Extending Proof Tree Preserving Interpolation to Sequences and Trees (Work in Progress)

Jürgen Christ  Jochen Hoenicke

University of Freiburg

July 8, 2013
Extending
Proof Tree Preserving Interpolation
to
Proof Tree Preserving Tree Interpolation
Outline

1 Motivation

2 Preliminaries
   • Interpolation in SAT
   • Interpolation in SMT

3 From Binary to Tree Interpolation

4 Tree Interpolation by Example

5 Conclusion
Uses of Tree Interpolation

- Hoare-style program verification [Henzinger 04]

```
procedure f(n) returns res
if (n <= 0)
    res := 0

assert res >= n
```

\[ n = 0 \]
\[ \downarrow \]
\[ n \leq 0 \]
\[ \downarrow \]
\[ res = 0 \]
\[ \downarrow \]
\[ res < n \]
Hoare-style program verification [Henzinger 04]

```
procedure f(n) returns res
if (n <= 0)
  res := 0
assert res >= n
```
Uses of Tree Interpolation

- Hoare-style program verification [Henzinger 04, Heizmann 10]

Procedure $f(n)$ returns $res$

- if $(n \leq 0)$
  - $res := 0$
- else
  - $res := n + \text{call} \ f(n - 1)$

assert $res \geq n$
Uses of Tree Interpolation

- Hoare-style program verification [Henzinger 04, Heizmann 10]

procedure $f(n)$ returns $\text{res}$
if ($n \leq 0$)
  $\text{res} := 0$
else
  $\text{res} := n + \text{call } f(n - 1)$
assert $\text{res} \geq n$
Uses of Tree Interpolation

- Hoare-style program verification [Henzinger 04, Heizmann 10]
- Verification of multi-threaded programs and higher order programs [Rybalchenko 12]
- Incremental update checking [Sery 11]
- Solving non-recursive Horn clauses [Rybalchenko 11]
- Inductive Dataflow Graphs [Podelski 13]
- ...
Tree Interpolation Problem
Tree Interpolation Problem

$\wedge F_i$ is unsatisfiable
Tree Interpolation Problem

\[ F_0 \wedge F_i \text{ is unsatisfiable} \]
Tree Interpolation Problem

\[ \bigwedge F_i \text{ is unsatisfiable} \]

Tree Inductivity:

\[ I_0 \equiv \bot \]
Tree Interpolation Problem

\[ F_i \text{ is unsatisfiable} \]

**Tree Inductivity:**

- \[ I_0 \equiv \bot \]
- Child interpolants and parent imply parent interpolant
Tree Interpolation Problem

\[ \bigwedge F_i \text{ is unsatisfiable} \]

Tree Inductivity:
- \( I_0 \equiv \perp \)
- Child interpolants and parent imply parent interpolant
- Interpolant only contains symbols occurring inside and outside the current subtree
Outline

1 Motivation

2 Preliminaries
   - Interpolation in SAT
   - Interpolation in SMT

3 From Binary to Tree Interpolation

4 Tree Interpolation by Example

5 Conclusion
Binary Interpolation

For $A \land B \models_T \bot$:

- $A \models_T I$,
- $B \land I \models_T \bot$,
- $symb(I) \subseteq symb(A) \cap symb(B)$
For $A \land B \models_T \bot$:

- $A \models_T I$,
- $B \land I \models_T \bot$,
- $symb(I) \subseteq symb(A) \cap symb(B)$
Outline

1. Motivation

2. Preliminaries
   - Interpolation in SAT
   - Interpolation in SMT

3. From Binary to Tree Interpolation

4. Tree Interpolation by Example

5. Conclusion
Resolution Refutation

Proof consists of

- leaves representing input clauses,
Resolution Refutation

Proof consists of

- leaves representing input clauses,
- inner nodes derived by resolution

\[
\frac{C_1 \lor \ell}{C_1 \lor C_2} \quad \frac{C_2 \lor \neg \ell}{C_1 \lor C_2}
\]

\[
\frac{P \lor Q}{P} \quad \frac{P \lor \neg Q}{-P}
\]
Resolution Refutation

Proof consists of

- leaves representing input clauses,
- inner nodes derived by resolution

\[ \begin{align*}
C_1 \lor \ell & \quad C_2 \lor \neg \ell \\
\hline
C_1 \lor C_2
\end{align*} \]

- the root node representing the empty clause.

\[ \begin{align*}
P \lor Q & \quad P \lor \neg Q \\
\hline
P & \quad \neg P
\end{align*} \]
Labelled Resolution Refutation

Label each clause in the resolution refutation with \textit{partial interpolant}

\[ P \lor Q : I_{P \lor Q} \quad P \lor \neg Q : I_{P \lor \neg Q} \]

\[ P : I_P \quad \neg P : I_{\neg P} \]

\[ \bot : I_\bot \]
Labelled Resolution Refutation

Label each clause in the resolution refutation with \textit{partial interpolant}

\[
P \lor Q : I_{P \lor Q} \quad P \lor \neg Q : I_{P \lor \neg Q}
\]

- \textbf{Syntactic rules for leaves}

\[
P : I_P \\
\neg P : I_{\neg P} \\
\bot : I_{\bot}
\]
Labelled Resolution Refutation

Label each clause in the resolution refutation with *partial interpolant*

\[
\begin{align*}
\ell \in A & \quad C_1 \lor \ell : l_1 \\
& \quad C_2 \lor \neg \ell : l_2 \\
& \quad C_1 \lor C_2 : l_1 \lor l_2 \\
\ell \in B & \quad C_1 \lor \ell : l_1 \\
& \quad C_2 \lor \neg \ell : l_2 \\
& \quad C_1 \lor C_2 : l_1 \land l_2 \\
\end{align*}
\]

- Syntactic rules for leaves
- Interpolant of resolved based on interpolants of antecedents and pivot
Labelled Resolution Refutation

Label each clause in the resolution refutation with partial interpolant

\[
\begin{array}{ll}
\ell \in A & C_1 \lor \ell : l_1 \\
& C_2 \lor \neg \ell : l_2 \\
\hline & C_1 \lor C_2 : l_1 \lor l_2 \\
\ell \in \mathcal{B} & C_1 \lor \ell : l_1 \\
& C_2 \lor \neg \ell : l_2 \\
\hline & C_1 \lor C_2 : l_1 \land l_2 \\
\end{array}
\]

- Syntactic rules for leaves
- Interpolant of resolved based on interpolants of antecedents and pivot

\( I_{\bot} \) is desired interpolant.
Partial interpolant $I_C$ of clause $C$ is interpolant of

$$A \land B \land \neg C$$
Partial Interpolants

Partial interpolant $I_C$ of clause $C$ is interpolant of

$$A \land B \land \neg C$$

How to split $\neg C$?
Partial interpolant $I_C$ of clause $C$ is interpolant of

$$A \land B \land \neg C$$

Define $\neg C \downarrow A$ and $\neg C \downarrow B$ such that

- $symb(\neg C \downarrow A) \subseteq symb(A)$
- $symb(\neg C \downarrow B) \subseteq symb(B)$
- $\neg C \iff \neg C \downarrow A \land \neg C \downarrow B$
Partial Interpolants

Partial interpolant $I_C$ of clause $C$ is interpolant of

$$A \land B \land \neg C$$

Define $\neg C \downarrow A$ and $\neg C \downarrow B$ such that

- $symb(\neg C \downarrow A) \subseteq symb(A)$
- $symb(\neg C \downarrow B) \subseteq symb(B)$
- $\neg C \iff \neg C \downarrow A \land \neg C \downarrow B$

Partial interpolant $I_C$ is interpolant of $A \land ((\neg C) \downarrow A)$ and $B \land ((\neg C) \downarrow B)$. 
1 Motivation

2 Preliminaries
   • Interpolation in SAT
   • Interpolation in SMT

3 From Binary to Tree Interpolation

4 Tree Interpolation by Example

5 Conclusion
Additional Leaves

- Theory lemmas
- Theory combination lemmas

\[ x \leq y \lor x \neq y \]
\[ x \geq y \lor x \neq y \]
\[ x < y \lor x > y \lor x = y \]
Additional Leaves

- Theory lemmas
- Theory combination lemmas

\[
\begin{align*}
x & \leq y \lor x \neq y \\
x & \geq y \lor x \neq y \\
x & < y \lor x > y \lor x = y
\end{align*}
\]

might contain literals that are not in the input formulas
Mixed Literals

- literals that contain symbols only in $A$ and symbols only in $B$: $a = b$
Mixed Literals

- Literals that contain symbols only in $A$ and symbols only in $B$: $a = b$
- Literals do not occur in input formulas
Mixed Literals

- literals that contain symbols only in $A$ and symbols only in $B$: $a = b$
- literals do not occur in input formulas
- created by
  - theory combination (Nelson-Oppen, Ackermannization),
  - cuts and extended branches used to solve integer arithmetic,
  - ...
Mixed Literals

- literals that contain symbols only in $A$ and symbols only in $B$: $a = b$
- literals do not occur in input formulas
- created by
  - theory combination (Nelson-Oppen, Ackermannization),
  - cuts and extended branches used to solve integer arithmetic,
  - …

What is $a = b \downarrow A$ and $a = b \downarrow B$?
Purification:
replace $a \leq b$ by $a \leq x \land x \leq b$
similar to purification in Nelson-Oppen
Interpolation and Mixed Literals

Purification:
replace \( a \leq b \) by \( a \leq x \land x \leq b \)
similar to purification in Nelson-Oppen

Interpolation:
Remove purification variable on resolution:

\[
\begin{align*}
C_1 \lor a \leq b : l_1(x_1) & \quad C_2 \lor \neg(a \leq b) : l_2(x_2) \\
C_1 \lor C_2 : l_3
\end{align*}
\]
Purification:
replace $a \leq b$ by $a \leq x \land x \leq b$
similar to purification in Nelson-Oppen

Interpolation:
Remove purification variable on resolution:
\[
\begin{align*}
C_1 \lor a \leq b : I_1(x_1) & \quad C_2 \lor \neg(a \leq b) : I_2(x_2) \\
C_1 \lor C_2 : l_3
\end{align*}
\]

Rules for uninterpreted functions and linear arithmetic [TACAS 2013]
Outline

1 Motivation

2 Preliminaries
   • Interpolation in SAT
   • Interpolation in SMT

3 From Binary to Tree Interpolation

4 Tree Interpolation by Example

5 Conclusion
Binary Interpolation:

\[
\begin{align*}
C_3 &: l_3 & C_4 &: l_4 \\
C_1 &: l_1 & C_2 &: l_2 \\
\bot &: l_{\bot}
\end{align*}
\]
Idea

Binary Interpolation:

\[ C_3 : I_3 \quad C_4 : I_4 \]

\[ C_1 : I_1 \quad C_2 : I_2 \]

\[ I_1 \lor I_2 \]

\[ I_1 \land I_2 \]

\[ I_1 \quad I_2 \]

\[ I_{\bot} \]

\[ I_{\bot} \]

\[ I_1 \land I_2 \]

\[ I_1 \lor I_2 \]

\[ I_1 \land I_2 \]

\[ I_1 \lor I_2 \]
Idea

Tree Interpolation:

\[ C_3 : I_3 \quad C_4 : I_4 \]

\[ C_1 : I_1 \quad C_2 : I_2 \]

\[ \bot : I_{\bot} \]
Tree Interpolation:

\[
\begin{align*}
I_3 & : C_4 : I_4 \\
C_2 & : I_2 \\
I_1 & : I_2 \lor I_1 \land I_2 \\
I_1 & : I_2 \lor I_1 \\
I_2 & : I_1 \land I_2 \\
I_1 & : C_2 : I_2 \\
I_2 & : C_3 : I_3 \\
I_3 & : C_1 : I_1
\end{align*}
\]
Idea

Tree Interpolation:

Repeated binary interpolation
Partial tree interpolant $I_C$ for clause $C$ is tree interpolant of

\[ F_0 \]
\[ \uparrow \]
\[ F_1 \]
\[ F_2 \]
\[ F_3 \]
\[ \land \neg C \]

How to split $\neg C$?
Partial tree interpolant $I_C$ for clause $C$ is tree interpolant of

\[ F_0 \land ((\neg C) \downarrow v_0) \]

\[ F_1 \land ((\neg C) \downarrow v_1) \]

\[ F_2 \land ((\neg C) \downarrow v_2) \quad F_3 \land ((\neg C) \downarrow v_3) \]

- One purification function per node
- $\ell \leftrightarrow \exists \overline{x}. \land_v \ell \downarrow v$
Projection of Mixed Literals

- one auxiliary variable for every node in which literal is mixed
- projection of $a = b$:
Projection of Mixed Literals

- one auxiliary variable for every node in which literal is mixed
- projection of $a = b$:
Projection of Mixed Literals

- one auxiliary variable for every node in which literal is mixed
- projection of $a = b$:
Projection of Mixed Literals

- one auxiliary variable for every node in which literal is mixed
- projection of $a = b$:
1. Motivation

2. Preliminaries
   - Interpolation in SAT
   - Interpolation in SMT

3. From Binary to Tree Interpolation

4. Tree Interpolation by Example

5. Conclusion
$\{q, r\}$
$q \neq r$

$\{c, d\}$
$c = d$

$\{b, d, r, f(\cdot)\}$
$d = b \land f(b) = r$

$\{a, c, q, f(\cdot)\}$
$a = c \land q = f(a)$

\[
a = b \lor a \neq c \lor c \neq d \lor d \neq b \quad a \neq b \lor q \neq f(a) \lor f(b) \neq r \lor q = r
\]

\[
a \neq c \lor c \neq d \lor d \neq b \lor q \neq f(a) \lor f(b) \neq r \lor q = r
\]
Interpolation Problem and Proof Excerpt

\[ \{q, r\} \]

\[ q \neq r \]

\[ \{c, d\} \]

\[ c = d \]

\[ \{a, c, q, f(\cdot)\} \]

\[ a = c \land q = f(a) \]

\[ a = b \lor a \neq c \lor c \neq d \lor d \neq b \]

\[ a \neq b \lor q \neq f(a) \lor f(b) \neq r \lor q = r \]

\[ a \neq c \lor c \neq d \lor d \neq b \lor q \neq f(a) \lor f(b) \neq r \lor q = r \]
Projection: \(a = b \land q = f(a) \land q \neq r \land f(b) = r\)

\[
\begin{align*}
\{q, r\} \\
\{c, d\} & \quad \{b, d, r, f(\cdot)\} \\
\{a, c, q, f(\cdot)\}
\end{align*}
\]
Projection: $a = b \land q = f(a) \land q \neq r \land f(b) = r$
Projection: \( a = b \land q = f(a) \land q \neq r \land f(b) = r \)

\[
\begin{align*}
\{q, r\} \\
q \neq r \land x_2 = x_3
\end{align*}
\]

\[
\{c, d\} \\
x_1 = x_2
\]

\[
\{a, c, q, f(\cdot)\} \\
q = f(a) \land a = x_1
\]

\[
\{b, d, r, f(\cdot)\} \\
f(b) = r \land x_3 = b
\]
Interpolation: \( a = b \land q = f(a) \land f(b) = r \land q \neq r \)

\[
q \neq r \land x_2 = x_3 \\
x_1 = x_2 \land f(b) = r \land x_3 = b \\
q = f(a) \land a = x_1
\]
Interpolation: \( a = b \land q = f(a) \land f(b) = r \land q \neq r \)

\[
q \neq r \land x_2 = x_3 \\
x_1 = x_2 \quad f(b) = r \land x_3 = b \\
q = f(a) \land a = x_1
\]

\[
q \quad f(a) \quad \quad \quad f(b) \quad r
\]

\[
a \quad x_1 \quad x_2 \quad x_3 \quad b
\]
Interpolation: \( a = b \land q = f(a) \land f(b) = r \land q \neq r \)

\[
\begin{align*}
q \neq r \land x_2 &= x_3 \\
\quad x_1 &= x_2 \quad f(b) = r \land x_3 = b \\
q &= f(a) \land a = x_1
\end{align*}
\]

\[
\begin{array}{c}
q \rightarrow f(a) \\
\downarrow \\
a \quad x_1 \quad x_2 \quad x_3 \quad b \\
\end{array}
\]

\[
\begin{array}{c}
f(b) \rightarrow r \\
\downarrow \\
f(x_3) = r \\
\quad \quad q = f(x_1)
\end{array}
\]
Interpolation: \( a = b \land q = f(a) \land f(b) = r \land q \neq r \)

\[
q \neq r \land x_2 = x_3 \\
x_1 = x_2 \quad f(b) = r \land x_3 = b \\
q = f(a) \land a = x_1
\]

\[
q = f(x_2) \quad f(x_3) = r \\
q = f(x_1)
\]
Projection: $a = c \land c = d \land d = b \land a \neq b$

\[
\begin{align*}
\{q, r\} &
\end{align*}
\]
\[
\begin{align*}
\{c, d\} &
\end{align*}
\]
\[
\begin{align*}
\{b, d, r, f(\cdot)\} &
\end{align*}
\]
\[
\begin{align*}
\{a, c, q, f(\cdot)\} &
\end{align*}
\]
Projection: \( a = c \land c = d \land d = b \land a \neq b \)

\[
\begin{align*}
\{a, c, q, f(\cdot)\} & \downarrow \downarrow \downarrow \downarrow \\
a = c & \downarrow \\
\{a, c, q, f(\cdot)\} & \downarrow \downarrow \\
c = d & \downarrow \\
\{c, d\} & \downarrow \\
\{q, r\} & \\
\{b, d, r, f(\cdot)\} & \downarrow \\
d = b & \\
\end{align*}
\]
Projection: \( a = c \land c = d \land d = b \land a \neq b \)

\[
\begin{align*}
\{q, r\} \\
X_2 \cap X_3 &= \emptyset \\
\{c, d\} \\
c &= d \land X_1 \subseteq X_2 \\
\{a, c, q, f(\cdot)\} \\
a = c \land a \in X_1 \\
\{b, d, r, f(\cdot)\} \\
d &= b \land b \in X_3
\end{align*}
\]

- \( X_1, X_2, X_3 \) set-valued
- \( X_i \) separates \( a \) and \( b \)
- No reasoning about sets required in the solver
Interpolation: $a = c \land c = d \land d = b \land a \neq b$

$X_2 \cap X_3 = \emptyset$

c = d \land X_1 \subseteq X_2 \quad d = b \land b \in X_3$

$a = c \land a \in X_1$

$\perp$
Interpolation: \( a = c \land c = d \land d = b \land a \neq b \)

\[
X_2 \cap X_3 = \emptyset
\]

\[
c = d \land X_1 \subseteq X_2 \quad d = b \land b \in X_3
\]

\[
a = c \land a \in X_1
\]

\[
a \ |
\]

\[
X_1 \not\subseteq X_2 \not\subseteq X_3
\]

\[
a \leftarrow c \leftarrow d \leftarrow b
\]
Interpolation: $a = c \land c = d \land d = b \land a \neq b$
Interpolation: \( a = c \land c = d \land d = b \land a \neq b \)

\[ X_2 \cap X_3 = \emptyset \]

\[ c = d \land X_1 \subseteq X_2 \quad d = b \land b \in X_3 \]

\[ a = c \land a \in X_1 \]

\[ X_1 \dashv X_2 \dashv X_3 \]

\[ a \quad c \quad d \quad b \]

\[ d \in X_2 \]

\[ d \in X_3 \]

\[ c \in X_1 \]
partial interpolant for $C_1 \lor a = b$ has form $I_1[s \in X]$
   “If $s \in X$ holds, then $s = a$ resp. $s = b$ (whichever is in the subtree)”

partial interpolant for $C_2 \lor a \neq b$ has form $I_2(x)$
   “$I_2(x)$ holds for $a$ resp. $b$ (whichever is in the subtree)”
partial interpolant for $C_1 \lor a = b$ has form $l_1[s \in X]$
"If $s \in X$ holds, then $s = a$ resp. $s = b$ (whichever is in the subtree)"

partial interpolant for $C_2 \lor a \neq b$ has form $l_2(x)$
"$l_2(x)$ holds for $a$ resp. $b$ (whichever is in the subtree)"

partial interpolant for the resolvent $C_1 \lor C_2$

$$l_1[l_2(s)]$$
Interpolating the Resolution Step

\[ C_1 \lor a = b : d \in X_2 \land d \in X_3 \]

\[ C_2 \lor a \neq b : q = f(x_2) \land f(x_3) = r \]

\[ c \in X_1 \]

\[ q = f(x_1) \]

\[ C_1 \lor C_2 : \]
Interpolating the Resolution Step

\( C_1 \lor a = b : d \in X_2 \quad d \in X_3 \)

\( C_2 \lor a \neq b : q = f(x_2) \quad f(x_3) = r \)

\( c \in X_1 \)

\( q = f(x_1) \)

\( C_1 \lor C_2 : q = f(d) \quad f(d) = r \)

\( q = f(c) \)
Outline

1 Motivation

2 Preliminaries
   - Interpolation in SAT
   - Interpolation in SMT

3 From Binary to Tree Interpolation

4 Tree Interpolation by Example

5 Conclusion
We extended our interpolation scheme to sequence and tree interpolation.

Tree interpolation is repeated binary interpolation.

Scheme computes quantifier-free interpolants in the combination of UF and LA, in particular in QF_UFLIA.

No need to manipulate resolution proof.

Independent of the solver or proof search.

Correctness proofs still work in progress.
Conclusion

- We extended our interpolation scheme to sequence and tree interpolation.
- Tree interpolation is repeated binary interpolation.
- Scheme computes quantifier-free interpolants in the combination of UF and LA, in particular in QF_UFLIA.
- No need to manipulate resolution proof.
- Independent of the solver or proof search.
- Correctness proofs still work in progress.
- Scheme is implemented in SMTInterpol.

http://ultimate.informatik.uni-freiburg.de/smtinterpol

Thanks for your attention 😊