Non-linear Interpolant Generation and Its Application to Program Verification

Naijun Zhan

State Key Laboratory of Computer Science, Institute of Software, CAS

Joint work with Liyun Dai, Ting Gan, Bow-Yaw Wang, Bican Xia, and Hengjun Zhao

Probabilistic and Hybrid Workshop
Sept. 24-27, 2013

Part I: Non-linear Interpolant Generation

Motivation

- Current program verification techniques suffer from scalability.
- Compositional way has been thought as an effective solution to the problem.
- Interpolation-based techniques are inherently local and modular, which can be used to scale up these techniques of program verification:
    - Theorem proving: Nelson-Oppen method, SMT;
    - Model-checking: BMC, CEGAR;
    - Abstraction interpretation;
    - Machine learning based approaches.
- Synthesizing Craig interpolants is the cornerstone of interpolation based techniques.

Related work on synthesizing Craig interpolants

- [McMillan 05] on quantifier-free theory of linear inequality with UF;
- [Henzinger et al 04] on a theory with arithmetic and pointer expressions, and call-by-value functions;
- [YorshMusuvathi 05] on a class of first-order theories;
- [Kapur et al 06] on theories of arrays, sets and multisets;
- [RybalchenkoSofronie-Stokkermans 10] to reduce the synthesis of Craig interpolants of the combined theory of linear arithmetic and uninterpreted function symbols to constraint solving.

But little work on how to synthesize non-linear interpolants

Interpolants

Given two formulae ϕ and ψ of $\mathcal{T}$ with $\vdash_{\mathcal{T}} (\phi \wedge \psi) \Rightarrow \bot$, then we say a formula Θ is an interpolant of ϕ and ψ, if $\vdash_{\mathcal{T}} \phi \Rightarrow \Theta$, $\vdash_{\mathcal{T}} (\psi \wedge \Theta) \Rightarrow \bot$, and Θ contains only symbols that ϕ and ψ share.

Semi-algebraic system

A semi-algebraic system (SAS) $\mathcal{T}(\mathbf{x})$ is of the form $\bigwedge_{j=0}^{k} f_j(\mathbf{x}) \rhd_j 0$, where f_j are polynomials in $\mathbb{R}[\mathbf{x}]$ and $\rhd_j \in \{=, \neq, \geq\}$.

Problem description

- Let $\phi_1 = \bigvee_{t=1}^{m} \mathcal{T}_{1t}(\mathbf{x}_1), \phi_2 = \bigvee_{l=1}^{n} \mathcal{T}_{2l}(\mathbf{x}_2)$, and $\phi_1 \wedge \phi_2 \models \bot$, the problem is to find a PF I in which all polynomials are in $\mathbb{R}[\mathbf{x}_1 \cap \mathbf{x}_2]$ s.t. $\phi_1 \models I$ and $I \wedge \phi_2 \models \bot$
- If for each t and l, there is an interpolant I_{tl} for SASs $\mathcal{T}_{1t}(\mathbf{x}_1)$ and $\mathcal{T}_{2l}(\mathbf{x}_2)$, then $I = \bigvee_{t=1}^{m} \bigwedge_{l=1}^{n} I_{tl}$ is an interpolant of ϕ_1 and ϕ_2.
- So, only need to consider how to construct interpolants for two SASs

Interpolants

Given two formulae ϕ and ψ of $\mathcal{T}$ with $\vdash_{\mathcal{T}} (\phi \wedge \psi) \Rightarrow \bot$, then we say a formula Θ is an interpolant of ϕ and ψ, if $\vdash_{\mathcal{T}} \phi \Rightarrow \Theta$, $\vdash_{\mathcal{T}} (\psi \wedge \Theta) \Rightarrow \bot$, and Θ contains only symbols that ϕ and ψ share.

Semi-algebraic system

A semi-algebraic system (SAS) $\mathcal{T}(\mathbf{x})$ is of the form $\bigwedge_{j=0}^{k} f_j(\mathbf{x}) \triangleright_j 0$, where f_j are polynomials in $\mathbb{R}[\mathbf{x}]$ and $\triangleright_j \in \{=, \neq, \geq\}$.

Problem description

- Let $\phi_1 = \bigvee_{t=1}^{m} \mathcal{T}_{1t}(\mathbf{x}_1), \phi_2 = \bigvee_{l=1}^{n} \mathcal{T}_{2l}(\mathbf{x}_2)$, and $\phi_1 \wedge \phi_2 \models \bot$, the problem is to find a PF I in which all polynomials are in $\mathbb{R}[\mathbf{x}_1 \cap \mathbf{x}_2]$ s.t. $\phi_1 \models I$ and $I \wedge \phi_2 \models \bot$
- If for each t and l, there is an interpolant I_{tl} for SASs $\mathcal{T}_{1t}(\mathbf{x}_1)$ and $\mathcal{T}_{2l}(\mathbf{x}_2)$, then $I = \bigvee_{t=1}^{m} \bigwedge_{l=1}^{n} I_{tl}$ is an interpolant of ϕ_1 and ϕ_2.
- So, only need to consider how to construct interpolants for two SASs

Interpolants

Given two formulae ϕ and ψ of $\mathcal{T}$ with $\vdash_{\mathcal{T}} (\phi \wedge \psi) \Rightarrow \bot$, then we say a formula Θ is an interpolant of ϕ and ψ, if $\vdash_{\mathcal{T}} \phi \Rightarrow \Theta$, $\vdash_{\mathcal{T}} (\psi \wedge \Theta) \Rightarrow \bot$, and Θ contains only symbols that ϕ and ψ share.

Semi-algebraic system

A semi-algebraic system (SAS) $\mathcal{T}(\mathbf{x})$ is of the form $\bigwedge_{j=0}^{k} f_j(\mathbf{x}) \rhd_j 0$, where f_j are polynomials in $\mathbb{R}[\mathbf{x}]$ and $\rhd_j \in \{=, \neq, \geq\}$.

Problem description

- Let $\phi_1 = \bigvee_{t=1}^{m} \mathcal{T}_{1t}(\mathbf{x}_1), \phi_2 = \bigvee_{l=1}^{n} \mathcal{T}_{2l}(\mathbf{x}_2)$, and $\phi_1 \wedge \phi_2 \models \bot$, the problem is to find a PF I in which all polynomials are in $\mathbb{R}[\mathbf{x}_1 \cap \mathbf{x}_2]$ s.t. $\phi_1 \models I$ and $I \wedge \phi_2 \models \bot$
- If for each t and l, there is an interpolant I_{tl} for SASs $\mathcal{T}_{1t}(\mathbf{x}_1)$ and $\mathcal{T}_{2l}(\mathbf{x}_2)$, then $I = \bigvee_{t=1}^{m} \bigwedge_{l=1}^{n} I_{tl}$ is an interpolant of ϕ_1 and ϕ_2.
- So, only need to consider how to construct interpolants for two SASs

Common varaiables

- A simply way by quantifier elimination (QE): applying QE to $\exists \mathbf{x}_1 - \mathbf{x}_2 . \phi_1(\mathbf{x}_1)$ and $\exists \mathbf{x}_2 - \mathbf{x}_1 . \phi_2(\mathbf{x}_2)$, and obtain two formulas on the common variables $\mathbf{x}_1 \cap \mathbf{x}_2$.
- A more efficient way by local variable elimination according to the programs to be verified.

Simplified problem

Thus, we only consider $\mathcal{T}_1 \wedge \mathcal{T}_2 \models \perp$, where

$$\mathcal{T}_1 = \begin{cases} f_1(\mathbf{x}) \geq 0, \ldots, f_{s_1}(\mathbf{x}) \geq 0, \\ g_1(\mathbf{x}) \neq 0, \ldots, g_{t_1}(\mathbf{x}) \neq 0, \\ h_1(\mathbf{x}) = 0, \ldots, h_{u_1}(\mathbf{x}) = 0 \end{cases} \qquad \mathcal{T}_2 = \begin{cases} f_{s_1+1}(\mathbf{x}) \geq 0, \ldots, f_s(\mathbf{x}) \geq 0, \\ g_{t_1+1}(\mathbf{x}) \neq 0, \ldots, g_t(\mathbf{x}) \neq 0, \\ h_{u_1+l}(\mathbf{x}) = 0, \ldots, h_u(\mathbf{x}) = 0 \end{cases}$$

Common varaiables

- A simply way by quantifier elimination (QE): applying QE to $\exists \mathbf{x}_1 - \mathbf{x}_2 . \phi_1(\mathbf{x}_1)$ and $\exists \mathbf{x}_2 - \mathbf{x}_1 . \phi_2(\mathbf{x}_2)$, and obtain two formulas on the common variables $\mathbf{x}_1 \cap \mathbf{x}_2$.
- A more efficient way by local variable elimination according to the programs to be verified.

Simplified problem

Thus, we only consider $\mathcal{T}_1 \wedge \mathcal{T}_2 \models \perp$, where

$$\mathcal{T}_1 = \begin{cases} f_1(\mathbf{x}) \geq 0, \ldots, f_{s_1}(\mathbf{x}) \geq 0, \\ g_1(\mathbf{x}) \neq 0, \ldots, g_{t_1}(\mathbf{x}) \neq 0, \\ h_1(\mathbf{x}) = 0, \ldots, h_{u_1}(\mathbf{x}) = 0 \end{cases} \qquad \mathcal{T}_2 = \begin{cases} f_{s_1+1}(\mathbf{x}) \geq 0, \ldots, f_s(\mathbf{x}) \geq 0, \\ g_{t_1+1}(\mathbf{x}) \neq 0, \ldots, g_t(\mathbf{x}) \neq 0, \\ h_{u_1+l}(\mathbf{x}) = 0, \ldots, h_u(\mathbf{x}) = 0 \end{cases}$$

Step 1: Reduction by Positivestellensatz Theorem

Basic definitions

- A polynomial ideal $\mathcal{I}$: i) $0 \in \mathcal{I}$; ii) $p_1, p_2 \in \mathcal{I}$ implies $p_1 + p_2 \in \mathcal{I}$; iii) $fg \in \mathcal{I}$ whenever $f \in \mathcal{I}$ and $g \in \mathbb{R}[\mathbf{x}]$.
- A polynomial p is called **sums of square** (**SOS**), if it can be represented as of the form $f_1^2 + \ldots + f_n^2$.
- The *multiplicative monoid* $Mult(P)$ generated by a set of polynomial P is the set of finite products of the elements of P (the empty product is 1).
- The cone $\mathcal{C}(P)$ for a finite set $P \subseteq \mathbb{R}[\mathbf{x}]$ is $\{\sum_{i=1}^{r} q_i p_i \mid q_1, \ldots, q_r$ are **SOS**, $p_1, \ldots, p_r \in Mult(P)\}$.

Positivestellensatz Theorem

$\mathcal{T}_1 \wedge \mathcal{T}_2$ has no real solutions iff there exist $f \in \mathcal{C}(\{f_1, \ldots, f_s\})$, $g \in Mult(\{g_1, \ldots, g_t\})$ and $h \in \mathcal{I}(\{h_1, \ldots, h_u\})$ s.t. $f + g^2 + h \equiv 0$.

Step 1: Reduction by Positivestellensatz Theorem

Basic definitions

- A polynomial ideal $\mathcal{I}$: i) $0 \in \mathcal{I}$; ii) $p_1, p_2 \in \mathcal{I}$ implies $p_1 + p_2 \in \mathcal{I}$; iii) $fg \in \mathcal{I}$ whenever $f \in \mathcal{I}$ and $g \in \mathbb{R}[\mathbf{x}]$.
- A polynomial p is called **sums of square** (**SOS**), if it can be represented as of the form $f_1^2 + \ldots + f_n^2$.
- The *multiplicative monoid* $Mult(P)$ generated by a set of polynomial P is the set of finite products of the elements of P (the empty product is 1).
- The cone $\mathcal{C}(P)$ for a finite set $P \subseteq \mathbb{R}[\mathbf{x}]$ is $\{\sum_{i=1}^{r} q_i p_i \mid q_1, \ldots, q_r$ are **SOS**, $p_1, \ldots, p_r \in Mult(P)\}$.

Positivestellensatz Theorem

$\mathcal{T}_1 \wedge \mathcal{T}_2$ has no real solutions iff there exist $f \in \mathcal{C}(\{f_1, \ldots, f_s\})$, $g \in Mult(\{g_1, \ldots, g_t\})$ and $h \in \mathcal{I}(\{h_1, \ldots, h_u\})$ s.t. $f + g^2 + h \equiv 0$.

Reduction

Thus, $\mathcal{T}_1 \wedge \mathcal{T}_2 \models \bot$ iff $f + g^2 + h \equiv 0$, for some

$$\begin{aligned} g &= \Pi_{i=1}^{t} g_i^{2m}, \\ h &= q_1 h_1 + \cdots + q_{u_1} h_{u_1} + \cdots + q_u h_u, \\ f &= p_0 + p_1 f_1 + \cdots + p_s f_s + p_{12} f_1 f_2 + \cdots + p_{1\ldots s} f_1 \ldots f_s. \end{aligned}$$

in which q_i and p_i are **SOS**.

Restricted solution

- If f can be represented as $p_0 + f_{\mathcal{T}_1} + f_{\mathcal{T}_2}$, where $f_{\mathcal{T}_1} = \sum_{v \subseteq 1,\ldots,s_1} p_v \Pi_{i \in v} f_i$, $f_{\mathcal{T}_2} = \sum_{v \subseteq s_1+1,\ldots,s} p_v \Pi_{i \in v} f_i$, in which $\forall \mathbf{x}. p_0(\mathbf{x}) > 0$ and $p_v \in$ **SOS**, then let
$$\begin{aligned} h_{\mathcal{T}_1} &= q_1 h_1 + \cdots + q_{u_1} h_{u_1}, \\ h_{\mathcal{T}_2} &= h - h_{\mathcal{T}_1}, \\ q &= f_{\mathcal{T}_1} + g^2 + h_{\mathcal{T}_1} + \frac{q_0}{2} = -(f_{\mathcal{T}_2} + h_{\mathcal{T}_2}) - \frac{q_0}{2}. \end{aligned}$$
- Let $I = q(\mathbf{x}) > 0$. Obviously, $\mathcal{T}_1 \models I$ and $I \wedge \mathcal{T}_2 \models \bot$.

Reduction

Thus, $\mathcal{T}_1 \wedge \mathcal{T}_2 \models \bot$ iff $f + g^2 + h \equiv 0$, for some

$$\begin{array}{rcl} g & = & \Pi_{i=1}^{t} g_i^{2m}, \\ h & = & q_1 h_1 + \cdots + q_{u_1} h_{u_1} + \cdots + q_u h_u, \\ f & = & p_0 + p_1 f_1 + \cdots + p_s f_s + p_{12} f_1 f_2 + \cdots + p_{1\ldots s} f_1 \ldots f_s. \end{array}$$

in which q_i and p_i are **SOS**.

Restricted solution

- If f can be represented as $p_0 + f_{\mathcal{T}_1} + f_{\mathcal{T}_2}$, where $f_{\mathcal{T}_1} = \sum_{v \subseteq 1,\ldots,s_1} p_v \Pi_{i \in v} f_i$, $f_{\mathcal{T}_2} = \sum_{v \subseteq s_1+1,\ldots,s} p_v \Pi_{i \in v} f_i$, in which $\forall \mathbf{x}. p_0(\mathbf{x}) > 0$ and $p_v \in$ **SOS**, then let
$h_{\mathcal{T}_1} = q_1 h_1 + \cdots + q_{u_1} h_{u_1}$,
$h_{\mathcal{T}_2} = h - h_{\mathcal{T}_1}$,
$q = f_{\mathcal{T}_1} + g^2 + h_{\mathcal{T}_1} + \frac{q_0}{2} = -(f_{\mathcal{T}_2} + h_{\mathcal{T}_2}) - \frac{q_0}{2}$.
- Let $I = q(\mathbf{x}) > 0$. Obviously, $\mathcal{T}_1 \models I$ and $I \wedge \mathcal{T}_2 \models \bot$.

A running example

```
1  if ($x*x+y*y<1$)                          $g_1 = 1 - x^2 - y^2 > 0$
2  {   /* initial values */
3     while ($x*x+y*y<3$)                    $g_2 = 3 - x^2 - y^2 > 0$
4     {   $x := x*x+y-1$;                    $f_1 = x^2 + y - 1 - x' = 0$
5         $y := y+x*y+1$;                    $f_2 = y + x'y + 1 - y' = 0$
6         if ($x*x-2*y*y-4>0$)               $g_3 = x'^2 - 2y'^2 - 4 > 0$
7            /* unsafe area */
8            error();    } }
```

- The property to be verified is that **error()** procedure will never be executed.
- Suppose there is an execution segment $1 \to 3 \to 4 \to 5 \to 6 \to 8$.
- Let $\phi \triangleq g_1 > 0 \wedge f_1 = 0 \wedge f_2 = 0$ and $\psi \triangleq g_3 > 0$.
- The execution segment is infeasible iff $\phi \wedge \psi$ is unsatisfiable.

A running example

```
1  if (x * x + y * y < 1)                         $g_1 = 1 - x^2 - y^2 > 0$
2  {    /* initial values */
3     while (x * x + y * y < 3)                   $g_2 = 3 - x^2 - y^2 > 0$
4     {    x := x * x + y - 1;                    $f_1 = x^2 + y - 1 - x' = 0$
5          y := y + x * y + 1;                    $f_2 = y + x'y + 1 - y' = 0$
6          if (x * x - 2 * y * y - 4 > 0)         $g_3 = x'^2 - 2y'^2 - 4 > 0$
7              /* unsafe area */
8              error();    } }
```

- The property to be verified is that **error()** procedure will never be executed.
- Suppose there is an execution segment $1 \to 3 \to 4 \to 5 \to 6 \to 8$.
- Let $\phi \triangleq g_1 > 0 \wedge f_1 = 0 \wedge f_2 = 0$ and $\psi \triangleq g_3 > 0$.
- The execution segment is infeasible iff $\phi \wedge \psi$ is unsatisfiable.

Cont'd

So, by the above analysis, if there exist $\delta_1, \ldots, \delta_7$ s.t.

$$g_1\delta_2 + f_1(\delta_3 - \delta_4) + f_2(\delta_5 - \delta_6) + g_3\delta_7 + \delta_1 + 1 \equiv 0,$$

where $\delta_1, \ldots, \delta_6 \in \mathbb{R}[x, y, x', y'], \delta_7 \in \mathbb{R}[x', y']$ are sums of squares (**SOS**), then $g_3\delta_7 + \frac{1}{2} < 0$ is an interpolant for ϕ and ψ.

Step 2: Construction of f, g, h by SDP

Showing the basic idea by continuing the example

- Set $\deg(\delta_1) = \deg(\delta_2) = \cdots = \deg(\delta_7) = 4$. Then $\delta_1 = (Z_2^4)^T Q_1 (Z_2^4)$, $\delta_2 = (Z_2^4)^T Q_2 (Z_2^4), \ldots, \delta_7 = (Z_2^4)^T Q_7 (Z_2^4)$, where
  - $Q_1, \ldots, Q_7$ are 15×15-symmetric matrices, and all entries of $Q_1, \ldots, Q_7$ are parameters.
  - $Z_2^4 = [1, x, y, x', y', xy, xx', xy', x'y, x'y', yy', x^2, y^2, x'^2, y'^2]$.

- $g_1\delta_2+f_1(\delta_3-\delta_4)+f_2(\delta_5-\delta_6)+g_3\delta_7+\delta_1+1=\sum_{i+j+k+m\le 4} c^{i,j,k,m}x^iy^jx'^ky'^m$,

  where $c_{i,j,k,m}=\langle A_{i,j,k,m},X\rangle$, $X=diag\{Q_1,Q_2,\ldots,Q_7\}$, entries of $A_{i,j,k,m}$ comes from coefficients of g_1,f_1,f_2,g_3, e.g.,

$$A_{0,0,0,0}=\begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & -1 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots\cdots & -4 \end{pmatrix}$$

- Resulted SDP: $\inf_{X\in Sym_{105}} \langle C,X\rangle$ s.t. $X\succeq 0, \langle A_{i,j,k,m},X\rangle=0$ $(i,j,k,m=1,\ldots,4)$.
- Solving the SDP, obtain $\delta_1,\ldots\delta_7$ with tool AiSat. Thus, $g_3\delta_7+\frac{1}{2}<0$ is an interpolant.
- In addition, we can verify that it is an inductive invariant by QE.

- $g_1\delta_2+f_1(\delta_3-\delta_4)+f_2(\delta_5-\delta_6)+g_3\delta_7+\delta_1+1 = \sum_{i+j+k+m\le 4} c^{i,j,k,m}x^iy^jx'^ky'^m$,

  where $c_{i,j,k,m} = \langle A_{i,j,k,m}, X\rangle$, $X = diag\{Q_1, Q_2, \ldots, Q_7\}$, entries of $A_{i,j,k,m}$ comes from coefficients of g_1, f_1, f_2, g_3, e.g.,

$$A_{0,0,0,0} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & -1 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots\cdots & -4 \end{pmatrix}$$

- Resulted SDP: $\inf_{X\in Sym_{\mathbf{105}}} \langle C, X\rangle$ s.t. $X \succeq 0, \langle A_{i,j,k,m}, X\rangle = 0$ $(i,j,k,m = 1,\ldots,4)$.
- Solving the SDP, obtain $\delta_1,\ldots\delta_7$ with tool **AiSat**. Thus, $g_3\delta_7 + \frac{1}{2} < 0$ is an interpolant.
- In addition, we can verify that it is an inductive invariant by QE.

- $g_1\delta_2+f_1(\delta_3-\delta_4)+f_2(\delta_5-\delta_6)+g_3\delta_7+\delta_1+1=\sum_{i+j+k+m\leq 4} c^{i,j,k,m}x^iy^jx'^ky'^m$,

  where $c_{i,j,k,m}=\langle A_{i,j,k,m},X\rangle$, $X=diag\{Q_1,Q_2,\dots,Q_7\}$, entries of $A_{i,j,k,m}$ comes from coefficients of g_1,f_1,f_2,g_3, e.g.,

$$A_{0,0,0,0}=\begin{pmatrix}1 & \cdots & 0 & \cdots & 0 & \cdots\cdots & 0\\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots\\ 0 & \cdots & 1 & \cdots & 0 & \cdots\cdots & 0\\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & \cdots & -1 & \cdots\cdots & 0\\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & \cdots & 0 & \cdots\cdots & -4\end{pmatrix}$$

- Resulted SDP: $\inf_{X\in Sym_{\mathbf{105}}}\langle C,X\rangle$ s.t. $X\succeq 0, \langle A_{i,j,k,m},X\rangle=0$ $(i,j,k,m=1,\dots,4)$.
- Solving the SDP, obtain $\delta_1,\dots\delta_7$ with tool **AiSat**. Thus, $g_3\delta_7+\frac{1}{2}<0$ is an interpolant.
- In addition, we can verify that it is an inductive invariant by QE.

- $g_1\delta_2+f_1(\delta_3-\delta_4)+f_2(\delta_5-\delta_6)+g_3\delta_7+\delta_1+1=\sum_{i+j+k+m\leq 4} c^{i,j,k,m}x^i y^j x'^k y'^m$,

  where $c_{i,j,k,m}=\langle A_{i,j,k,m}, X\rangle$, $X=diag\{Q_1,Q_2,\ldots,Q_7\}$, entries of $A_{i,j,k,m}$ comes from coefficients of g_1,f_1,f_2,g_3, e.g.,

$$A_{0,0,0,0}=\begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & \cdots & 0 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & -1 & \cdots\cdots & 0 \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots\cdots & -4 \end{pmatrix}$$

- Resulted SDP: $\inf_{X\in Sym_{\mathbf{105}}} \langle C, X\rangle$ s.t. $X\succeq 0, \langle A_{i,j,k,m}, X\rangle = 0$ $(i,j,k,m=1,\ldots,4)$.
- Solving the SDP, obtain $\delta_1,\ldots\delta_7$ with tool **AiSat**. Thus, $g_3\delta_7+\frac{1}{2}<0$ is an interpolant.
- In addition, we can verify that it is an inductive invariant by QE.

Discussions

- The approach is **sound**, but not **complete** in general.
- **Under which condition will the approach become complete?**

Quadratic module

The **quadratic module** generated by $g_1, \ldots, g_m$ is the set $\mathcal{M}(g_1, \ldots, g_m) = \{\delta_0 + \sum_{j=1}^{m} \delta_j g_j \mid \delta_0, \delta_j$ are **SOS**$\}$.

Archimedean condition

- A quadratic module $\mathcal{M}(g_1, \ldots, g_m)$ is said to be $\mathrm{Archimedean}$ if $\forall p \in \mathbb{R}[\mathbf{x}], \exists n \in \mathbb{N}. n \pm p \in \mathcal{M}(f_1, \ldots, f_s)$.
- Let $\mathcal{T}_1' = f_1(\mathbf{x}) \geq 0, \ldots, f_{s_1}(\mathbf{x}) \geq 0$ and $\mathcal{T}_2' = f_{s_1+1}(\mathbf{x}) \geq 0, \ldots, f_s(\mathbf{x}) \geq 0$ be two SASs, which contains constraints $c_l \leq x_i \leq c_r$ for every $x_i \in \mathbf{x}$, where c_l and c_r are reals. We can always obtain a system $\{f_1(\mathbf{x}), \ldots, f_{s'}(\mathbf{x})\}$ s.t. $\mathcal{M}(f_1, \ldots, f_{s'})$ is *Archimedean* and $f_1 \geq 0 \wedge \cdots \wedge f_s \geq 0 \Leftrightarrow f_1 \geq 0 \wedge \cdots \wedge f_{s'} \geq 0$.

Theorem

If $\mathcal{T}_1' \wedge \mathcal{T}_2'$ is unsatisfiable, then $-1 \in \mathcal{M}(f_1, \ldots, f_{s'})$.

Quadratic module

The **quadratic module** generated by $g_1, \ldots, g_m$ is the set $\mathcal{M}(g_1, \ldots, g_m) = \{\delta_0 + \sum_{j=1}^{m} \delta_j g_j \mid \delta_0, \delta_j$ are **SOS**$\}$.

Archimedean condition

- A quadratic module $\mathcal{M}(g_1, \ldots, g_m)$ is said to be Archimedean if $\forall p \in \mathbb{R}[\mathbf{x}], \exists n \in \mathbb{N}. n \pm p \in \mathcal{M}(f_1, \ldots, f_s)$.
- Let $\mathcal{T}_1' = f_1(\mathbf{x}) \geq 0, \ldots, f_{s_1}(\mathbf{x}) \geq 0$ and $\mathcal{T}_2' = f_{s_1+1}(\mathbf{x}) \geq 0, \ldots, f_s(\mathbf{x}) \geq 0$ be two SASs, which contains constraints $c_l \leq x_i \leq c_r$ for every $x_i \in \mathbf{x}$, where c_l and c_r are reals. We can always obtain a system $\{f_1(\mathbf{x}), \ldots, f_{s'}(\mathbf{x})\}$ s.t. $\mathcal{M}(f_1, \ldots, f_{s'})$ is *Archimedean* and $f_1 \geq 0 \wedge \cdots \wedge f_s \geq 0 \Leftrightarrow f_1 \geq 0 \wedge \cdots \wedge f_{s'} \geq 0$.

Theorem

If $\mathcal{T}_1' \wedge \mathcal{T}_2'$ is unsatisfiable, then $-1 \in \mathcal{M}(f_1, \ldots, f_{s'})$.

Quadratic module

The **quadratic module** generated by $g_1, \ldots, g_m$ is the set $\mathcal{M}(g_1, \ldots, g_m) = \{\delta_0 + \sum_{j=1}^{m} \delta_j g_j \mid \delta_0, \delta_j$ are **SOS**$\}$.

Archimedean condition

- A quadratic module $\mathcal{M}(g_1, \ldots, g_m)$ is said to be $\mathrm{Archimedean}$ if $\forall p \in \mathbb{R}[\mathbf{x}], \exists n \in \mathbb{N}. n \pm p \in \mathcal{M}(f_1, \ldots, f_s)$.
- Let $\mathcal{T}_1' = f_1(\mathbf{x}) \geq 0, \ldots, f_{s_1}(\mathbf{x}) \geq 0$ and $\mathcal{T}_2' = f_{s_1+1}(\mathbf{x}) \geq 0, \ldots, f_s(\mathbf{x}) \geq 0$ be two SASs, which contains constraints $c_l \leq x_i \leq c_r$ for every $x_i \in \mathbf{x}$, where c_l and c_r are reals. We can always obtain a system $\{f_1(\mathbf{x}), \ldots, f_{s'}(\mathbf{x})\}$ s.t. $\mathcal{M}(f_1, \ldots, f_{s'})$ is *Archimedean* and $f_1 \geq 0 \wedge \cdots \wedge f_s \geq 0 \Leftrightarrow f_1 \geq 0 \wedge \cdots \wedge f_{s'} \geq 0$.

Theorem

If $\mathcal{T}_1' \wedge \mathcal{T}_2'$ is unsatisfiable, then $-1 \in \mathcal{M}(f_1, \ldots, f_{s'})$.

Revised algorithm

- There exist $\sigma_0, \ldots, \sigma_{s'}$ which are **SOS** s.t. $-1 = \sigma_0 + \sigma_1 f_1 + \cdots + \sigma_{s_1} f_{s_1} + \sigma_{s_1+1} f_{s_1+1} + \cdots + f_{s'} \sigma_{s'}$.
- Then, $-(\frac{1}{2} + \sigma_{s_1+1} f_{s_1+1} + \cdots + \sigma_{s'} f_{s'}) = \frac{1}{2} + \sigma_0 + \sigma_1 f_1 + \cdots + \sigma_{s_1} f_{s_1}$.
- Let $q(\mathbf{x}) = \frac{1}{2} + \sigma_0 + \sigma_1 f_1 + \cdots + \sigma_{s_1} f_{s_1}$, we have $\forall \mathbf{x} \in \mathcal{T}_1 . q(\mathbf{x}) > 0$ and $\forall \mathbf{x} \in \mathcal{T}_2 . f_{s'}(\mathbf{x}) \geq 0,$ so $q(\mathbf{x}) < 0$. Thus, $I = q(\mathbf{x}) > 0$ is an interpolant of $\mathcal{T}_1$ and $\mathcal{T}_2$.

Discussions

Reasonability: Only bounded numbers with finite precision can be represented in computer;

Necessity: Let $\mathcal{T}_1 = \{x_1 \geq 0, x_2 \geq 0\}$ and $\mathcal{T}_2 = \{-x_1 x_2 - 1 \geq 0\}$. So, $\mathcal{T}_1 \wedge \mathcal{T}_2 = \emptyset$ is not *Archimedean* and unsatisfiable, but $-1 \notin \mathcal{M}(x_1, x_2, -x_1 x_2 - 1)$.

Complexity

- The total cost of the revised algorithm is polynomial in $b_f u \binom{n+b_f/2}{n} \binom{n+b_f}{n}$.
- For a given problem in which n, u are fixed, the complexity of the algorithm becomes polynomial in b_f.
- The upper bound on b_f is at least triply exponential in u and n.

Part II: Applications to Program Verification

Invariant

Inductive invariant of loop

Given a Hoare triple $\{Pre\}$ **while** B **do** P $\{Post\}$, we say a formula θ is an invariant of the loop, if

- $\theta \Rightarrow Pre$
- $\{\theta \wedge B\}\ P\ \{\theta\}$
- $\theta \wedge \neg B \Rightarrow Post$

General framework

- To negate $Post$;
- For each single execution of P, generating an interpolant between Pre and $\neg Post$;
- Using QE to verify the disjunct of the obtained interpolants form an inductive invariant of the loop.

Invariant

Inductive invariant of loop

Given a Hoare triple $\{Pre\}$ **while** B **do** P $\{Post\}$, we say a formula θ is an invariant of the loop, if

- $\theta \Rightarrow Pre$
- $\{\theta \wedge B\}\ P\ \{\theta\}$
- $\theta \wedge \neg B \Rightarrow Post$

General framework

- To negate $Post$;
- For each single execution of P, generating an interpolant between Pre and $\neg Post$;
- Using QE to verify the disjunct of the obtained interpolants form an inductive invariant of the loop.

Invariant

Inductive invariant of loop

Given a Hoare triple $\{Pre\}$ **while** B **do** P $\{Post\}$, we say a formula θ is an invariant of the loop, if

- $\theta \Rightarrow Pre$
- $\{\theta \wedge B\}\ P\ \{\theta\}$
- $\theta \wedge \neg B \Rightarrow Post$

General framework

- To negate $Post$;
- For each single execution of P, generating an interpolant between Pre and $\neg Post$;
- Using QE to verify the disjunct of the obtained interpolants form an inductive invariant of the loop.

Example

Source code

```
1     vc := 0;
2        /* the initial veclocity  */
3     fr := 1000;
4        /* the initial force */
5     ac := 0.0005 * fr;
6         /* the initial acceleration */
7     while ( 1 )
8     {    fa := 0.5418 * vc * vc;
9              /* the force control  */
10       fr := 1000 − fa;
11       ac := 0.0005 * fr;
12       vc := vc + ac;
13         assert(vc < 49.61);
14           /* the safety velocity  */ }
```

- Safety property is that the velocity of the car cannot surpass $49.61m/s$.
- Suppose $(vc < 49.61) \to 8 \to 10 \to 11 \to 12 \to 13$ $(vc \geq 49.61)$.
- By applying **AiSat**, we can obtain an interpolant $-1.3983vc + 69.358 > 0$, which guarantees $vc < 49.61$.
- It is easy to verify $-1.3983vc + 69.358 > 0$ is an inductive invariant.

Comparison with other approaches

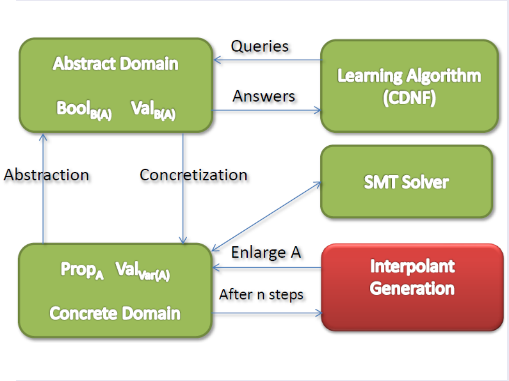
Interpolation based vs QE based

- Interpolation based is complete under *Archimedean condition*, while QE based is complete related to the predefined templates.
- Interpolation based is more efficient, its complexity is polynomial in the given degree, whose upper bound is at least triply exponential in the numbers of variables and constraints; while QE based is doubly exponential in the number of parameters and variables in the predefined templates in general.
- Interpolation based has to consider error issue because of numerical computation, while QE based does not need.

Combining interpolation generation with QE to invariant generation can improve efficiency, also makes error is controllable.

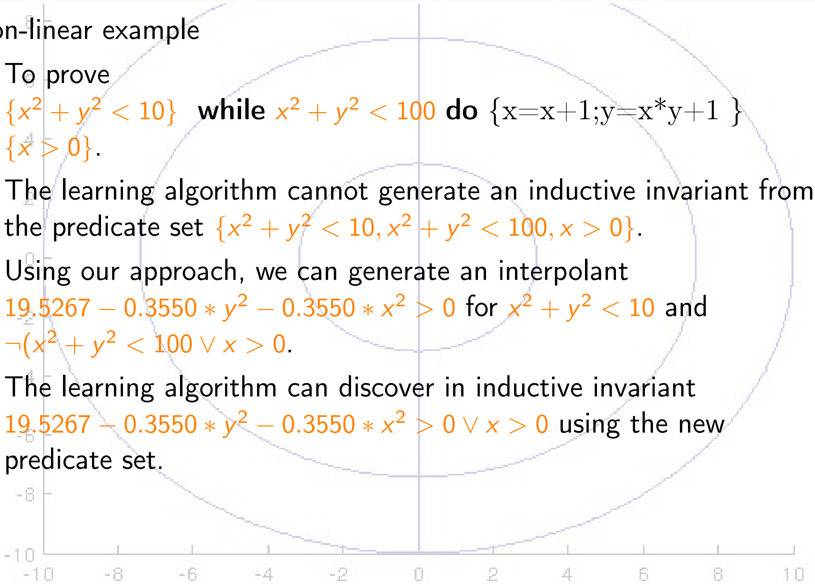
Combining with machine-learning

General framework



- Two parts: Learning + discovering predicate set.
- Learning=CDNF+predicate abstraction+SMT solver.
- Discovering predicate set is by interpolation.
- Learning part is incomplete: after *n* steps, if no invariant is synthesized, either **restart** or **enlarge** the predicate set by interpolation.
- Previous work is only applicable to linear cases.

Extending to non-linear cases



A non-linear example

- To prove $\{x^2 + y^2 < 10\}$ **while** $x^2 + y^2 < 100$ **do** {x=x+1;y=x*y+1 } $\{x > 0\}$.
- The learning algorithm cannot generate an inductive invariant from the predicate set $\{x^2 + y^2 < 10, x^2 + y^2 < 100, x > 0\}$.
- Using our approach, we can generate an interpolant $19.5267 - 0.3550 * y^2 - 0.3550 * x^2 > 0$ for $x^2 + y^2 < 10$ and $\neg(x^2 + y^2 < 100 \vee x > 0$.
- The learning algorithm can discover in inductive invariant $19.5267 - 0.3550 * y^2 - 0.3550 * x^2 > 0 \vee x > 0$ using the new predicate set.

Conclusion

- We first summarize our recent work on generating non-linear interpolants by semi-definite programming.
- Then we show how to apply the results to program verification, including invariant generation and combining with machine-learning based techniques.

Future work

- Some issues related to nonlinear interpolant generation, like
  - How to relax the Archimedean condition.
  - How to combine non-linear arithmetic with other well-established decidable first order theories.
  - To investigate errors caused by numerical computation in **SDP** is quite interesting.
- To investigate combining with other verification techniques, like CEGAR, BMC, SMT and so on.
- To investigate the possibility to the verification of hybrid systems.

Conclusion

- We first summarize our recent work on generating non-linear interpolants by semi-definite programming.
- Then we show how to apply the results to program verification, including invariant generation and combining with machine-learning based techniques.

Future work

- Some issues related to nonlinear interpolant generation, like
  - How to relax the Archimedean condition.
  - How to combine non-linear arithmetic with other well-established decidable first order theories.
  - To investigate errors caused by numerical computation in **SDP** is quite interesting.
- To investigate combining with other verification techniques, like CEGAR, BMC, SMT and so on.
- To investigate the possibility to the verification of hybrid systems.

Conclusion

- We first summarize our recent work on generating non-linear interpolants by semi-definite programming.
- Then we show how to apply the results to program verification, including invariant generation and combining with machine-learning based techniques.

Future work

- Some issues related to nonlinear interpolant generation, like
  - How to relax the Archimedean condition.
  - How to combine non-linear arithmetic with other well-established decidable first order theories.
  - To investigate errors caused by numerical computation in **SDP** is quite interesting.
- To investigate combining with other verification techniques, like CEGAR, BMC, SMT and so on.
- To investigate the possibility to the verification of hybrid systems.

References

1. L. Dai, B. Xia and N. Zhan: Generating non-linear interpolants by semi-definite programming. CAV 2013.
2. Y. Chen, B. Xia, L. Yang and N. Zhan: Generating polynomial invariants by DISCOVERER and QEPCAD. Formal Methods and Hybrid Real-time Systems.
3. Yungbum Jung, Wonchan Lee, Bow-Yaw Wang, Kwangkeun Yi: Predicate generation for learning-based quantifier-free loop invariant inference. TACAS 2011.
4. Soonho Kong, Yungbum Jung, Cristina David, Bow-Yaw Wang, Kwangkeun Yi: Automatically inferring quantified Loop Invariants by Algorithmic Learning from Simple Templates. APLAS 2010.

Thanks & Questions?