LRA Interpolants from No Man’s Land

Leonardo Alt, Antti E. J. Hyvärinen, and Natasha Sharygina
University of Lugano, Switzerland
THE LEGEND OF GNOME ANN

TIME AND TIDE WAIT FOR GNOME ANN.

THE WICKED FLEE WHEN GNOME ANN PURSUETH.
—PROVERBS 28:1

WHAT THEREFORE GOD HATH JOINED TOGETHER, LET GNOME ANN PUT ASUNDER.
—MARK 10:9

TIME RIPENS ALL THINGS; GNOME ANN IS BORN WISE.
—MIGUEL DE CERVANTES

OUR MISSION: TO BOLDLY GO WHERE GNOME ANN HAS GONE BEFORE.

FOOL! NO MAN CAN KILL ME.
I AM GNOME ANN!
Motivation

The goal: Finding the right proof

The tool: Make interpolation on LRA more flexible

The application: LRA for abstractions in software model checking

The keywords: SMT solving, function summaries, labeled interpolation systems
Interpolants

Given two formulas $A$ and $B$ such that

$$A \land B \rightarrow \bot$$

an interpolant is a formula $I$ such that

$$\text{Vars}(I) \subseteq \text{Vars}(A) \cap \text{Vars}(B)$$
$$A \rightarrow I$$
$$I \land B \rightarrow \bot$$
Interpolants

Given two formulas $A$ and $B$ such that

$$A \land B \to \bot$$

an interpolant is a formula $I$ such that

$$\text{Vars}(I) \subseteq \text{Vars}(A) \cap \text{Vars}(B)$$

$$A \to I$$

$$I \land B \to \bot$$

[Diagram of set inclusion]
Interpolants

Given two formulas $A$ and $B$ such that

$$A \land B \rightarrow \bot$$

an interpolant is a formula $I$ such that

$$\text{Vars}(I) \subseteq \text{Vars}(A) \cap \text{Vars}(B)$$

$$A \rightarrow I$$

$$I \land B \rightarrow \bot$$
Interpolants

Given two formulas $A$ and $B$ such that

$$A \land B \rightarrow \bot$$

an interpolant is a formula $I$ such that

$$\text{Vars}(I) \subseteq \text{Vars}(A) \cap \text{Vars}(B)$$
$$A \rightarrow I$$
$$I \land B \rightarrow \bot$$
Interpolation in Proofs

1. Find a concrete proof for a simple case
2. Generalise the proof
3. Try to prove the general case
Interpolation in Proofs

1. Find a concrete proof for a simple case

2. Generalise the proof

3. Try to prove the general case

\[ (A \land T(x_0, x_1) \land \ldots \land T(x_{k-1}, x_k) \land Err(x_k)) \land (B) \]
Interpolation in Proofs

1. Find a concrete proof for a simple case

2. Generalise the proof

3. Try to prove the general case

\[
\begin{align*}
\begin{array}{c}
S(x_0) \\
\wedge \\
T(x_0, x_1) \\
\wedge \ldots \\
\wedge T(x_{k-1}, x_k) \\
\wedge Err(x_k)
\end{array}
\end{align*}
\]

\[
\begin{align*}
I(x) & \wedge T(x, x') \\
\rightarrow \\
I(x')
\end{align*}
\]
Example: HiFrog

Given a C program and a set of assertions
Example: HiFrog

Given a C program and a set of assertions
1. Construct a BMC instance of the program
Example: HiFrog

Given a C program and a set of assertions

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
Example: HiFrog

Given a C program and a set of assertions

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
Example: HiFrog

Given a C program and a set of assertions

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
4. Use the interpolant for checking the consequent assertions
Example: HiFrog

Given a C program and a set of assertions

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
4. Use the interpolant for checking the consequent assertions
Example: HiFrog

Given a C program and a set of assertions:

1. Construct a BMC instance of the program
2. Check the first assertion against the BMC instance
3. Compute an interpolant out of the proof
4. Use the interpolant for checking the consequent assertions
Duality of Interpolants
Duality of Interpolants

\[ A \quad B \]
Duality of Interpolants
Duality of Interpolants

\[
\neg I'
\]

\[A\]

\[B\]
What is LRA

Given a set of linear inequalities over real-valued variables, determine if there are values for the variables that satisfy all the inequalities

\[ 0 \leq 0.5x + 3y - 2z \]
\[ 5 > y \]
\[ x^2 + 2xy + y^2 > 1 \]

In 2 dimensions: determine whether half planes have a non-empty intersection
Solving LRA in SMT

The theory solver for LRA is based on the Simplex algorithm
Simplex in SMT

A pre-processing step:

- All inequalities are written so that left side is a constant and right side a linear expression.

We end up with two types of entities:

- Bounds on variables
- Bounds on sums of the variables

The idea is to repeatedly adjust variable values to satisfy bounds on the sums, and change the role of the variables and the sums.
Simplex Example

\[ 0 \leq 0.5x + 3y - 2z \]
\[ 0.3 \geq x + 2y \]
\[ 1 < x - 2y + 3z \]
Simplex Example
Simplex Example
Simplex Example
Simplex Example

\[ x \leq \]
\[ y \geq \]
\[ y < \]
Simplex Example
Simplex Example

Just enough to fix the equality
Simplex Example

$$u = 0.5x + 3y - 2z$$

Just enough to fix the equality
Simplex Example

\[ u = 0.5x + 3y - 2z \]
\[ x = 2u - 6y + 4z \]

Just enough to fix the equality
Simplex Example

\[ u = 0.5x + 3y - 2z \]
\[ x = 2u - 6y + 4z \]

Just enough to fix the equality
Simplex Example

After each adjustment

1. The expressions change
2. One variable is fixed to its bound
Simplex Example

After each adjustment

1. The expressions change
2. One variable is fixed to its bound
Simplex Example

After each adjustment

1. The expressions change
2. One variable is fixed to its bound

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable
Simplex Example

After each adjustment
1. The expressions change
2. One variable is fixed to its bound

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

A conflict is the unsatisfied expression and the set of expressions currently bounding its variables.
Simplex Example

After each adjustment
1. The expressions change
2. One variable is fixed to its bound

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

\[ 1 > 0.5v - u \]
Simplex Example

After each adjustment

1. The expressions change
2. One variable is fixed to its bound

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

A conflict is the unsatisfied expression and the set of expressions currently bounding its variables.

\[(1 > 0.5v - u) \land (u > 3)\]
### Simplex Example

After each adjustment

1. The expressions change
2. One variable is fixed to its bound

If an expression bound cannot be satisfied since the variables are at their bounds, the problem is unsatisfiable

\[(1 > 0.5v - u) \land (u > 3) \land (v < 3)\]
LRA Interpolation

Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u, v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and

$(v < 3) \in A, (1 > 0.5v - u) \in A$. 
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$ and the bounds for the variables $u, v$ were

\[ u > 3 \]
\[ v < 3 \]

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$. 

LRA Interpolation
Assume that the expression bound that could not be satisfied was \( 1 > 0.5v - u \) and the bounds for the variables \( u, \ v \) were \( u > 3 \) \( v < 3 \). Assume that \((u > 3) \in B\) and \((v < 3) \in A\), \((1 > 0.5v - u) \in A\).
LRA Interpolation

Assume that the expression bound that could not be satisfied was \( 1 > 0.5v - u \) and the bounds for the variables \( u, v \) were:

- \( u > 3 \)
- \( v < 3 \)

Assume that \((u > 3) \in B\) and \((v < 3) \in A, (1 > 0.5v - u) \in A\).

The interpolant for \( A \) is obtained by summing to the expression the bounds in \( A \) multiplied by their factors in the expression:

\[
(0.5v - u - 1)
\]
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$. and the bounds for the variables $u, v$ were $u > 3$ $v < 3$ Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$. The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$$(0.5v - u - 1) + 0.5(3 - v)$$
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$ and the bounds for the variables $u$, $v$ were $u > 3$, $v < 3$. Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$. The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression: $(0.5v - u - 1) + 0.5(3 - v)$.
Assume that the expression bound that could not be satisfied was \(1 > 0.5v - u\)

and the bounds for the variables \(u, v\) were

\[u > 3\]
\[v < 3\]

Assume that \((u > 3) \in B\) and 
\((v < 3) \in A, (1 > 0.5v - u) \in A\).

The interpolant for \(A\) is obtained by summing to the expression the bounds in \(A\) multiplied by their factors in the expression:

\[-u - 1 + 1.5\]
LRA Interpolation

Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$

and the bounds for the variables $u, v$ were

$u > 3$

$v < 3$

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$0 < -u - 1 + 1.5$
LRA Interpolation

Assume that the expression bound that could not be satisfied was \( 1 > 0.5v - u \) and the bounds for the variables \( u, v \) were

\[ u > 3 \]
\[ v < 3 \]

Assume that \((u > 3) \in B\) and \((v < 3) \in A\), \((1 > 0.5v - u) \in A\).

The interpolant for \( A \) is obtained by summing to the expression the bounds in \( A \) multiplied by their factors in the expression:

\[ u < 0.5 \]
Assume that the expression bound that could not be satisfied was $1 > 0.5v - u$ and the bounds for the variables $u, v$ were $u > 3$ and $v < 3$.

Assume that $(u > 3) \in B$ and $(v < 3) \in A$, $(1 > 0.5v - u) \in A$.

The interpolant for $A$ is obtained by summing to the expression the bounds in $A$ multiplied by their factors in the expression:

$$u < 0.5$$
Duality-based Interpolation for LRA

Given a primal interpolant

\[ I = c_1 \leq t(x), \]

the dual interpolant has the form

\[ I' = c_2 < t(x) \]
Given a primal interpolant

\[ I = c_1 \leq t(x), \]

the dual interpolant has the form

\[ I' = c_2 < t(x) \]
Duality-based Interpolation for LRA

Given a primal interpolant

\[ I = c_1 \leq t(x), \]

the dual interpolant has the form

\[ I' = c_2 < t(x). \]
Duality-based Interpolation for LRA

Given a primal interpolant

\[ I = c_1 \leq t(x), \]

the dual interpolant has the form

\[ I' = c_2 < t(x) \]
Given a primal interpolant

\[ I = c_1 \leq t(x), \]

the dual interpolant has the form

\[ I' = c_2 < t(x) \]

All the inequalities of the form

\[ c < t(x), \quad c_1 \leq c < c_2 \]

are also interpolants for \( A \)
Given a primal interpolant

\[ I = c_1 \leq t(x), \]

the dual interpolant has the form

\[ I' = c_2 < t(x) \]

All the inequalities of the form

\[ c < t(x), \quad c_1 \leq c < c_2 \]

are also interpolants for \( A \)

\[ I \rightarrow I' \]
Duality-based Interpolation for LRA

Given a primal interpolant
\[ I = c_1 \leq t(x), \]
the dual interpolant has the form
\[ I' = c_2 < t(x) \]

All the inequalities of the form
\[ c < t(x), \quad c_1 \leq c < c_2 \]
are also interpolants for \( A \)

\[ I \rightarrow I' \]
Interpolant Duality Visualized
Interpolant Duality Visualized

$c_1 = t(x)$

$c_2 = t(x)$

$A$

$B$
Interpolant Duality Visualized

\[ a_1 = t(x) \]

\[ a \text{ Land} \]

\[ b_2 = t(x) \]

\[ b \text{ Land} \]
Interpolant Duality Visualized

No man’s land

$A$ Land

$B$ Land
Interpolant Duality Visualized

\[ \alpha(c_2 - c_1) + c_1 = t(x) \]

\[ c_1 = t(x) \]

\[ c_2 = t(x) \]

A Land

B Land

No man’s land
Interpolant Duality Visualized

\[ a(c_2 - c_1) + c_1 = t(x) \]

\[ c_1 = t(x) \]

No man’s land

A Land

B Land

I AM GNOME ANN!
Experiments on SV-COMP and HiFrog
The Architecture Overview

OpenSMT
Partitions $A \land B$

SMT Solver
Labelling

Proof analysis
Boolean
LRA

Interpolation Module

Interpolator
Boolean
LRA

Model Checker
SAT/UNSAT Interpolant

PARTITIONS $A$ and $B$

UNSAT proof

Proof analysis
UNSAT proof

statistics

Partitions $A$ and $B$
Implemented in HiFrog

Sources + Assertions → Assertion traversal

Assertion traversal → SMT Encoder → Propositional summaries

SMT Encoder → LRA summaries

Interpolating SMT Solver → Interpolation-based summaries

Interpolating SMT Solver → Theory solvers → proof

Theory solvers → proof

Prop ITP → proof compressor

Prop ITP → LRA ITP → Error trace

Error trace → Summary refiner

Summary refiner → Interpolating SMT Solver

Interpolating SMT Solver → SAT

SAT → Interpolating SMT Solver

Interpolating SMT Solver → Assertion holds
Results on SMT-LIB

Experiments with three LRA labelling functions:

Strong: the primal interpolant
Weak: the dual interpolant
$c = 0.5$: the interpolant between dual and primal
## Experiments on HiFrog

<table>
<thead>
<tr>
<th>ITP</th>
<th>floppy1</th>
<th>kbfilt1</th>
<th>diskperf1</th>
<th>mem</th>
<th>disk</th>
<th>Σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strong</td>
<td>27100</td>
<td>5120</td>
<td>39900</td>
<td>25600</td>
<td>47600</td>
<td>145000</td>
</tr>
<tr>
<td>c = 0.5</td>
<td>25100</td>
<td>5120</td>
<td>39200</td>
<td>25100</td>
<td>41500</td>
<td>136000</td>
</tr>
<tr>
<td>Weak</td>
<td>24800</td>
<td>5380</td>
<td>39200</td>
<td>25600</td>
<td>64000</td>
<td>159000</td>
</tr>
</tbody>
</table>

Number of HiFrog refinements (fixed propositional ITP algorithm)

- The difference between minimum and maximum is ~ 15%
- The $c = 0.5$ ITP provides the best results
Related Work

Nikolaj Bjørner, Arie Gurfinkel:
Property Directed Polyhedral Abstraction. VMCAI 2015

Pudlák:

McMillan:

D’Silva, Kroening, Purandare, and Weissenbacher:
Interpolant Strength. VMCAI 2010.

Albarghouthi, McMillan:
Beautiful interpolants. CAV 2013.

Dutertre, de Moura:

Alt, Hyvärinen, Asadi, and Sharygina:
Conclusions

LRA interpolation with controlled strength
Provides an infinite family of interpolants based on interpolation duality
Integrated into a model checker

Future work

Better heuristics for the labelling function
Apply to fix-point computations in other MC applications

Implementations available at
http://verify.inf.usi.ch/hifrog,
http://verify.inf.usi.ch/opensmt