Interpolation over Nonlinear Arithmetic
Towards Program Reasoning and Verification

Mingshuai Chen

chenms@cs.rwth-aachen.de  moves.rwth-aachen.de/people/chenms


MOVES · November 2019
What Is Interpolation?

Interpolation /ɪntəˈpəːləʃ(ə)n/

MATHEMATICS

“the insertion of an intermediate value or term into a series by estimating or calculating it from surrounding known values.”

[OXFORD Dictionary]
What Is Interpolation?

**Interpolation /ɪntəˈpələʃ(ə)n/**

MATHEMATICS

“the insertion of an intermediate value or term into a series by estimating or calculating it from surrounding known values.”

[OXFORD Dictionary]
What Is Interpolation?

**Interpolation** /ˌɪntəˈpəˌleɪʃ(ə)n/

**MATHEMATICS**

“the insertion of an intermediate value or term into a series by estimating or calculating it from surrounding known values.”

[OXFORD Dictionary]

**LOGICAL REASONING**

\[ P \models Q \quad P \models R \models Q \]
What Is Interpolation?

**Interpolation** /ɪntəˈpələʃ(ə)n/

**MATHEMATICS**

“the insertion of an intermediate value or term into a series by estimating or calculating it from surrounding known values.”

[OXFORD Dictionary]

**LOGICAL REASONING**

\[ P \models Q \quad P \models R \models Q \]

\[ P \land Q \models \bot \quad P \models R \text{ and } R \land Q \models \bot \]
Interpolants as Loop Invariants

Example ([Sharma et al., CAV’12])

\[
\begin{align*}
&x := 0; \, y := 0; \\
&\text{while } (*) \\
&\quad \{ x := x + 1; \, y := y + 1; \} \\
&\text{while } (x \neq 0) \\
&\quad \{ x := x - 1; \, y := y - 1; \} \\
&\text{if } (y \neq 0) \\
&\quad \text{error ()};
\end{align*}
\]
Bounded model checking and interpolation can be combined to produce an over-approximate image operator that can be used in symbolic model checking. The intuition behind this is as follows. A bounded model checking problem consists of a set of constraints – initial condition and first instance of the transition constraint are in set $I$ and $P$ is implied by the initial condition and the first transition constraint, and these constraints are translated to conjunctive normal form, and, as appropriate, instantiated for each time frame $0$ to $k$. Since $I$ and $P$ are unsatisfiable, meaning that no state satisfying $T$ and $B$ are exactly $s$. Further, this over-approximate image operation can be iterated to compute an over-approximate image of the forward image of the initial state.

In the figure, $A$ and $B$ are exactly $s$. The common variables of $A$ and $B$ are depicted in Figure 2. The common variables of $A$ and $B$ are exactly $s$. As depicted in Figure 1.

Interpolants as Loop Invariants

Example ([Sharma et al., CAV’12])

```plaintext
x := 0; y := 0;
while (*)
    {x := x + 1; y := y + 1; }
while (x ≠ 0)
    {x := x - 1; y := y - 1; }
if (y ≠ 0)
    error ()
```
Interpolants as Loop Invariants

Example ([Sharma et al., CAV’12])

\[
\begin{aligned}
  x &:= 0; y := 0; \\
  \text{while} (\star) & \{ x := x + 1; y := y + 1; \} \\
  \text{while} (x \neq 0) & \{ x := x - 1; y := y - 1; \} \\
  \text{if} (y \neq 0) & \text{error ();}
\end{aligned}
\]

\[
A \equiv x_1 = 0 \land y_1 = 0
\]

\[
\begin{aligned}
  \text{ite} (b, & x = x_1 \land y = y_1, \\
  & x = x_1 + 1 \land y = y_1 + 1)
\end{aligned}
\]
Interpolants as Loop Invariants

Example ([Sharma et al., CAV’12])

\[ x := 0; \ y := 0; \]
while (*)
\[
\{ x := x + 1; \ y := y + 1; \}
\]

while (\( x \neq 0 \))
\[
\{ x := x - 1; \ y := y - 1; \}
\]
if (\( y \neq 0 \))
\[
\text{error ()};
\]

\( A \cong x_1 = 0 \land y_1 = 0\)

\[ \text{ite} (b, \]
\[
x = x_1 \land y = y_1,
\]
\[
x = x_1 + 1 \land y = y_1 + 1
\]

\( B \cong \text{ite} (x = 0, \]
\[
x_2 = x \land y_2 = y,
\]
\[
x_2 = x - 1 \land y_2 = y - 1) \land
\]
\[
x_2 = 0 \land \neg(y_2 = 0)
Interpolants as Loop Invariants

Example ([Sharma et al., CAV’12])

\[
x := 0; \ y := 0;
\]

while (\(*\))
\[
\{x := x + 1; \ y := y + 1; \}
\]
while (\(x \neq 0\))
\[
\{x := x - 1; \ y := y - 1; \}
\]
if (\(y \neq 0\))
\[
\text{error ()};
\]

\[
A \triangleq x_1 = 0 \land y_1 = 0 \land \ 
\]
\[
ite (b, \ 
\]
\[
x = x_1 \land y = y_1, \quad x = x_1 + 1 \land y = y_1 + 1
\]
\[
B \triangleq \ite (x = 0, \ 
\]
\[
x_2 = x \land y_2 = y, \quad x_2 = x - 1 \land y_2 = y - 1) \land \ 
\]
\[
x_2 = 0 \land \lnot(y_2 = 0)
\]

\[
A \land B \models \bot. \quad l(x, y) \triangleq x = y \text{ s.t. } A \models l \text{ and } l \land B \models \bot.
\]
Interpolants as Loop Invariants

Example ([Sharma et al., CAV ’12])

\[ x := 0; y := 0; \]
while (x ≠ 0)
\[ x := x - 1; y := y - 1; \}
if (y ≠ 0)
\[ \text{error}(); \]

\[ A \equiv x_1 = 0 \land y_1 = 0 \land \text{ite} (b, \]
\[ x = x_1 \land y = y_1, \]
\[ x = x_1 + 1 \land y = y_1 + 1) \]
\[ B \equiv \text{ite} (x = 0, \]
\[ x_2 = x \land y_2 = y, \]
\[ x_2 = x - 1 \land y_2 = y - 1) \land \]
\[ x_2 = 0 \land \neg (y_2 = 0) \]

\[ A \land B \models \bot. \]
\[ l(x, y) \equiv x = y \quad \text{s.t.} \quad A \models l \text{ and } l \land B \models \bot. \]

\[ \begin{array}{cccccccc}
  & I & T & T & T & T & T & T & T & F \\
  & s_0 & s_1 & \ldots & s_k & & & & & \\
\end{array} \]

\[ \begin{array}{cccccccc}
  & A & B \\
  & I & T & T & T & T & T & T & T & F \\
  & s_0 & \Rightarrow p & s_k & & & & & & & \\
\end{array} \]

\[ \text{Figure} – \text{Bounded model checking.} \]
\[ \text{Figure} – \text{Computing image by interpolation.} \]
The bottleneck of existing formal verification techniques lies in scalability.
The bottleneck of existing formal verification techniques lies in *scalability*.

Interpolation helps in scaling these verification techniques due to its inherent capability of *local and modular reasoning*:

- **Nelson-Oppen method**: equivalently decomposing a formula of a composite theory into formulas of its component theories;
- **SMT**: combining different decision procedures to verify programs with complicated data structures;
- **Bounded model-checking**: generating invariants to verify infinite-state systems due to McMillan;
- ...
Interpolant synthesis plays the central role in interpolation-based techniques:
Interpolation-based Verification

**Interpolant synthesis plays the central role in interpolation-based techniques:**

😊 Well-established methods to synthesize interpolants for various theories, e.g., decidable fragments of FOL, LA, multi-sets, etc., and combinations thereof.
Interpolation-based Verification

Interpolant synthesis plays the central role in interpolation-based techniques:

نظرًاً بسبب تأسيس طريقة للإبتداء في شبكة للطرق المختلفة، فإنه يعود إلى دقة التدفق الدقيق، إذ أن نتائج التسبيط يمكن أن تكون متعلقة في التطبيقات، والثابتات، وغيرها من الطرق، وبدافع الظروف الرائحة.

SAT-based: generate interpolants for LA from (resolution) unsatisfiability proofs.

Interpolation-based Verification

Interpolant synthesis plays the central role in interpolation-based techniques:

- Well-established methods to synthesize interpolants for various theories, e.g., decidable fragments of FOL, LA, multi-sets, etc., and combinations thereof.

  - SAT-based: generate interpolants for LA from (resolution) unsatisfiability proofs.
    

  - Constraint solving-based: reduce interpolation for LA to linear programming by Motzkin's transposition theorem.
    
Interpolation-based Verification

Interpolant synthesis plays the central role in interpolation-based techniques:

😊 Well-established methods to synthesize interpolants for various theories, e.g., decidable fragments of FOL, LA, multi-sets, etc., and combinations thereof.

- SAT-based: generate interpolants for LA from (resolution) unsatisfiability proofs.
- Constraint solving-based: reduce interpolation for LA to linear programming by Motzkin’s transposition theorem.

😊 Little work on synthesizing nonlinear ones: [Kupferschmid & Becker, FORMATS ’11], [Dai et al., CAV ’13], [Gao & Zufferey, TACAS ’16], [Okudono et al., APLAS ’17].
Interpolation-based Verification

**Interpolant synthesis plays the central role in interpolation-based techniques:**

😊 Well-established methods to synthesize interpolants for various theories, e.g., decidable fragments of FOL, LA, multi-sets, etc., and combinations thereof.

- SAT-based: generate interpolants for LA from (resolution) unsatisfiability proofs.
- Constraint solving-based: reduce interpolation for LA to linear programming by Motzkin’s transposition theorem.

😊 Little work on synthesizing nonlinear ones: [Kupferschmid & Becker, FORMATS ’11], [Dai et al., CAV ’13], [Gao & Zufferey, TACAS ’16], [Okudono et al., APLAS ’17].

- Reduce interpolation for concave quadratic polynomial inequalities to semi-definite programming. Tool: NLFIntp.
Interpolation-based Verification

Interpolant synthesis plays the central role in interpolation-based techniques:

😊 Well-established methods to synthesize interpolants for various theories, e.g., decidable fragments of FOL, LA, multi-sets, etc., and combinations thereof.

- SAT-based: generate interpolants for LA from (resolution) unsatisfiability proofs.
- Constraint solving-based: reduce interpolation for LA to linear programming by Motzkin’s transposition theorem.

😊 Little work on synthesizing nonlinear ones: [Kupferschmid & Becker, FORMATS ’11], [Dai et al., CAV ’13], [Gao & Zufferey, TACAS ’16], [Okudono et al., APLAS ’17].

- Reduce interpolation for concave quadratic polynomial inequalities to semi-definite programming. Tool: NLFIntp.
- Counterexample-guided learning of polynomial interpolants for the general quantifier-free theory of NLA. Tool: NIL.
Interpolation-based Verification

Interpolant synthesis plays the central role in interpolation-based techniques:

😊 Well-established methods to synthesize interpolants for various theories, e.g.,
decidable fragments of FOL, LA, multi-sets, etc., and combinations thereof.

- **SAT-based**: generate interpolants for LA from (resolution) unsatisfiability proofs.

- **Constraint solving-based**: reduce interpolation for LA to linear programming by Motzkin’s transposition theorem.

😊 Little work on synthesizing nonlinear ones: [Kupferschmid & Becker, FORMATS ’11], [Dai et al., CAV ’13], [Gao & Zufferey, TACAS ’16], [Okudono et al., APLAS ’17].

- **Reduce interpolation for concave quadratic polynomial inequalities to semi-definite programming. Tool**: NLFIntp.

- **Counterexample-guided learning of polynomial interpolants for the general quantifier-free theory of NLA. Tool**: NIL.
Outline

1. Interpolation vs. Classification
2. Learning Nonlinear Interpolants
3. Implementation and Evaluation
4. Concluding Remarks
Outline

1. Interpolation vs. Classification
   - Craig Interpolation
   - Binary Classification
   - Interpolants as Classifiers

2. Learning Nonlinear Interpolants
   - SVMs with Nonlinear Space Transformation
   - The NIL Algorithm and its Variants

3. Implementation and Evaluation
   - Performance over Benchmarks
   - Perturbations in Parameters

4. Concluding Remarks
   - Summary
Craig Interpolation

Craig Interpolant

Given $\phi$ and $\psi$ in a theory $\mathcal{T}$ s.t. $\phi \land \psi \models_{\mathcal{T}} \bot$, a formula $I$ is a (reverse) interpolant of $\phi$ and $\psi$ if (1) $\phi \models_{\mathcal{T}} I$; (2) $I \land \psi \models_{\mathcal{T}} \bot$; and (3) $\text{var}(I) \subseteq \text{var}(\phi) \cap \text{var}(\psi)$. 
Craig Interpolation

Craig Interpolant

Given $\phi$ and $\psi$ in a theory $\mathcal{T}$ s.t. $\phi \land \psi \models_{\mathcal{T}} \perp$, a formula $I$ is a (reverse) interpolant of $\phi$ and $\psi$ if (1) $\phi \models_{\mathcal{T}} I$; (2) $I \land \psi \models_{\mathcal{T}} \perp$; and (3) $\text{var}(I) \subseteq \text{var}(\phi) \cap \text{var}(\psi)$.

Example (over nonlinear $\mathcal{T}$)

\[
A \equiv -x_1^2 + 4x_1 + x_2 - 4 \geq 0 \land -x_1 - x_2 + 3 - y^2 > 0
\]
\[
B \equiv -3x_1^2 - x_2^2 + 1 \geq 0 \land x_2 - z^2 \geq 0
\]
\[
I \equiv -3 + 2x_1 + x_1^2 + \frac{1}{2}x_2^2 > 0
\]
Given a training dataset $X = X^+ \cup X^-$ of positive/negative sample points, find a classifier $C: X \mapsto \{\top, \bot\}$, s.t. (1) $\forall \bar{x} \in X^+. C(\bar{x}) = \top$; and (2) $\forall \bar{x} \in X^- . C(\bar{x}) = \bot$. There could be (infinitely) many valid classifiers. Support Vector Machine (SVM) finds a separating hyperplane that yields the largest distance (functional margin) to the nearest positive and negative samples (support vectors), which boils down to convex optimizations.
Binary Classification

Given a training dataset $X = X^+ \cup X^-$ of positive/negative sample points, find a classifier $C: X \mapsto \{\top, \bot\}$, s.t. (1) $\forall \vec{x} \in X^+, \ C(\vec{x}) = \top$; and (2) $\forall \vec{x} \in X^-, \ C(\vec{x}) = \bot$. 

$X^+$

- •
- •
- •
Given a training dataset $X = X^+ \cup X^-$ of positive/negative sample points, find a classifier $C: X \mapsto \{\top, \bot\}$, s.t. (1) $\forall \vec{x} \in X^+. \ C(\vec{x}) = \top$; and (2) $\forall \vec{x} \in X^- . \ C(\vec{x}) = \bot$. There could be (infinitely) many valid classifiers. Support Vector Machine (SVM) finds a separating hyperplane that yields the largest distance (functional margin) to the nearest positive and negative samples (support vectors), which boils down to convex optimizations.
Binary Classification

Given a training dataset $X = X^+ \cup X^-$ of positive/negative sample points, find a classifier $C: X \mapsto \{\top, \bot\}$, s.t. (1) $\forall \vec{x} \in X^+. \ C(\vec{x}) = \top$; and (2) $\forall \vec{x} \in X^-\. \ C(\vec{x}) = \bot$. 

There could be (infinitely) many valid classifiers. Support Vector Machine (SVM) finds a separating hyperplane that yields the largest distance (functional margin) to the nearest positive and negative samples (support vectors), which boils down to convex optimizations.
Given a training dataset \( X = X^+ \cup X^- \) of positive/negative sample points, find a classifier \( C: X \mapsto \{ \top, \bot \} \), s.t. (1) \( \forall \tilde{x} \in X^+. \ C(\tilde{x}) = \top \); and (2) \( \forall \tilde{x} \in X^- . \ C(\tilde{x}) = \bot . \)

There could be (infinitely) many valid classifiers.
**Binary Classification**

Given a training dataset $X = X^+ \cup X^-$ of positive/negative sample points, find a classifier $C: X \mapsto \{T, \bot\}$, s.t. (1) $\forall \vec{x} \in X^+. \ C(\vec{x}) = T$; and (2) $\forall \vec{x} \in X^-.. \ C(\vec{x}) = \bot$.

**Support Vector Machine (SVM)** finds a separating hyperplane that yields the largest distance (functional margin) to the nearest positive and negative samples (support vectors), which boils down to convex optimizations.
Linear interpolants can be viewed as hyperplane classifiers, [Sharma et al., CAV ’12]: sampling from $\llbracket \phi \rrbracket$ and $\llbracket \psi \rrbracket \rightarrow$ building a hyperplane classifier $\rightarrow$ refining by CEs.
Linear interpolants can be viewed as hyperplane classifiers, [Sharma et al., CAV ’12]: sampling from $[\phi]$ and $[\psi]$ $\rightarrow$ building a hyperplane classifier $\rightarrow$ refining by CEs.

$X^+$ and $X^-$ might not be linearly separable (often the case when sampled from nonlinear $\phi$ and $\psi$, resp.):

\[
A \equiv (x < 2.5 \Rightarrow y \geq 2 \sin(x)) \\
\land (x \geq 2.5 \land x < 5 \Rightarrow y \geq 0.125x^2 + 0.41) \\
\land (x \geq 5 \land x \leq 6 \Rightarrow y \geq 6.04 - 0.5x)
\]

\[
B \equiv (x < 3 \Rightarrow y \leq x \cos(0.1e^x) - 0.083) \\
\land (x \geq 3 \land x \leq 6 \Rightarrow y \leq -x^2 + 10x - 22.35)
\]
Interpolation vs. Classification

Linear interpolants can be viewed as hyperplane classifiers, [Sharma et al., CAV ’12]: sampling from $[\phi]$ and $[\psi] \rightarrow$ building a hyperplane classifier $\rightarrow$ refining by CEs.

$X^+$ and $X^-$ might not be linearly separable (often the case when sampled from nonlinear $\phi$ and $\psi$, resp.):

$$A \equiv (x < 2.5 \Rightarrow y \geq 2 \sin(x))$$
$$\land (x \geq 2.5 \land x < 5 \Rightarrow y \geq 0.125x^2 + 0.41)$$
$$\land (x \geq 5 \land x \leq 6 \Rightarrow y \geq 6.04 - 0.5x)$$

$$B \equiv (x < 3 \Rightarrow y \leq x \cos(0.1e^x) - 0.083)$$
$$\land (x \geq 3 \land x \leq 6 \Rightarrow y \leq -x^2 + 10x - 22.35)$$

Encoding interpolants as logical combinations of linear constraints.
Interpolation vs. Classification

😊 Linear interpolants can be viewed as hyperplane classifiers, [Sharma et al., CAV ’12]:
sampling from $[\phi]$ and $[\psi]$ $\rightarrow$ building a hyperplane classifier $\rightarrow$ refining by CEs.

😊 $X^+$ and $X^-$ might not be linearly separable (often the case when sampled from nonlinear $\phi$ and $\psi$, resp.):

\[ A \equiv (x < 2.5 \Rightarrow y \geq 2 \sin(x)) \]
\[ \land (x \geq 2.5 \land x < 5 \Rightarrow y \geq 0.125x^2 + 0.41) \]
\[ \land (x \geq 5 \land x \leq 6 \Rightarrow y \geq 6.04 - 0.5x) \]

\[ B \equiv (x < 3 \Rightarrow y \leq x \cos(0.1e^x) - 0.083) \]
\[ \land (x \geq 3 \land x \leq 6 \Rightarrow y \leq -x^2 + 10x - 22.35) \]

😊 Encoding interpolants as logical combinations of linear constraints.

😊 Yielding rather complex interpolants (even of an infinite length in the worst case).
Interpolation vs. Classification

Linear interpolants can be viewed as hyperplane classifiers, [Sharma et al., CAV ’12]: sampling from $\lceil \phi \rceil$ and $\lceil \psi \rceil \rightarrow$ building a hyperplane classifier $\rightarrow$ refining by CEs.

$X^+$ and $X^-$ might not be linearly separable (often the case when sampled from nonlinear $\phi$ and $\psi$, resp.):

$$A \triangleq (x < 2.5 \Rightarrow y \geq 2 \sin(x))$$
$$\land (x \geq 2.5 \land x < 5 \Rightarrow y \geq 0.125x^2 + 0.41)$$
$$\land (x \geq 5 \land x \leq 6 \Rightarrow y \geq 6.04 - 0.5x)$$

$$B \triangleq (x < 3 \Rightarrow y \leq x \cos(0.1e^x) - 0.083)$$
$$\land (x \geq 3 \land x \leq 6 \Rightarrow y \leq -x^2 + 10x - 22.35)$$

Encoding interpolants as logical combinations of linear constraints.

Yielding rather complex interpolants (even of an infinite length in the worst case).

NIL : learning nonlinear interpolants.
Outline

1 Interpolation vs. Classification
   - Craig Interpolation
   - Binary Classification
   - Interpolants as Classifiers

2 Learning Nonlinear Interpolants
   - SVMs with Nonlinear Space Transformation
   - The NIL Algorithm and its Variants

3 Implementation and Evaluation
   - Performance over Benchmarks
   - Perturbations in Parameters

4 Concluding Remarks
   - Summary
Figure – 2-dimensional input space
Space Transformation & Kernel Trick

**Figure** – 2-dimensional input space
Space Transformation & Kernel Trick

**Figure** – 2-dimensional input space $\mapsto$ 3-dimensional feature (monomial) space with linear separation.
Space Transformation & Kernel Trick

**Figure** – 2-dimensional input space $\mapsto$ 3-dimensional feature (monomial) space with linear separation.
Space Transformation & Kernel Trick

**Figure** – 2-dimensional input space $\mapsto$ 3-dimensional feature (monomial) space with linear separation.

**Optimal-margin classifier $I$:**

$$
\sum_{i=1}^{n} \alpha_i \kappa(\tilde{x}_i, x) = 0
$$
Space Transformation & Kernel Trick

Figure – 2-dimensional input space $\mapsto$ 3-dimensional feature (monomial) space with linear separation.

Optimal-margin classifier $I$:

$$\sum_{i=1}^{n} \alpha_i \kappa(\tilde{x}_i, x) = 0$$

- Kernel function
- Support vectors
Space Transformation & Kernel Trick

Figure – 2-dimensional input space $\leftrightarrow$ 3-dimensional feature (monomial) space with linear separation.

**Optimal-margin classifier $I$:**

$$\sum_{i=1}^{n} \alpha_i \kappa(\bar{x}_i, x) = \Phi(\bar{x}_i)^T \Phi(x) = 0$$

- **kernel function**
- **support vectors**
Space Transformation & Kernel Trick

**Figure** – 2-dimensional input space $\mapsto$ 3-dimensional feature (monomial) space with linear separation.

**Optimal-margin classifier $I$:**

$$\sum_{i=1}^{n} \alpha_i \kappa(\bar{x}_i, x) = \Phi(\bar{x}_i)^T \Phi(x) = (\beta \bar{x}_i^T x + \theta)^m = 0$$

- kernel function
- support vectors
Space Transformation & Kernel Trick

**Optimal-margin classifier /:**

\[
\sum_{i=1}^{n} \alpha_i \kappa(\tilde{x}_i, x) = \Phi(\tilde{x}_i)^T \Phi(x) = (\beta \tilde{x}_i^T x + \theta)^m = 0
\]

- **Kernel function**
- **Support vectors**
- **Polynomial degree describing complexity of the monomial space**

*Figure* – 2-dimensional input space \(\mapsto\) 3-dimensional feature (monomial) space with linear separation.
The NIL Algorithm

1. Given mutually contradictory nonlinear $\phi$ and $\psi$ over common variables $x$.
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree $m$) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.
The NIL Algorithm

1. Given mutually contradictory nonlinear $\phi$ and $\psi$ over common variables $x$.
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree $m$) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.
The NIL Algorithm

1. Given mutually contradictory nonlinear $\phi$ and $\psi$ over common variables $x$.
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree $m$) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.
The NIL Algorithm

1. Given mutually contradictory nonlinear $\phi$ and $\psi$ over common variables $x$.
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree $m$) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.

$J_{\phi} \cap J_{\psi}$

© Sound, and complete when $J_{\phi} \cap J_{\psi}$ are bounded sets with positive functional margin.

§ Quantifier Elimination (QE) is involved in checking interpolants and generating CEs.

§ May not terminate in cases with zero functional margin.
The NIL Algorithm

1. Given mutually contradictory nonlinear $\phi$ and $\psi$ over common variables $x$.
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree $m$) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.

Sound, and complete when $[\phi]$ and $[\psi]$ are bounded sets with positive functional margin.
The NIL Algorithm

1. Given mutually contradictory nonlinear \( \phi \) and \( \psi \) over common variables \( x \).
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree \( m \)) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.

Sound, and complete when \([\phi]\) and \([\psi]\) are bounded sets with positive functional margin.

Quantifier Elimination (QE) is involved in checking interpolants and generating CEs \(^1\).

---

\(^1\) SMT-solving techniques over nonlinear arithmetic do not suffice.
The NIL Algorithm

1. Given mutually contradictory nonlinear $\phi$ and $\psi$ over common variables $x$.
2. Generate sample points by, e.g., (uniformly) scattering random points.
3. Find a classifier by SVMs (with kernel-degree $m$) as a candidate interpolant.
4. Refine the candidate by CEs till it being verified as a true interpolant.

Sound, and complete when $[\phi]$ and $[\psi]$ are bounded sets with positive functional margin.

Quantifier Elimination (QE) is involved in checking interpolants and generating CEs.

May not terminate in cases with zero functional margin.

---

1. SMT-solving techniques over nonlinear arithmetic do not suffice.
## Comparison with Naïve QE-Based Method

<table>
<thead>
<tr>
<th>QE-based method</th>
<th>NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logical strength</strong></td>
<td>medium ⇒ robust</td>
</tr>
<tr>
<td>strongest: $\exists y. \phi(x, y)$</td>
<td></td>
</tr>
<tr>
<td>weakest: $\forall z. \neg \psi(x, z)$</td>
<td></td>
</tr>
<tr>
<td><strong>Complexity of $I$</strong></td>
<td>simple</td>
</tr>
<tr>
<td>direct projection ⇒ complex</td>
<td></td>
</tr>
<tr>
<td>single polynomial ⇒ simple</td>
<td></td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>$n \times$ doubly exponential</td>
</tr>
</tbody>
</table>
### Comparison with Naïve QE-Based Method

<table>
<thead>
<tr>
<th></th>
<th>QE-based method</th>
<th>NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logical strength</strong></td>
<td>strongest: $\exists y. \phi(x, y)$</td>
<td>medium $\Rightarrow$ robust</td>
</tr>
<tr>
<td></td>
<td>weakest: $\forall z. \neg \psi(x, z)$</td>
<td></td>
</tr>
<tr>
<td><strong>Complexity of $I$</strong></td>
<td>direct projection $\Rightarrow$ complex</td>
<td>single polynomial $\Rightarrow$ simple</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td>doubly exponential</td>
<td>$n \times$ doubly exponential</td>
</tr>
</tbody>
</table>
## Comparison with Naïve QE-Based Method

<table>
<thead>
<tr>
<th>QE-based method</th>
<th>NIL</th>
</tr>
</thead>
</table>
| **Logical strength** | strongest: $\exists y. \phi(x, y)$  
weakest: $\forall z. \neg \psi(x, z)$ | medium $\Rightarrow$ robust |
| **Complexity of $I$** | direct projection $\Rightarrow$ complex  
single polynomial $\Rightarrow$ simple |
| **Efficiency** | doubly exponential | $n \times$ doubly exponential |
## Comparison with Naïve QE-Based Method

<table>
<thead>
<tr>
<th>QE-based method</th>
<th>NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical strength</td>
<td>strongest: $\exists y. \phi(x, y)$</td>
</tr>
<tr>
<td>weakest: $\forall z. \neg\psi(x, z)$</td>
<td></td>
</tr>
<tr>
<td>Complexity of $I$</td>
<td>direct projection $\Rightarrow$ complex</td>
</tr>
<tr>
<td>Efficiency</td>
<td>doubly exponential</td>
</tr>
</tbody>
</table>
## Comparison with Naïve QE-Based Method

<table>
<thead>
<tr>
<th>QE-based method</th>
<th>NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logical strength</strong></td>
<td></td>
</tr>
<tr>
<td>strongest: $\exists y. \phi(x, y)$</td>
<td>medium $\Rightarrow$ robust</td>
</tr>
<tr>
<td>weakest: $\forall z. \neg \psi(x, z)$</td>
<td></td>
</tr>
<tr>
<td><strong>Complexity of $\mathcal{I}$</strong></td>
<td></td>
</tr>
<tr>
<td>direct projection $\Rightarrow$ complex</td>
<td>single polynomial $\Rightarrow$ simple</td>
</tr>
<tr>
<td><strong>Efficiency</strong></td>
<td></td>
</tr>
<tr>
<td>doubly exponential</td>
<td>$n \times$ doubly exponential</td>
</tr>
</tbody>
</table>

QE + template?
## Comparison with Naïve QE-Based Method

<table>
<thead>
<tr>
<th>QE-based method</th>
<th>NIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical strength</td>
<td></td>
</tr>
<tr>
<td>strongest: $\exists y. \phi(x, y)$</td>
<td>medium $\Rightarrow$ robust</td>
</tr>
<tr>
<td>weakest: $\forall z. \neg \psi(x, z)$</td>
<td></td>
</tr>
<tr>
<td>Complexity of (I)</td>
<td></td>
</tr>
<tr>
<td>direct projection $\Rightarrow$ complex</td>
<td>single polynomial $\Rightarrow$ simple</td>
</tr>
<tr>
<td>Efficiency</td>
<td></td>
</tr>
<tr>
<td>doubly exponential</td>
<td>$n \times$ doubly exponential</td>
</tr>
</tbody>
</table>

QE + template? $\Rightarrow$ **Too many unknown parameters.**
\( \text{NIL}_\delta : \) For Cases with Zero Functional Margin

\[ \phi \quad \psi \]
NILδ: For Cases with Zero Functional Margin

NILδ is sound and complete if $J_\phi$ and $J_\psi$ are bounded sets even with zero functional margin. May not converge to an actual interpolant when $J_\phi$ or $J_\psi$ is unbounded.
NIL$\delta$: For Cases with Zero Functional Margin

May not converge to an actual interpolant when $\mathcal{J}_{\Phi}$ or $\mathcal{J}_{\Psi}$ is unbounded.
NIL$_{\delta}$: For Cases with Zero Functional Margin

- NIL is sound and complete if $J_\phi$ and $J_\psi$ are bounded sets even with zero functional margin.

- May not converge to an actual interpolant when $J_\phi$ or $J_\psi$ is unbounded.
**NIL_δ**: For Cases with Zero Functional Margin

The NIL Algorithm & its Variants

The image depicts a diagram illustrating two sets, $\phi$ and $\psi$, separated by a line. The points in $\phi$ are represented by black dots, and the points in $\psi$ are represented by white dots. The line indicates the boundary between the two sets. The notation $\delta$ represents the margin between these sets.
NIL_{\delta} : For Cases with Zero Functional Margin

\[ \delta \text{-sound, and } \delta \text{-complete if } \lfloor \phi \rfloor \text{ and } \lfloor \psi \rfloor \text{ are bounded sets even with zero functional margin.} \]
\( \text{NIL}_\delta : \) For Cases with Zero Functional Margin

- \( \delta \)-sound, and \( \delta \)-complete if \([\phi]\) and \([\psi]\) are bounded sets even with zero functional margin.

- May not converge to an actual interpolant when \([\phi]\) or \([\psi]\) is unbounded.
NIL*_{δ,B} : For Unbounded Cases with Varying Tolerance
\textbf{NIL}_{\delta, B}^* : For Unbounded Cases with Varying Tolerance
The NIL Algorithm & its Variants

$\text{NIL}_{\delta,B}^*$: For Unbounded Cases with Varying Tolerance

$\phi, \psi$
NIL$^*_{\delta,B}$: For Unbounded Cases with Varying Tolerance
NIL^*_\delta,B : For Unbounded Cases with Varying Tolerance
The NIL Algorithm & its Variants

\[ \text{NIL}_{\delta, B}^* : \text{For Unbounded Cases with Varying Tolerance} \]
**NIL**\(\delta, B\) : For Unbounded Cases with Varying Tolerance
NIL: For Unbounded Cases with Varying Tolerance

The sequence of candidate interpolants converges to an actual interpolant.
Outline

1. Interpolation vs. Classification
   - Craig Interpolation
   - Binary Classification
   - Interpolants as Classifiers

2. Learning Nonlinear Interpolants
   - SVMs with Nonlinear Space Transformation
   - The NIL Algorithm and its Variants

3. Implementation and Evaluation
   - Performance over Benchmarks
   - Perturbations in Parameters

4. Concluding Remarks
   - Summary
NIL: an open-source tool in Wolfram Mathematica.

- LIBSVM: SVM classifications;
- Reduce²: verification of candidate interpolants;
- FindInstance: generation of counterexamples;
- Rational recovery: rounding off floating-point computations [Lang, Springer NY ’12].

## Benchmark Examples

### Benchmark Examples Table

<table>
<thead>
<tr>
<th>Category</th>
<th>ID</th>
<th>Name</th>
<th>(\varphi)</th>
<th>(\psi)</th>
<th>(\ell)</th>
<th>Time/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dummy</td>
<td>1</td>
<td>(x &lt; -1)</td>
<td>(y - x^2 - 1 &lt; 0)</td>
<td>(x \geq 1)</td>
<td>(y + x^2 + 1 = 0)</td>
<td>(x &lt; 0)</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Necklace</td>
<td>((-4 + 4)x^2 + y^2 - 1 \leq 0)</td>
<td>((-4 + 4)x^2 + y^2 - 9 \geq 0)</td>
<td>((-4 + 4)x^2 + y^2 - 9 \geq 0)</td>
<td>(-y &lt; 0)</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Face</td>
<td>((-1)^2 - 2x^2 + 3x - y^2)</td>
<td>((-1)^2 + y^2 - 4 \leq 0)</td>
<td>((-1)^2 + y^2 - 1 \geq 0)</td>
<td>(x \geq 0)</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Twisted</td>
<td>(1/20( (-1)^2 - y^2 ) + 2x^2 - y^2)</td>
<td>(x^2 + 1/6( x^2 + 2x^2 + y^2 ))</td>
<td>(x^2 + y^2 - 4 \leq 0)</td>
<td>1.25</td>
</tr>
</tbody>
</table>

### Learning Nonlinear Interpolants

- Benchmark Examples

### Implementation & Evaluation

- Benchmark Examples

### Concluding Remarks

- Benchmark Examples

---

**Mingshuai Chen** · i2, RWTH Aachen Univ.  
**Interpolation over Nonlinear Arithmetic**  
**MOVES Seminar · Aachen · 2019**  
19 / 25
Beyond the scope of concave quadratic formulas as required in [Gan et al., IJCAR’16]:
Adjacent and sharper cases as in [Okudono et al., APLAS’17]:

![Diagram 1](image1.png)

![Diagram 2](image2.png)
Formulas sharing parallel or coincident boundaries:
Visualizations in *NIL*

Transcendental cases from [Gao & Zufferey, TACAS’16] and [Kupferschmid & Becker, FORMATS’11], yet with simpler interpolants:
Three-dimensional case from [Dai et al., CAV ’13], yet with simpler interpolants:
### Interpolants of Simpler Forms

<table>
<thead>
<tr>
<th>Name</th>
<th>Interpolants by NIL</th>
<th>Interpolants from the sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJCAR16-1</td>
<td>$1 - \frac{3x_1}{4} - \frac{x_2}{2} &lt; 0$</td>
<td>$-3 + 2x_1 + x_2^2 + \frac{1}{2}x_2^2 &gt; 0$</td>
</tr>
<tr>
<td>CAV13-1</td>
<td>$-1 + \frac{x_2}{2} - \frac{y}{3} + \frac{xy}{3} - \frac{y^2}{4} &lt; 0$</td>
<td>$436.45(x^2 - 2y^2 - 4) + \frac{1}{2} \leq 0$</td>
</tr>
<tr>
<td></td>
<td>$105x^4 + x^2(140y^2 + 24y(5z + 7) + 35z(3z + 8)) +$</td>
<td>$-14629.26 + 2983.44x_3 + 10972.97x_3^2 +$</td>
</tr>
<tr>
<td></td>
<td>$2(70y^3 z + 5y^2 (12z^2 + 21z + 28) - 14y(6z^3 + 5z^2 + 10) - 35(3z^4 + 8z^2 + 4z - 9)) &lt; 14(x(20x^2(z + 1) + 10y^2(z + 2) - 3y(4z^2 - 5z + 4) - 20z(z^2 + 2))$</td>
<td>$297.62x_2 + 297.64x_2x_3 + 0.02x_2x_3^2 + 9625.61x_2 - 1161.80x_2x_3 + 0.01x_2x_3^2 + 811.93x_3^2 +$</td>
</tr>
<tr>
<td>CAV13-2</td>
<td>$-1 + \frac{2\text{c}_{199}}{99} &lt; 0$</td>
<td>$2745.14x_3^2 - 10648.11x_1 + 3101.42x_1x_3 +$</td>
</tr>
<tr>
<td>CAV13-3</td>
<td>$2 + y &lt; y^2$</td>
<td>$8y^2 - 68y - 102 \geq 0$</td>
</tr>
<tr>
<td>Sharper-1</td>
<td>$y &gt; 0$</td>
<td>$8y + 4x^2 &gt; 0$</td>
</tr>
<tr>
<td>Sharper-2</td>
<td>$x_1 &lt; x_2$</td>
<td>$-x_1 + x_2 &gt; 0$</td>
</tr>
<tr>
<td>IJCAR16-2</td>
<td>$2x_2 + 4y &gt; 5$</td>
<td>$716.77 + 1326.74(ya) + 1.33(ya)^2 + 433.90(ya)^3 +$</td>
</tr>
<tr>
<td>CAV13-4</td>
<td>$15x^2 &lt; 4 + 20y$</td>
<td>$668.16(xa) - 155.86(xa)(ya) + 317.29(xa)(ya)^2 + 222.00(xa)^2 + 592.39(xa)^2(ya) + 271.11(xa)^3 &gt; 0$</td>
</tr>
<tr>
<td>TACAS16</td>
<td>$y &gt; 1.8 \lor (0.59 \leq y \leq 1.8 \land -1.35 \leq x \leq 1.35) \lor$</td>
<td>$y \geq 0 \land -0.3 \leq x \leq 0.3$</td>
</tr>
<tr>
<td></td>
<td>$(0.09 \leq y &lt; 0.59 \land -0.77 \leq x \leq 0.77) \lor$</td>
<td></td>
</tr>
</tbody>
</table>

Mingshuai Chen · i2, RWTH Aachen Univ.  Interpolation over Nonlinear Arithmetic  MOVES Seminar · Aachen · 2019  Concluding Remarks
## Interpolants of Simpler Forms

<table>
<thead>
<tr>
<th>Name</th>
<th>Interpolants by NIL</th>
<th>Interpolants from the sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>IJCAR16-1</td>
<td>$1 - \frac{3x_1}{4} - \frac{x_2}{2} &lt; 0$</td>
<td>$-3 + 2x_1 + x_2^2 + \frac{1}{2}x_2^2 &gt; 0$</td>
</tr>
<tr>
<td>CAV13-1</td>
<td>$-1 + \frac{x_2}{2} - \frac{y}{3} + \frac{xy}{3} - \frac{y^2}{4} &lt; 0$</td>
<td>$436.45(x^2 - 2y^2 - 4) + \frac{1}{2} \leq 0$</td>
</tr>
<tr>
<td>CAV13-2</td>
<td>$105x^4 + x^2(140y^2 + 24y(5z + 7) + 35z(3z + 8)) + 2(70y^3z + 5y^2(12z^2 + 21z + 28) - 14y(6z^3 + 5z^2 + 10) - 35(3z^4 + 8z^2 + 4z - 9)) &lt; 14x(20x^2(z + 1) + 10y^2(z + 2) - 3y(4z^2 - 5z + 4) - 20z(z^2 + 2))$</td>
<td>$-14629.26 + 2983.44x_3 + 10972.97x_3^2 + 297.62x_2 + 297.64x_2x_3 + 0.02x_2x_3^2 + 9625.61x_2 - 1161.80x_2x_3^2 + 0.01x_2x_3^2 + 811.93x_3^2 + 2745.14x_2^2 - 10648.11x_1 + 3101.42x_1x_3 + 8646.17x_1x_3^2 + 511.84x_1x_2x_3 + 1034x_1x_2x_3 + 0.02x_1x_2x_3^2 + 9233.66x_1x_2^2 + 1342.55x_1x_2^2x_3 - 138.70x_1x_2^2 + 11476.61x_1^2 - 3737.70x_1^2x_3 + 4071.65x_1^2x_3^2 - 2153.00x_1x_2x_3 + 373.14x_1^2x_2x_3 + 7616.18x_1^2x_2^2 + 8950.77x_1^3 + 1937.92x_1^3x_3 - 64.07x_1^3x_2 + 4827.25x_1^4 &gt; 0$</td>
</tr>
<tr>
<td>CAV13-3</td>
<td>$-1 + \frac{2v_1}{99} &lt; 0$</td>
<td>$-1.3983v_1 + 69.358 &gt; 0$</td>
</tr>
<tr>
<td>Sharper-1</td>
<td>$2 + y &lt; y^2$</td>
<td>$34y^2 - 68y - 102 \geq 0$</td>
</tr>
<tr>
<td>Sharper-2</td>
<td>$y &gt; 0$</td>
<td>$8y + 4x^2 &gt; 0$</td>
</tr>
<tr>
<td>IJCAR16-2</td>
<td>$x_1 &lt; x_2$</td>
<td>$-x_1 + x_2 &gt; 0$</td>
</tr>
<tr>
<td>CAV13-4</td>
<td>$2xa + 4ya &gt; 5$</td>
<td>$716.77 + 1326.74(ya) + 1.33(ya)^2 + 433.90(ya)^3 + 668.16(xa) - 155.86(xa)(ya) + 317.29(xa)(ya)^2 + 222.00(xa)^2 + 592.39(xa)^2(ya) + 271.11(xa)^3 &gt; 0$</td>
</tr>
<tr>
<td>TACAS16</td>
<td>$15x^2 &lt; 4 + 20y$</td>
<td>$y &gt; 1.8 \lor (0.59 \leq y \leq 1.8) \land (-1.35 \leq x \leq 1.35) \lor (0.09 \leq y &lt; 0.59 \land -0.77 \leq x \leq 0.77) \lor (y &gt; 0 \land -0.3 &lt; x \leq 0.3)$</td>
</tr>
</tbody>
</table>
Perturbation-Resilient Interpolants

(a) $\epsilon$-perturbations in the radii

(b) Interpolant resilient to $\epsilon$-perturbations

Figure – Introducing $\epsilon$-perturbations (say with $\epsilon$ up to 0.5) in $\phi$ and $\psi$. The synthesized interpolant is hence resilient to any $\epsilon$-perturbation in the radii satisfying $-0.5 \leq \epsilon \leq 0.5$. 
Outline

1. Interpolation vs. Classification
   - Craig Interpolation
   - Binary Classification
   - Interpolants as Classifiers

2. Learning Nonlinear Interpolants
   - SVMs with Nonlinear Space Transformation
   - The NIL Algorithm and its Variants

3. Implementation and Evaluation
   - Performance over Benchmarks
   - Perturbations in Parameters

4. Concluding Remarks
   - Summary
Concluding Remarks

Problem: We face that

- polynomial constraints have been shown useful to express invariant properties for programs and hybrid systems,
- little work on synthesizing nonlinear interpolants, which either restricts the input formulae or yields complex results.
Concluding Remarks

**Problem:** We face that

- polynomial constraints have been shown useful to express invariant properties for programs and hybrid systems,
- little work on synthesizing nonlinear interpolants, which either restricts the input formulae or yields complex results.

**Status:** We present

- a unified, counterexample-guided method for generating polynomial interpolants over the general quantifier-free theory of nonlinear arithmetic,
- soundness of NIL, and sufficient conditions for its completeness and convergence,
- Experimental results indicating that our method suffices to address more interpolation tasks, including those with perturbations in parameters, and in many cases synthesizes simpler interpolants.
Concluding Remarks

Problem: We face that

- polynomial constraints have been shown useful to express invariant properties for programs and hybrid systems,
- little work on synthesizing nonlinear interpolants, which either restricts the input formulae or yields complex results.

Status: We present

- a unified, counterexample-guided method for generating polynomial interpolants over the general quantifier-free theory of nonlinear arithmetic,
- soundness of NIL, and sufficient conditions for its completeness and convergence,
- Experimental results indicating that our method suffices to address more interpolation tasks, including those with perturbations in parameters, and in many cases synthesizes simpler interpolants.

Future Work: We plan to

- improve the efficiency of NIL by substituting the general purpose QE procedure with alternative methods,
- combine nonlinear arithmetic with EUFs, by resorting to, e.g., predicate-abstraction techniques,
- investigate the performance of NIL over different classification techniques, e.g., the widespread regression-based methods.
Probabilistic Craig Interpolants?
Probabilistic Craig Interpolants?

- Generalized Craig Interpolation for stochastic-SAT: resolution-based BMC of MDPs.

- Generalized Craig Interpolation for stochastic-SMT: resolution-based UMC of PHA.