Implementation of Uniform Interpolating Algorithms

Master thesis defense

Jose Abel Castellanos Joo

October 26, 2020

Table of Contents I

Table of Contents II

Applications of Interpolants and Uniform Interpolants

- Guide bounded model checking algorithms [7]
- Invariant generation for quantifier-free theories [8]
- Strongest interpolants [5]
- Optimization of ontologies [6] (i.e. elimination of non-relevant predicates)
- Privacy-protecting ontologies [4] (i.e. predicate hiding)

First Order Interpolants

Definition

Let Σ be a first-order signature. Let $\mathcal{T}$ be a $\Sigma-$theory in first-order logic. Given two logical formulas α and β such that $\models_{\mathcal{T}} \alpha \wedge \beta \rightarrow \bot$, an interpolant γ for (α, β) satisfies:

- $\models_{\mathcal{T}} \alpha \rightarrow \gamma$
- $\models_{\mathcal{T}} \beta \wedge \gamma \rightarrow \bot$
- γ refers only $\alpha\beta -$ *common* symbols.

Theorem (Craig Interpolation Theorem)

For any ψ, ϕ first-order logic formulas such that $\models \psi \wedge \phi \rightarrow \bot$, the interpolant γ of (ψ, ϕ) exists.

Example

Let $\alpha = \neg(P \wedge Q) \rightarrow (\neg R \wedge Q)$ and $\beta = \neg(S \rightarrow P) \wedge \neg(S \rightarrow \neg R)$

The interpolant of (α, β) is $\alpha[Q/\top] \vee \alpha[Q/\bot] \cong R \rightarrow P$

1. $\neg(P \wedge Q) \rightarrow (\neg R \wedge Q)$
2. $\neg(P \wedge Q) \rightarrow \neg(R \vee \neg Q)$
3. $(R \vee \neg Q) \rightarrow (P \wedge Q)$
4. $(Q \rightarrow R) \rightarrow (P \wedge Q)$
5. $R \rightarrow (Q \rightarrow R)$
6. $R \rightarrow (P \wedge Q)$
7. $(P \wedge Q) \rightarrow P$
8. $R \rightarrow P$

1. $R \rightarrow P$
2. $\neg(S \rightarrow P)$
3. $\neg(S \rightarrow \neg R)$
4. $S \wedge \neg P$
5. $S \wedge R$
6. P
7. $\bot$

Uniform Interpolants

Definition

Let $\mathcal{T}$ be a theory and an existential formula $\exists \underline{e}.\phi(\underline{e}, \underline{z})$; call *Res*$(\exists \underline{e}.\phi(\underline{e}, \underline{z}))$ the set $\{\theta(\underline{y}, \underline{z}) | \models_{\mathcal{T}} \exists \underline{e}.\phi(\underline{e}, \underline{z}) \to \theta(\underline{y}, \underline{z})\}$.
A quantifier-free formula $\psi(\underline{z})$ is called a $\mathcal{T}$-uniform interpolant if

- $\psi(\underline{z}) \in Res(\exists \underline{e}.\phi(\underline{e}, \underline{z}))$
- $\forall \theta(\underline{y}, \underline{z}) \in Res(\exists \underline{e}.\phi(\underline{e}, \underline{z})). \models_{\mathcal{T}} \psi(\underline{z}) \to \theta(\underline{y}, \underline{z})$

Key ideas / Steps

- Preprocessing: Flatten formula by introducing new constants
- Phase I: Elimination of uncommon constants
- Phase II: Elimination of uncommon function symbols
- Phase III: Elimination of uncommon symbols conditionally
- Phase IV: Interpolant generation

Preprocessing

The algorithm introduces new constants by equating each subexpression in the AST of the original input.

For each sub-term t in the input formula assign a fresh unique constant $\mathfrak{a}_t$. Additionally, for each sub-term t generate new equations of the form:

- $c = \mathfrak{a}_c$, if t is a constant c
- $f(\mathfrak{a}_{t_1}, \ldots, \mathfrak{a}_{t_n}) = \mathfrak{a}_{f(t_1,\ldots,t_n)}$, if t is a function application of the form $f(t_1, \ldots, t_n)$

Elimination of uncommon constant

This step builds an equivalence relation $\mathcal{E}$ of the $f - equations$ introduced in the Flattening step using a congruence closure algorithm such that the representatives are common terms whenever possible.

Uncommon terms appearing in the current conjunction of equations are replaced by their representatives.

Elimination of uncommon function symbols

This step produces for all pairs of $f - \mathit{equations}$ $(f(\mathfrak{a}_1, \ldots, \mathfrak{a}_n) = \mathfrak{c}, f(\mathfrak{b}_1, \ldots, \mathfrak{b}_n) = \mathfrak{d})$ Horn clauses of the form $\bigwedge_{i=1}^{n}(\mathit{repr}_{\mathcal{E}}(\mathfrak{a}_i) = \mathit{repr}_{\mathcal{E}}(\mathfrak{b}_i)) \rightarrow \mathit{repr}_{\mathcal{E}}(\mathfrak{c}) = \mathit{repr}_{\mathcal{E}}(\mathfrak{d})$ when either of the two following situations happen:

- The outermost symbol of the $f - \mathit{equations}$ is an uncommon symbol.
- There is at least one constant argument in any of the $f - \mathit{equations}$ that is an uncommon constant.

Elimination of uncommon symbols conditionally

We identify the Horn clauses $h := \bigwedge_i (c_i = d_i) \rightarrow a = b$ that have *common antecedents* and uncommon head equations. We perform the following procedure:

- if a and b are both uncommon terms: replace the equation $a = b$ appearing in the antecedents of all the current Horn clauses by $antecedent(h)$.
- if either a is common and b uncommon: replace b by a in all the current Horn clauses h' and append $antecedent(h)$ to $antecedent(h')$.
- if either a is uncommon and b common: Proceed similarly as in the previous case.

We repeat this step until we cannot produce any new Horn clauses.

Interpolant generation

For each Horn clause of the form $\bigwedge_i (a_i = b_i) \rightarrow u = c$ where the antecedent is common, the term u in its head equation is an uncommon term, and the term c is a common term, replace every instance of u appearing in each $f - equation$ by c to generate Horn clauses with antecedent $\bigwedge_i a_i = b_i$.

Return the conjunction of formulas obtained as the interpolant.

Combining two data structures I

The modification of Phase III implemented in this thesis work combines and extends the algorithms and data structures introduced in [2, 9].

The implementation of the congruence closure algorithm in [9] extends the usual *Find*, *Merge* operations on the union-find data structure with the *Explain* operator.

Combining two data structures II

Conditional Congruence Closure Algorithm:

Step 1. Let $\mathcal{E}$ be an empty equivalence class for all the terms in the term tree of the Horn clauses in H, i.e. there is an equivalence class for each term in H.

Step 2. Insert all the Horn clauses in H to the Gallier data structure and update $\mathcal{E}$ according to Gallier's algorithm.

Step 3. For all the common equations $a = b$ in the Horn clauses $h \in H$, Merge a and b in $\mathcal{E}$. Update Gallier's data structure accordingly.

Step 4. Return $\mathcal{E}$ as the conditional congruence closure for H.

A Simple Example

Let us consider $\alpha = \{f(x_1) \neq f(x_2)\}$ with the set of symbols to eliminate $U = \{f\}$.

The implementation produces the following trace for $\alpha; U$:

Output of implementation

```
Before conditionalEliminationEqs
Horn clauses produced
0. 0x5648882bc710 (Leader) (= c_x2 c_x1) -> (= (a_f c_x2) (a_f c_x1))
1. 0x5648882d7dd0 (Leader) (= (a_f c_x2) (a_f c_x1)) -> false
Number of horn clauses: 2
Executing conditionalElimination
After conditionalEliminationEqs/Before conditionalEliminationHcs
Horn clauses produced
0. 0x5648882bc710 (Leader) (= c_x2 c_x1) -> (= (a_f c_x2) (a_f c_x1))
1. 0x5648882d7dd0 (Leader) (= (a_f c_x2) (a_f c_x1)) -> false
Number of horn clauses: 2
Executing conditionalEliminationfor Horn clauses
After conditionalEliminationHcs
Horn clauses produced
0. 0x5648882bc710 (Leader) (= c_x2 c_x1) -> (= (a_f c_x2) (a_f c_x1))
1. 0x5648882d7dd0 (Leader) (= (a_f c_x2) (a_f c_x1)) -> false
Number of horn clauses: 2
Horn clauses produced
0. 0x5648882deea0 (Leader) (= c_x2 c_x1) -> false
Number of horn clauses: 1
(ast-vector
(=> (= x2 x1) false))
```

Discussing the Output

It is clear that $f(x_1) \neq f(x_2) \models x_1 \neq x_2$

Lemma

$x_1 \neq x_2$ implies any θ such that $f(x_1) \neq f(x_2) \models \theta$

Proof.

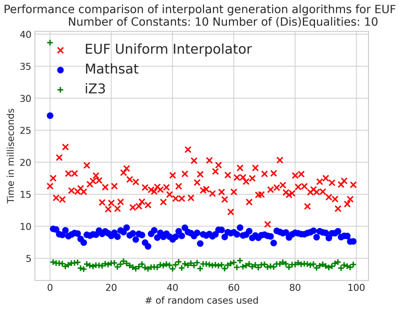
Proof by induction on θ

- Base case: $f(x_1) \neq f(x_2) \models x_1 \neq x_2 \wedge x_1 = x_1 \wedge x_2 = x_2$
- Inductive step:
  - Case $f(x_1) \neq f(x_2) \models \psi \wedge \phi$: Thus, $f(x_1) \neq f(x_2) \models \psi$, $f(x_1) \neq f(x_2) \models \phi$. By IH. $x_1 \neq x_2 \rightarrow \psi$ and $x_1 \neq x_2 \rightarrow \phi$ Thus, $x_1 \neq x_2 \rightarrow \psi \wedge \phi$
  - Case $f(x_1) \neq f(x_2) \models \psi \vee \phi$: EUF is convex. W.L.O.G., $f(x_1) \neq f(x_2) \models \phi$. By IH. $x_1 \neq x_2 \rightarrow \phi$. Thus, $x_1 \neq x_2 \rightarrow \psi \vee \phi$.

Performance comparison with other interpolating systems

It is important to mention that iZ3 and Mathsat *do not generate uniform interpolants*.

The examples generated where randomly created using function symbols with arity less than or equal to 3, but not equal to 1.



Key ideas / Steps

Kapur's algorithm uses these inference rules:

$$\frac{ax + ax \leq c \qquad a \in \{-1, 0, 1\} \text{ and } x \in Vars}{ax \leq \lfloor \frac{c}{2} \rfloor} \text{ Normalize}$$

$$\frac{s_1x_1 + s_2x_2 \leq c_1 \qquad -s_2x_2 + s_3x_3 \leq c_2}{s_1x_1 + s_3x_3 \leq c_1 + c_2} \text{ Elim}$$

Data structures implemented

- Indexing data structure which encodes inequalities of the input formula using natural numbers.
  $Position : \text{ UTVPI-term } \rightarrow \mathbb{N}$ is a bijection.
- Array of numbers *Bounds* indexed by the numeral representation of the inequalities representing the minimum bound of the encoded inequality, i.e.
  $Bounds[Position(a_1 x_1 + a_2 x_2)] = c$
- Data structure to keep track of the signs of variables to be eliminated in the inequalities for efficient matching.
- Data structure to represent an UTVPI term in normal form endowed with addition and subtraction operations and the *Position* function mentioned in the first item.

A Simple Example

Let us consider $\alpha = \{-x_2 - x_1 + 3 \geq 0, x_1 + x_3 + 1 \geq 0, -x_3 - x_4 - 6 \geq 0, x_5 + x_4 + 1 \geq 0\}$; $U = \{x_1\}$.

Our implementation produced the following output:

$$x_2 - x_3 \leq 4 \wedge x_4 + x_3 \leq -6 \wedge -x_5 - x_4 \leq 1$$

Output of implementation

```
Processing
(+ c_x2 a_x1)
Updating structure with
x_2 + x_1 <= 3
Processing
(- (- c_x3) a_x1)
Updating structure with
- x_3 - x_2 <= 1
Processing
(+ c_x4 c_x3)
Updating structure with
x_4 + x_3 <= -6
Processing
(- (- c_x5) c_x4)
Updating structure with
- x_5 - x_4 <= 1
Removing this var: x_0
Removing this var: x_1
Removing this var: x_2
Reducing x_2 + x_1 and - x_3 - x_2
Result: - x_3 + x_1
Removing this var: x_3
Removing this var: x_4
Removing this var: x_5
(ast-vector
  (<= (+ (- x3) x2) 4)
  (<= (+ x4 x3) (- 6))
(<= (- (- x5) x4) 1))
```

Discussing the output

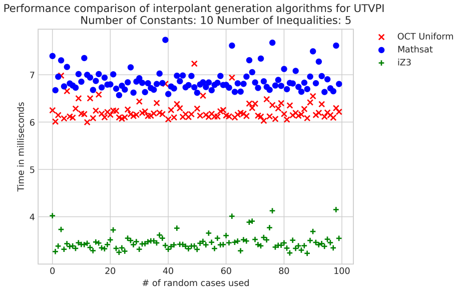
It is easy to see that
$\alpha \models_{UTVPI} x_2 - x_3 \leq 4 \wedge x_4 + x_3 \leq -6 \wedge -x_5 - x_4 \leq 1.$

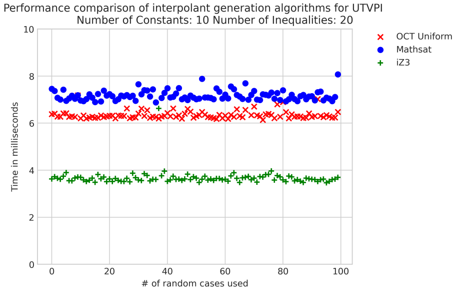
Using Fourier-Motzkin, we can check that
$\models_{UTVPI} \alpha \iff \models_{UTVPI} x_2 - x_3 \leq 4 \wedge x_4 + x_3 \leq -6 \wedge -x_5 - x_4 \leq 1$

Thus, for every θ such that $\models_{UTVPI} \alpha \rightarrow \theta$ we have that
$\models_{UTVPI} x_2 - x_3 \leq 4 \wedge x_4 + x_3 \leq -6 \wedge -x_5 - x_4 \leq 1 \rightarrow \theta$

Performance comparison with other interpolating systems

It is important to mention that iZ3 and Mathsat *do not generate uniform interpolants for this theory*.





The combined theory does not have the UI property

In previous chapters, we discussed uniform interpolanting algorithms for the EUF and UTVPI theories respectively. This was possible since each theory satisfies the Uniform Interpolant Property (UIP).

Nonetheless, as shown in [1], the UIP does not hold for combined theory of EUF and integer difference logic (IDL).

The same counter-example applies for the theory combination of EUF and UTPVI. Hence, there cannot be an algorithm for computing the uniform interpolant for this combined theory.

Weakening condition

This weakening condition allows prove soundness in the proposed algorithm:

For all variables x to eliminate in the UTVPI components of the input formula ψ, either:

- $\psi \models_{EUF+UTVPI} x \leq n_1$ and $\psi \models_{EUF+UTVPI} -x \leq n_2$ where $n_1, n_2 \in \mathbb{Z}$, or
- There exists $a_1x + a_2y$ with y a common variable such that $\psi \models_{EUF+UTVPI} a_1x + a_2y \leq n_1$ and $\psi \models_{EUF+UTVPI} -a_1x - a_2y \leq n_2$, where $a_1, a_2 \in \{-1, 0, 1\}$ and $n_1, n_2 \in \mathbb{Z}$

UI algorithm for EUF [3]

1. Simplification rules.

2. DAG update rule.

3. e-Free Literal rule.

4. Branch equalities or disequalities from the difference set of compatible $f - equations$.

Partially sound UI algorithm for EUF + UTVPI I

5. Eliminate uncommon UTVPI terms: if there are UTVPI inequalities of the form $a_i x + a_j e_j \leq k_1$ and $a_k y - a_j e_j \leq k_2$ in ψ, then introduce to ϕ the UTVPI inequality $a_i x + a_k y \leq k_1 + k_2$.
6. Normalize UTVPI inequalities: if there is a UTVPI inequality of the form $a_i x + a_i x \leq k$ in the formula state, then remove it and insert to ψ the UTVPI inequality $a_i x \leq \lfloor k/2 \rfloor$
7. Normalize bounds: if there are two UTVPI inequalities of the form $a_i x + a_j y \leq k_1$, $a_i x + a_j y \leq k_2$ in the formula state with $\{k_1, k_2\} \in \mathbb{N}$, then remove them both and insert to ψ the UTVPI inequality $a_i x + a_j y \leq min(k_1, k_2)$

Partially sound UI algorithm for EUF + UTVPI II

8. Propagate fully bounded uncommon UTVPI inequalities: if there are two UTVPI inequalities in ψ of the form $a_i e_i + a_j e_j \leq k_1$ and $-a_i e_i - a_j e_j \leq k_2$, where $a_i \in \{1, -1\}, a_j \in \{1, 0, -1\}$ and $i > j$ or e_i is uncommon and e_j is common then non-deterministically apply the following rule:

- Remove both $a_i e_i + a_j e_j \leq k_1$ and $-a_i e_i - a_j e_j \leq k_2$ from ψ and replace every e_i by $l - a_i a_j e_j$ where $l \in \{-a_j k_2, -a_j k_2 + 1, \ldots, a_j k_1 - 1, a_j k_1\}$.

Simple Example I

Let us consider the following input formula $\{y - x \leq 0, -y + x \leq 10 y + x \leq 20, -y - x \leq -10, -e + x \leq 0, e - y \leq 0, f(e) = x\}$; $U = \{e\}$.

The normal form produced for the UTVPI component of the proposed algorithm is the following conjunction of inequalities: $x \leq 15, -x \leq -5, y \leq 10, y + x \leq 20, y - x \leq 0, -y \leq 0, -y + x \leq 10, -y - x \leq -10, e \leq 10, e + x \leq 25, e - x \leq 5, e + y \leq 20, e - y \leq 10, -e \leq -5, -e + x \leq 10, -e - x \leq -10, -e + y \leq 5, -e - y \leq -5$

Simple Example II

The final output produced by the algorith is

$$(f(5) = x \wedge \delta) \vee (f(6) = x \wedge \delta) \vee (f(7) = x \wedge \delta) \\ \vee (f(8) = x \wedge \delta) \vee (f(9) = x \wedge \delta) \vee (f(10) = x \wedge \delta)$$

where δ is $x \leq 15 \wedge -x \leq 5 \wedge y \leq 10 \wedge y + x \leq 20 \wedge y - x \leq 0 \wedge -y \leq 0 \wedge -y + x \leq 10 \wedge -y - x \leq -10$

Yorsh - Musuvathi combination framework

The implementation maintains a map data structure that keeps track of the *partial interpolants*. This ensures that the base case for the above formula $p(c)$ is replaced by previous clauses as required in [10].

These *partial interpolants* are computed from an unsatisfiability proof obtained by including the negation of the disjunction to the formula using the following definition

Partial Interpolants

Definition

Let $\langle A, B \rangle$ be a pair of clause sets such that $A \wedge B \vdash \bot$ and $\mathcal{T}$ be a proof of unsatisfiability of $A \wedge B$. We define $p(c)$ for a clause c in $\mathcal{T}$ by induction on the proof structure:

- if c is one of the input clauses then
    - if $c \in A$, then $p(c) := \bot$
    - if $c \in B$, then $p(c) := \top$
- otherwise, c is a result of resolution, i.e. let c_1, c_2 be two clauses of the form $x \vee c_1^{'}$, $\neg x \vee c_2^{'}$ respectively. The partial interpolant for c is defined as follows:
    - if $x \in A$ and $x \notin B$ (x is A-local), then $p(c) := p(c_1) \vee p(c_2)$
    - if $x \notin A$ and $x \in B$ (x is B-local), then $p(c) := p(c_1) \wedge p(c_2)$
    - otherwise (x is AB-common), then $p(c) := (x \vee p(c_1)) \wedge (\neg x \vee p(c_2))$

Implemented Modifications

Since introducing negations is necessary to compute partial interpolants, we noticed the following interaction with the theories involved in the thesis work:

- EUF case: negations of literals in this theory are just dis-equalities, which the interpolation algorithm implemented handles as Horn clauses with a false head term.
- UTVPI case: negations of literals in this theory are either dis-equalities or strict inequalities. The dis-equalities are purified an appended to the EUF component; strict inequalities of the form $x > y$ are replaced by non-strict inequalities of the form $x \geq y + 1$.

Conclusions

- The implementation and testing work in the implementation for the EUF and UTVPI theories confirm that the approach produces stronger interpolants compared to iZ3 and Mathsat.
- Uniform interpolantion for the theories studied is comparable in performance with well-known interpolant generating implementations.
- A new partially sound algorithm for the uniform interpolant generation for the combined theory of EUF + UTVPI was introduced in the sense that the algorithm is sound if the input formula satisfies particular requirements.

Future Work

- In order to improve the UTVPI implementation, it will be interesting to explore the use of heuristics to determine the order in which the uncommon variable should be eliminated.
- The implementation of the partially sound combination algorithm will be considered in the future if rule 8 can be replaced for propagation rules that avoid splitting.

References I

Diego Calvanese, Silvio Ghilardi, Alessandro Gianola, Marco Montali, and Andrey Rivkin.
Combined covers and beth definability.
In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning*, pages 181–200, Cham, 2020. Springer International Publishing.

Jean H. Gallier.
Fast algorithms for testing unsatisfiability of ground horn clauses with equations.
Journal of Symbolic Computation, 4(2):233 – 254, 1987.

References II

Silvio Ghilardi, Alessandro Gianola, and Deepak Kapur.
Compactly representing uniform interpolants for euf using (conditional) dags, 2020.

Bernardo Cuenca Grau and Boris Motik.
Pushing the limits of reasoning over ontologies with hidden content.
In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning*, KR'10, page 214–224. AAAI Press, 2010.

References III

Deepak Kapur.
Conditional congruence closure over uninterpreted and interpreted symbols.
Journal of Systems Science and Complexity, 32:317–355, 02 2019.

C. Lutz and F. Wolter.
Foundations for uniform interpolation and forgetting in expressive description logics.
In *IJCAI*, 2011.

References IV

K. L. McMillan.
Interpolation and sat-based model checking.
In Warren A. Hunt and Fabio Somenzi, editors, *Computer Aided Verification*, pages 1–13, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

Kenneth McMillan.
Interpolants from z3 proofs.
In *Formal Methods in Computer-Aided Design*, October 2011.

Robert Nieuwenhuis and Albert Oliveras.
Proof-producing congruence closure.
In Jürgen Giesl, editor, *Term Rewriting and Applications*, pages 453–468, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

References V

Greta Yorsh and Madanlal Musuvathi.
A combination method for generating interpolants.
In Robert Nieuwenhuis, editor, *Automated Deduction – CADE-20*, pages 353–368, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.