concerns decidable fragments of first-order logic, linear arithmetic, EUF and their combinations.
- Non-linear constraints are very common in the verification of programs and hybrid systems, but little work
  - Dai et al. [Dai,Xia & Zhan 13] proposed an approach to constructing non-linear interpolants based on **Nullstellensatz theorem** using SDP.
  - Too expensive on dealing with uncommon variables by quantifier elimination.
- **Our goal**: to investigate new efficient approach to Craig interpolant synthesis for nonlinear arithmetic.

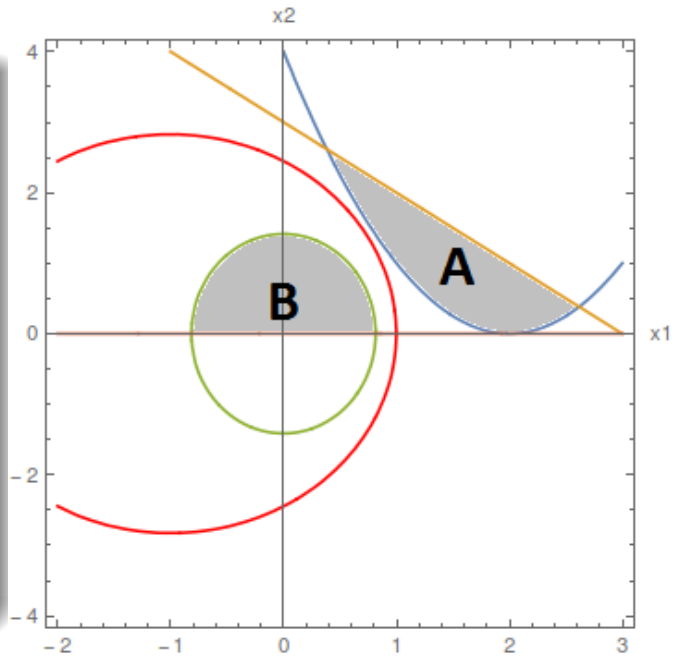
Overview

Example (running example)

Consider two formulas A and B with $A \wedge B \models \bot$, where

$$A := -{x_1}^2 + 4x_1 + x_2 - 4 \geq 0 \wedge -x_1 - x_2 + 3 - y^2 > 0,$$
$$B := -3{x_1}^2 - {x_2}^2 + 1 \geq 0 \wedge x_2 - z^2 \geq 0$$

We aim to generate an interpolant I for A and B, on the common variables (x_1 and x_2), such that $A \models I$ and $I \wedge B \models \bot$.



An intuitive description of a candidate interpolant is as the red curve (its boundary) in the above right figure, which separates A and B in the panel of x_1 and x_2.

Overview

- A polynomial time algorithm based on SDP for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:

Overview

- A polynomial time algorithm based on SDP for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  - If no nonpositive constant combination of nonstrict inequalities is a sum of squares polynomial, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials.

Overview

- A polynomial time algorithm based on SDP for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  - If no nonpositive constant combination of nonstrict inequalities is a sum of squares polynomial, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials.
  - Otherwise, linear equalities relating variables are deduced, resulting to interpolation subproblems with fewer variables on which the algorithm is recursively applied.

Overview

- A polynomial time algorithm based on SDP for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  - If no nonpositive constant combination of nonstrict inequalities is a sum of squares polynomial, an interpolant a la McMillan can be generated essentially using the linearization of quadratic polynomials.
  - Otherwise, linear equalities relating variables are deduced, resulting to interpolation subproblems with fewer variables on which the algorithm is recursively applied.
- An algorithm based on hierarchical reasoning for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and equality logic with uninterpreted functions (EUF).

Theorem (Motzkin's transposition theorem)

Let A and B be matrices and let $\vec{\alpha}$ and $\vec{\beta}$ be column vectors. Then there exists a vector $\mathbf{x}$ with $A\mathbf{x} - \vec{\alpha} \geq 0$ and $B\mathbf{x} - \vec{\beta} > 0$, iff for all row vectors $\mathbf{y}, \mathbf{z} \geq 0$:

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

Theorem (Motzkin's transposition theorem)

Let A and B be matrices and let $\vec{\alpha}$ and $\vec{\beta}$ be column vectors. Then there exists a vector $\mathbf{x}$ with $A\mathbf{x} - \vec{\alpha} \geq 0$ and $B\mathbf{x} - \vec{\beta} > 0$, iff for all row vectors $\mathbf{y}, \mathbf{z} \geq 0$:

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

Corollary

Let $A \in \mathbb{R}^{r \times n}$ and $B \in \mathbb{R}^{s \times n}$ be matrices and $\vec{\alpha} \in \mathbb{R}^r$ and $\vec{\beta} \in \mathbb{R}^s$ 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 $A\mathbf{x} - \vec{\alpha} \geq 0$ and $B\mathbf{x} - \vec{\beta} > 0$, iff there exist real numbers $\lambda_1, \ldots, \lambda_r \geq 0$ and $\eta_0, \eta_1, \ldots, \eta_s \geq 0$ such that

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

Concave quadratic polynomials

Definition (Concave Quadratic)

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

- f has total degree at most 2, i.e., it has the form $f = \mathbf{x}^T A\mathbf{x} + 2\vec{\alpha}^T\mathbf{x} + a$, where A is a real symmetric matrix, $\vec{\alpha}$ is a column vector and $a \in \mathbb{R}$;
- the matrix A is negative semi-definite ($A \preceq 0$), i.e., for any $\mathbf{x} \in \mathbb{R}^n, \mathbf{x}^T A\mathbf{x} \leq 0$.

Example

Take $f = -3{x_1}^2 - {x_2}^2 + 1$ in the running example, which is from the ellipsoid domain and can be expressed as

$$f = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^T \begin{pmatrix} -3 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + 1.$$

The corresponding $A = \begin{pmatrix} -3 & 0 \\ 0 & -1 \end{pmatrix} \preceq 0$. Thus, f is CQ.

Concave quadratic polynomials

- 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.

Concave quadratic polynomials

- 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 $f(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} + 2\vec{\alpha}^T\mathbf{x} + a$ can also be represented as an inner product of matrices, i.e., $\left\langle P, \begin{pmatrix} 1 & \mathbf{x}^T \\ \mathbf{x} & \mathbf{x}\mathbf{x}^T \end{pmatrix} \right\rangle$, where $P = \begin{pmatrix} a & \alpha^T \\ \alpha & A \end{pmatrix}$, $\langle (a_{ij}), (b_{ij}) \rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} * b_{ij}$.

Linearization of CQ polynomials

Definition (Linearization)

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

Linearization of CQ polynomials

Definition (Linearization)

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

let

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

$$K_1 \triangleq \{\mathbf{x} \mid \begin{pmatrix} 1 & \mathbf{x}^T \\ \mathbf{x} & \vec{X} \end{pmatrix} \succeq 0, \ \wedge_{i=1}^{r} \left\langle P_i, \begin{pmatrix} 1 & \mathbf{x}^T \\ \mathbf{x} & \vec{X} \end{pmatrix} \right\rangle \geq 0,$$

$$\wedge_{j=1}^{s} \left\langle Q_j, \begin{pmatrix} 1 & \mathbf{x}^T \\ \mathbf{x} & \vec{X} \end{pmatrix} \right\rangle > 0, \text{ for some } \vec{X}\}, \tag{3}$$

Linearization of CQ polynomials

Theorem

Let $f_1, \ldots, f_r$ and $g_1, \ldots, g_s$ be CQ polynomials, K and K_1 as above, then $K = K_1$.

Therefore, 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.

Problem formulation

Problem 1

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

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

in which $f_1, \ldots, f_r, g_1, \ldots, g_s$ are all CQ, $f_1, \ldots, f_{r_1}, g_1, \ldots, g_{s_1} \in \mathbb{R}[\mathbf{x}, \mathbf{y}]$, $f_{r_1+1}, \ldots, f_r, g_{s_1+1}, \ldots, g_s \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$, develop an algorithm to generate a (reverse) Craig interpolant I for ϕ and ψ, on the common variables of ϕ and ψ, such that $\phi \models I$ and $I \wedge \psi \models \bot$.

$\mathbf{x} = (x_1, \ldots, x_d)$, $\mathbf{y} = (y_1, \ldots, y_u)$ and $\mathbf{z} = (z_1, \ldots, z_v)$, where $d + u + v = n$.

NSC Condition

Definition (NSC)

Formulas ϕ and ψ in Problem 1, satisfy the non-existence of an SOS polynomial condition (**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.

Example

Formulas A and B in the running example do not satisfy **NSC**, since there exist $\delta_1 = 1, \delta_2 = 1, \delta_3 = 1$, s.t.

$$\begin{aligned} &- (\delta_1(-{x_1}^2 + 4x_1 + x_2 - 4) + \delta_2(-3{x_1}^2 - {x_2}^2 + 1) + \delta_3(x_2 - z^2)) \\ &= (2x_1 - 1)^2 + (x_2 - 1)^2 + z^2 \end{aligned}$$

is a non-zero SOS.

When NSC is satisfied

Theorem (Generalization of Motzkin's Transposition Theorem)

Let ϕ and ψ as defined in Problem 1 with $\phi \wedge \varphi \models \perp$, which satisfy **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 $h_2 \in \mathbb{R}[\mathbf{x}, \mathbf{z}]$ s.t.*

$$\sum_{i=1}^{r} \lambda_i f_i + \sum_{j=1}^{s} \eta_j g_j + \eta_0 + h_1 + h_2 \equiv 0, \tag{4}$$

$$\eta_0 + \eta_1 + \ldots + \eta_s = 1. \tag{5}$$

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

Computing interpolant using semi-definite programming

Let $W = \begin{pmatrix} 1 & \mathbf{x}^T & \mathbf{y}^T & \mathbf{z}^T \\ \mathbf{x} & \mathbf{x}\mathbf{x}^T & \mathbf{x}\mathbf{y}^T & \mathbf{x}\mathbf{z}^T \\ \mathbf{y} & \mathbf{y}\mathbf{x}^T & \mathbf{y}\mathbf{y}^T & \mathbf{y}\mathbf{z}^T \\ \mathbf{z} & \mathbf{z}\mathbf{x}^T & \mathbf{z}\mathbf{y}^T & \mathbf{z}\mathbf{z}^T \end{pmatrix}$, $f_i = \langle P_i, W \rangle$, $g_j = \langle Q_j, W \rangle$, where P_i and Q_j are $(n+1) \times (n+1)$ matrices, $h_1 = \langle M, W \rangle$, $h_2 = \langle \hat{M}, W \rangle$, and $M = (M_{ij})_{4\times 4}$, $\hat{M} = (\hat{M}_{ij})_{4\times 4}$ 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**, computing the interpolant is reduced to the following **SDP** feasibility problem:

Computing interpolant using semi-definite programming

Find: $\lambda_1, \ldots, \lambda_r, \eta_1, \ldots, \eta_s \in \mathbb{R}$, $M, \hat{M} \in \mathbb{R}^{(n+1)\times(n+1)}$ subject to

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

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

Generating interpolants when NSC holds

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.

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

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

When NSC is not satisfied

If ϕ and ψ do not satisfy **NSC**, i.e., an SOS polynomial $h(\mathbf{x},\mathbf{y},\mathbf{z}) = -(\sum_{i=1}^{r} \delta_i f_i)$ can be computed which can be split into two SOS polynomials $h_1(\mathbf{x},\mathbf{y})$ and $h_2(\mathbf{x},\mathbf{z})$. Then an SOS polynomial $f(\mathbf{x})$ such that $\phi \models f(\mathbf{x}) \geq 0$ and $\psi \models -f(\mathbf{x}) \geq 0$ can be constructed as

$$f(\mathbf{x}) = (\sum_{i=1}^{r_1} \delta_i f_i) + h_1 = -(\sum_{i=r_1+1}^{r} \delta_i f_i) - h_2, \delta_i \geq 0.$$

Lemma

If Problem 1 does not satisfy **NSC**, *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,*

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

When NSC is not satisfied

Theorem

With ϕ, ψ, $\phi_1, \phi_2, \psi_1, \psi_2$ as in previous Lemma, from an interpolant $I_{2,2}$ for ϕ_2 and ψ_2, $I := (f > 0) \vee (f \geq 0 \wedge I_{2,2})$ is an interpolant for ϕ and ψ.

When NSC is not satisfied

Theorem

With ϕ, ψ, $\phi_1, \phi_2, \psi_1, \psi_2$ as in previous Lemma, from an interpolant $I_{2,2}$ for ϕ_2 and ψ_2, $I := (f > 0) \vee (f \geq 0 \wedge I_{2,2})$ is an interpolant for ϕ and ψ.

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

Theorem

With ϕ, ψ, $\phi_1, \phi_2, \psi_1, \psi_2$ as in previous Lemma and h has $a_{n+1} > 0$, $f > 0$ is an interpolant for ϕ and ψ.

In case h does not have a constant, i.e., $a_{n+1} = 0$, equalities can be reduced from $h_1(\mathbf{x}, \mathbf{y}) = 0$, which can be used to eliminate at least one variable in ϕ_2; similarly, at least one variable in ψ_2 can be eliminated from $h_2(\mathbf{x}, \mathbf{z}) = 0$.

Generating interpolants for CQI

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** $Var(\phi) \subseteq Var(\psi)$ **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** $I := \text{IG-NSC}(\phi, \psi)$; **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' by eliminating variables due to $h_1 = h_2 = 0$;

7 $I' := \text{IG-CQI}(\phi', \psi')$;

8 $I := (f > 0) \vee (f \geq 0 \wedge I')$;

9 **return** I

Generating interpolants for CQI

Example

Recall the running example where

$$
\begin{aligned}
h &= (2x_1 - 1)^2 + (x_2 - 1)^2 + z^2 \\
&= \underbrace{\frac{1}{2}((2x_1 - 1)^2 + (x_2 - 1)^2)}_{h_1} + \underbrace{\frac{1}{2}((2x_1 - 1)^2 + (x_2 - 1)^2) + z^2}_{h_2} \\
f &= \delta_1(-{x_1}^2 + 4x_1 + x_2 - 4) + h_1 \\
&= -3 + 2x_1 + {x_1}^2 + \frac{1}{2}{x_2}^2
\end{aligned}
$$

We construct A' from A by setting $x_1 = \frac{1}{2}, x_2 = 1$ derived from $h_1 = 0$; similarly B' is constructed by setting $x_1 = \frac{1}{2}, x_2 = 1, z = 0$ in B as derived from $h_2 = 0$. It follows that, $A' := B' := \bot$ Thus, $I(A', B') := (0 > 0)$ is an interpolant for (A', B').

An interpolant for A and B is thus $(f(x) > 0) \vee (f(x) = 0 \wedge I(A', B'))$, i.e.

$$-3 + 2x_1 + {x_1}^2 + \frac{1}{2}{x_2}^2 > 0,$$

whose boundary corresponds to the red curve mentioned previously.

Combination with EUF

- $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$: a finite set of uninterpreted function symbols in *EUF*;

Combination with EUF

- $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$: a finite set of uninterpreted function symbols in *EUF*;
- $\Omega_{12} = \Omega_1 \cup \Omega_2$, $\Omega_{13} = \Omega_1 \cup \Omega_3$;

Combination with EUF

- $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$: a finite set of uninterpreted function symbols in *EUF*;
- $\Omega_{12} = \Omega_1 \cup \Omega_2$, $\Omega_{13} = \Omega_1 \cup \Omega_3$;
- $\mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]^{\Omega}$: 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 $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Combination with EUF

- $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$: a finite set of uninterpreted function symbols in *EUF*;
- $\Omega_{12} = \Omega_1 \cup \Omega_2$, $\Omega_{13} = \Omega_1 \cup \Omega_3$;
- $\mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]^{\Omega}$: 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 $\mathbf{x}, \mathbf{y}, \mathbf{z}$.

Combination with EUF

- $\Omega = \Omega_1 \cup \Omega_2 \cup \Omega_3$: a finite set of uninterpreted function symbols in *EUF*;
- $\Omega_{12} = \Omega_1 \cup \Omega_2$, $\Omega_{13} = \Omega_1 \cup \Omega_3$;
- $\mathbb{R}[\mathbf{x}, \mathbf{y}, \mathbf{z}]^{\Omega}$: 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

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

where $f_1, \ldots, f_r, g_1, \ldots, g_s$ 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, expressed using the common symbols $\mathbf{x}, \Omega_1$, i.e., I includes only polynomials in $\mathbb{R}[\mathbf{x}]^{\Omega_1}$.

Sketch of the idea (Algorithm IG-CQI-EUF)

1. Flatten and purify the formulas ϕ and ψ as $\overline{\phi}$ and $\overline{\psi}$ 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.

Sketch of the idea (Algorithm IG-CQI-EUF)

1. Flatten and purify the formulas ϕ and ψ as $\overline{\phi}$ and $\overline{\psi}$ 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.
2. Generate a set N of Horn clauses as

$$N = \{\bigwedge_{k=1}^{n} c_k = b_k \rightarrow c = b \mid \omega(c_1, \ldots, c_n) = c \in D, \omega(b_1, \ldots, b_n) = b \in D\},$$

where D consists of unit clauses of the form $\omega(c_1, \ldots, c_n) = c$, with $c_1, \ldots, c_n$ be variables and $\omega \in \Omega$.

Sketch of the idea (Algorithm IG-CQI-EUF)

1. Flatten and purify the formulas ϕ and ψ as $\overline{\phi}$ and $\overline{\psi}$ 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.
2. Generate a set N of Horn clauses as

   $$N = \{\textstyle\bigwedge_{k=1}^{n} c_k = b_k \rightarrow c = b \mid \omega(c_1, \ldots, c_n) = c \in D, \omega(b_1, \ldots, b_n) = b \in D\},$$

   where D consists of unit clauses of the form $\omega(c_1, \ldots, c_n) = c$, with $c_1, \ldots, c_n$ be variables and $\omega \in \Omega$.
3. Partition N into N_ϕ, N_ψ, and N_{mix} with all symbols in N_ϕ, N_ψ appearing in $\overline{\phi}$, $\overline{\psi}$, respectively, and N_{mix} consisting of symbols from both $\overline{\phi}, \overline{\psi}$.

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

Sketch of the idea (Algorithm IG-CQI-EUF)

1. Flatten and purify the formulas ϕ and ψ as $\overline{\phi}$ and $\overline{\psi}$ 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.
2. Generate a set N of Horn clauses as

   $$N = \{\textstyle\bigwedge_{k=1}^{n} c_k = b_k \rightarrow c = b \mid \omega(c_1, \ldots, c_n) = c \in D, \omega(b_1, \ldots, b_n) = b \in D\},$$

   where D consists of unit clauses of the form $\omega(c_1, \ldots, c_n) = c$, with $c_1, \ldots, c_n$ be variables and $\omega \in \Omega$.
3. Partition N into N_ϕ, N_ψ, and N_{mix} with all symbols in N_ϕ, N_ψ appearing in $\overline{\phi}$, $\overline{\psi}$, respectively, and N_{mix} consisting of symbols from both $\overline{\phi}, \overline{\psi}$.

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

4. If N_{mix} is empty, then $\mathrm{IG\text{-}CQI}$ can be applied.

Sketch of the idea (Algorithm IG-CQI-EUF)

1. Flatten and purify the formulas ϕ and ψ as $\overline{\phi}$ and $\overline{\psi}$ 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.
2. Generate a set N of Horn clauses as

   $$N = \{\textstyle\bigwedge_{k=1}^{n} c_k = b_k \rightarrow c = b \mid \omega(c_1, \ldots, c_n) = c \in D, \omega(b_1, \ldots, b_n) = b \in D\},$$

   where D consists of unit clauses of the form $\omega(c_1, \ldots, c_n) = c$, with $c_1, \ldots, c_n$ be variables and $\omega \in \Omega$.
3. Partition N into N_ϕ, N_ψ, and N_{mix} with all symbols in N_ϕ, N_ψ appearing in $\overline{\phi}$, $\overline{\psi}$, respectively, and N_{mix} consisting of symbols from both $\overline{\phi}, \overline{\psi}$.

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

4. If N_{mix} is empty, then $\mathrm{IG\text{-}CQI}$ can be applied.
5. Otherwise, if **NSC** holds, separating terms can be computed as in [Rybalchenko & Sofronie-Stokkermans 10] by solving linear inequality system, and thus N_{mix} can be replaced by N_{sep}^{ϕ} and N_{sep}^{ψ}.

Sketch of the idea (Algorithm IG-CQI-EUF)

1. Flatten and purify the formulas ϕ and ψ as $\overline{\phi}$ and $\overline{\psi}$ 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.
2. Generate a set N of Horn clauses as

   $$N = \{\textstyle\bigwedge_{k=1}^{n} c_k = b_k \rightarrow c = b \mid \omega(c_1, \ldots, c_n) = c \in D, \omega(b_1, \ldots, b_n) = b \in D\},$$

   where D consists of unit clauses of the form $\omega(c_1, \ldots, c_n) = c$, with $c_1, \ldots, c_n$ be variables and $\omega \in \Omega$.
3. Partition N into N_ϕ, N_ψ, and N_{mix} with all symbols in N_ϕ, N_ψ appearing in $\overline{\phi}$, $\overline{\psi}$, respectively, and N_{mix} consisting of symbols from both $\overline{\phi}, \overline{\psi}$.

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

4. If N_{mix} is empty, then $\mathrm{IG\text{-}CQI}$ can be applied.
5. Otherwise, if **NSC** holds, separating terms can be computed as in [Rybalchenko & Sofronie-Stokkermans 10] by solving linear inequality system, and thus N_{mix} can be replaced by N_{sep}^{ϕ} and N_{sep}^{ψ}.
6. Otherwise, a subproblem with fewer variables can be reduced similarly, and the algorithm is recursively called.

An illustrating example

Example

$$\begin{aligned}\phi :=& (f_1 = -(y_1 - x_1 + 1)^2 - x_1 + x_2 \geq 0) \wedge (y_2 = \alpha(y_1) + 1) \\ & \wedge (g_1 = -x_1^2 - x_2^2 - y_2^2 + 1 > 0), \\ \psi :=& (f_2 = -(z_1 - x_2 + 1)^2 + x_1 - x_2 \geq 0) \wedge (z_2 = \alpha(z_1) - 1) \\ & \wedge (g_2 = -x_1^2 - x_2^2 - z_2^2 + 1 > 0).\end{aligned}$$

An illustrating example

Example

$$\begin{aligned}\phi :=& (f_1 = -(y_1 - x_1 + 1)^2 - x_1 + x_2 \geq 0) \wedge (y_2 = \alpha(y_1) + 1) \\ & \wedge (g_1 = -x_1^2 - x_2^2 - y_2^2 + 1 > 0), \\ \psi :=& (f_2 = -(z_1 - x_2 + 1)^2 + x_1 - x_2 \geq 0) \wedge (z_2 = \alpha(z_1) - 1) \\ & \wedge (g_2 = -x_1^2 - x_2^2 - z_2^2 + 1 > 0).\end{aligned}$$

1. Flattening and purification gives

$$\overline{\phi} := (f_1 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0), \quad \overline{\psi} := (f_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0).$$

where $D = \{y = \alpha(y_1), z = \alpha(z_1)\}$, $N = (y_1 = z_1 \rightarrow y = z)$.

An illustrating example

Example

$$\begin{aligned}\phi :=& (f_1 = -(y_1 - x_1 + 1)^2 - x_1 + x_2 \geq 0) \wedge (y_2 = \alpha(y_1) + 1) \\ & \wedge (g_1 = -x_1^2 - x_2^2 - y_2^2 + 1 > 0), \\ \psi :=& (f_2 = -(z_1 - x_2 + 1)^2 + x_1 - x_2 \geq 0) \wedge (z_2 = \alpha(z_1) - 1) \\ & \wedge (g_2 = -x_1^2 - x_2^2 - z_2^2 + 1 > 0).\end{aligned}$$

1. Flattening and purification gives

$$\overline{\phi} := (f_1 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0), \quad \overline{\psi} := (f_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0).$$

where $D = \{y = \alpha(y_1), z = \alpha(z_1)\}$, $N = (y_1 = z_1 \to y = z)$.

2. **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. $h_1 = (y_1 - x_1 + 1)^2$, $h_2 = (z_1 - x_2 + 1)^2$. This gives

$$f := f_1 + h_1 = -f_2 - h_2 = -x_1 + x_2.$$

An illustrating example

3. An interpolant for $\overline{\phi}, \overline{\psi}$ is an interpolant of $((\overline{\phi} \wedge f > 0) \vee (\overline{\phi} \wedge f = 0))$ and $((\overline{\psi} \wedge -f > 0) \vee (\overline{\phi} \wedge f = 0))$ which simplifies to: $(f > 0) \vee (f \geq 0 \wedge I_2)$ where I_2 is an interpolant for $\overline{\phi} \wedge f = 0$ and $\overline{\psi} \wedge f = 0$. After replacing y_1 with $x_1 - 1$ in $\overline{\phi} \wedge f = 0$ and z_1 with $x_2 - 1$ in $\overline{\psi} \wedge f = 0$, we get

$$\overline{\phi'} := -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1,$$

$$\overline{\psi'} := x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1.$$

An illustrating example

3. An interpolant for $\overline{\phi}, \overline{\psi}$ is an interpolant of $((\overline{\phi} \wedge f > 0) \vee (\overline{\phi} \wedge f = 0))$ and $((\overline{\psi} \wedge -f > 0) \vee (\overline{\phi} \wedge f = 0))$ which simplifies to: $(f > 0) \vee (f \geq 0 \wedge I_2)$ where I_2 is an interpolant for $\overline{\phi} \wedge f = 0$ and $\overline{\psi} \wedge f = 0$. After replacing y_1 with $x_1 - 1$ in $\overline{\phi} \wedge f = 0$ and z_1 with $x_2 - 1$ in $\overline{\psi} \wedge f = 0$, we get

$$\overline{\phi'} := -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1,$$
$$\overline{\psi'} := x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1.$$

4. Recursively call IG-CQI-EUF until **NSC** is satisfied. $y_1 = z_1$ is deduced from linear inequalities in $\overline{\phi'}$ and $\overline{\psi'}$, and separating terms for y_1, z_1 are constructed:

$$\overline{\phi'} \models x_1 - 1 \leq y_1 \leq x_2 - 1, \quad \overline{\psi'} \models x_2 - 1 \leq z_1 \leq x_1 - 1.$$

Let $t = \alpha(x_2 - 1)$, then separate $y_1 = z_1 \rightarrow y = z$ into two parts:

$$y_1 = t^+ \rightarrow y = t, \quad t^+ = z_1 \rightarrow t = z.$$

Adding them to $\overline{\phi'}$ and $\overline{\psi'}$ respectively, we have

$$\overline{\phi'}_1 := -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y_1 = x_2 - 1 \rightarrow y = t,$$
$$\overline{\psi'}_1 := x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1 \wedge x_2 - 1 = z_1 \rightarrow t = z.$$

An illustrating example

4 Then

$$\begin{aligned}\overline{\phi'}_1 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \\ & \wedge (x_2 - 1 > y_1 \vee y_1 > x_2 - 1 \vee y = t), \\ \overline{\psi'}_1 :=& x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1 \wedge t = z.\end{aligned}$$

Thus,

$$\begin{aligned}\overline{\phi'}_1 :=& \overline{\phi'}_2 \vee \overline{\phi'}_3 \vee \overline{\phi'}_4, \textit{where} \\ \overline{\phi'}_2 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge x_2 - 1 > y_1, \\ \overline{\phi'}_3 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y_1 > x_2 - 1, \\ \overline{\phi'}_4 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y = t.\end{aligned}$$

Since $\overline{\phi'}_3 = \textit{false}$, then $\overline{\phi'}_1 = \overline{\phi'}_2 \vee \overline{\phi'}_4$. Then find interpolant $I(\overline{\phi'}_2, \overline{\psi'}_1)$ and $I(\overline{\phi'}_4, \overline{\psi'}_1)$.

An illustrating example

4. Then

$$\begin{aligned}\overline{\phi'}_1 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \\ & \wedge (x_2 - 1 > y_1 \vee y_1 > x_2 - 1 \vee y = t), \\ \overline{\psi'}_1 :=& x_1 - x_2 \geq 0 \wedge z_2 = z - 1 \wedge g_2 > 0 \wedge z_1 = x_2 - 1 \wedge t = z.\end{aligned}$$

Thus,

$$\begin{aligned}\overline{\phi'}_1 :=& \overline{\phi'}_2 \vee \overline{\phi'}_3 \vee \overline{\phi'}_4, \textit{ where} \\ \overline{\phi'}_2 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge x_2 - 1 > y_1, \\ \overline{\phi'}_3 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y_1 > x_2 - 1, \\ \overline{\phi'}_4 :=& -x_1 + x_2 \geq 0 \wedge y_2 = y + 1 \wedge g_1 > 0 \wedge y_1 = x_1 - 1 \wedge y = t.\end{aligned}$$

Since $\overline{\phi'}_3 = \textit{false}$, then $\overline{\phi'}_1 = \overline{\phi'}_2 \vee \overline{\phi'}_4$. Then find interpolant $I(\overline{\phi'}_2, \overline{\psi'}_1)$ and $I(\overline{\phi'}_4, \overline{\psi'}_1)$.

5. Finally we conclude that $I(\overline{\phi'}_2, \overline{\psi'}_1) \vee I(\overline{\phi'}_4, \overline{\psi'}_1)$ is an interpolant.

Implementation

- We have developed a prototype for putting together existing tools in *Mathematica*. An optimization library *AiSat* built on *CSDP* is used for solving SOS and SDP problems.
- We are currently developing a state of the art implementation of the above algorithms using C.

Evaluation results

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).

Concluding remarks

- Contributions

Concluding remarks

- Contributions
  1. A complete (relative to numeric computation), polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:

Concluding remarks

- Contributions
  1. A complete (relative to numeric computation), polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  2. An algorithm for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and *EUF*.

Concluding remarks

- Contributions
  1. A complete (relative to numeric computation), polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  2. An algorithm for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and *EUF*.
- Future work

Concluding remarks

- Contributions
  1. A complete (relative to numeric computation), polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  2. An algorithm for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and *EUF*.
- Future work
  - More efficient implementation.

Concluding remarks

- Contributions
  1. A complete (relative to numeric computation), polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  2. An algorithm for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and *EUF*.
- Future work
  - More efficient implementation.
  - To extend the proposed framework beyond concave quadratic polynomials.

Concluding remarks

- Contributions
  1. A complete (relative to numeric computation), polynomial time algorithm for generating interpolants from mutually contradictory conjunctions of concave quadratic polynomial inequalities over the reals:
  2. An algorithm for generating interpolants for the combination of quantifier-free theory of concave quadratic polynomial inequalities and *EUF*.
- Future work
  - More efficient implementation.
  - To extend the proposed framework beyond concave quadratic polynomials.
  - To investigate how results reported for nonlinear polynomial inequalities based on positive nullstellensatz and the Archimedian condition on variables can be exploited in the proposed framework for dealing with polynomial inequalities.