Tree Interpolation in Vampire

Régis Blanc (EPFL)
Ashutosh Gupta (IST Austria)
Laura Kovács (Chalmers)
Bernhard Kragl (TU Vienna)

Interpolation

Craig/Binary Interpolant

$$A \quad \wedge \quad B \quad \rightarrow \quad \bot$$

$$I$$

Interpolation

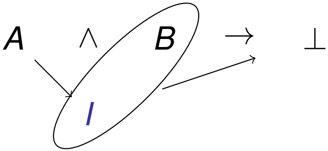
Craig/Binary Interpolant

$$A \quad \wedge \quad B \quad \rightarrow \quad \bot$$

$$A \searrow I$$

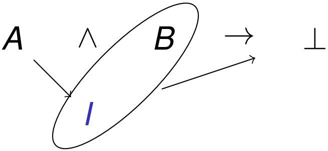
Interpolation

Craig/Binary Interpolant



Interpolation

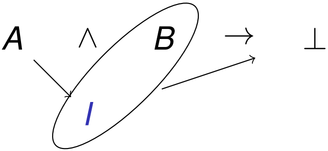
Craig/Binary Interpolant



$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$

Interpolation

Craig/Binary Interpolant



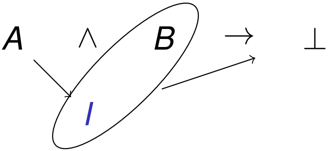
$$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$$

Sequence Interpolant

$$A_1 \quad \wedge \quad A_2 \quad \wedge \quad A_3 \quad \wedge \quad \cdots \quad \wedge \quad A_n \quad \rightarrow \quad \bot$$

Interpolation

Craig/Binary Interpolant



$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$

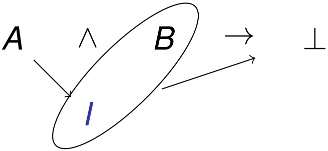
Sequence Interpolant

$$A_1 \quad \wedge \quad A_2 \quad \wedge \quad A_3 \quad \wedge \quad \cdots \quad \wedge \quad A_n \quad \rightarrow \quad \bot$$

$$I_1 \qquad I_2 \qquad \cdots \qquad I_{n-1}$$

Interpolation

Craig/Binary Interpolant



$$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$$

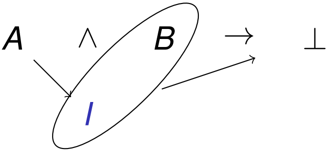
Sequence Interpolant

$$A_1 \quad \wedge \quad A_2 \quad \wedge \quad A_3 \quad \wedge \quad \cdots \quad \wedge \quad A_n \quad \rightarrow \quad \bot$$

$$I_1 \qquad I_2 \qquad \cdots \qquad I_{n-1}$$

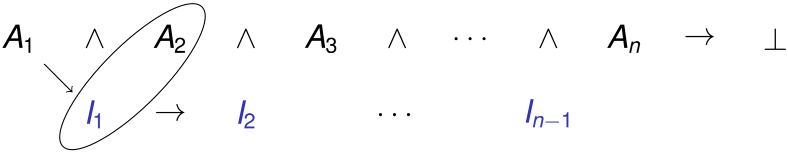
Interpolation

Craig/Binary Interpolant



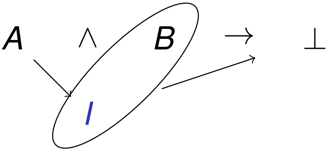
$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$

Sequence Interpolant



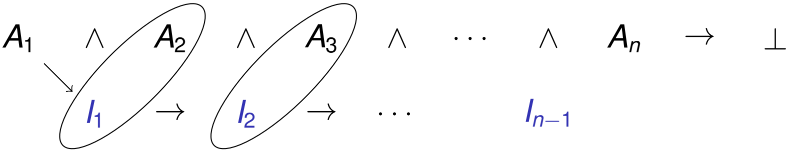
Interpolation

Craig/Binary Interpolant



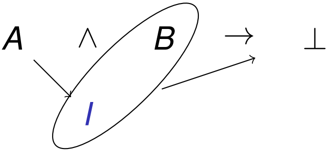
$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$

Sequence Interpolant



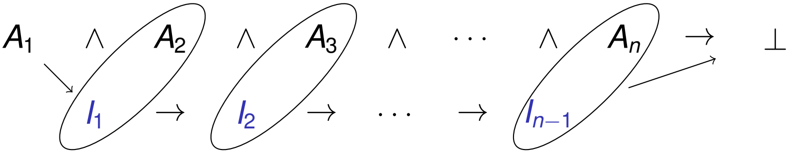
Interpolation

Craig/Binary Interpolant



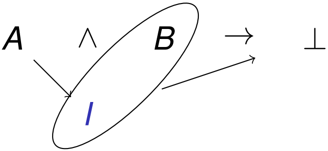
$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$

Sequence Interpolant



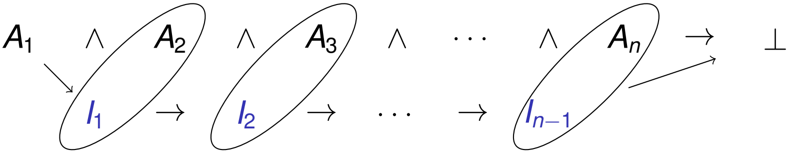
Interpolation

Craig/Binary Interpolant



$I \in \mathcal{L}(A) \cap \mathcal{L}(B)$

Sequence Interpolant



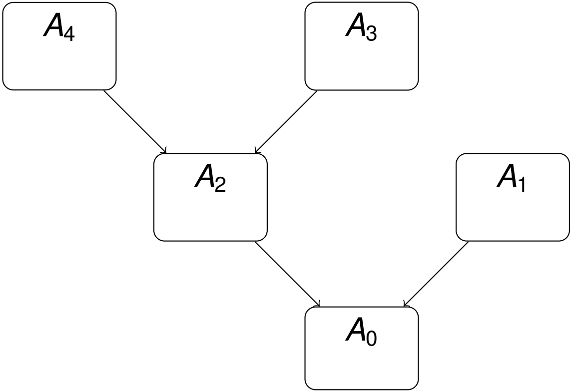
$I_1 \in \mathcal{L}(A_1) \cap \mathcal{L}(A_2, \ldots, A_n)$

$I_2 \in \mathcal{L}(A_1, A_2) \cap \mathcal{L}(A_3, \ldots, A_n)$

$I_{n-1} \in \mathcal{L}(A_1, \ldots, A_{n-1}) \cap \mathcal{L}(A_n)$

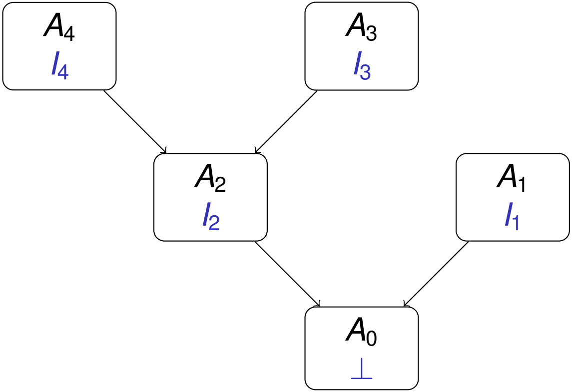
Tree interpolation

$$A_0 \wedge A_1 \wedge A_2 \wedge A_3 \wedge A_4 \rightarrow \bot$$



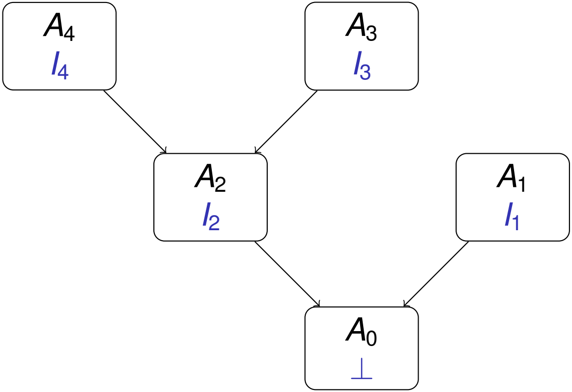
Tree interpolation

$$A_0 \wedge A_1 \wedge A_2 \wedge A_3 \wedge A_4 \rightarrow \bot$$



Tree interpolation

$$A_0 \wedge A_1 \wedge A_2 \wedge A_3 \wedge A_4 \rightarrow \bot$$

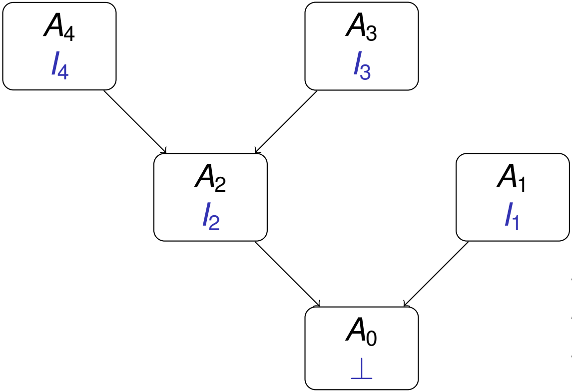


node $\wedge$ child Itps $\rightarrow$ Itp

$$A_4 \rightarrow I_4$$
$$A_3 \rightarrow I_3$$
$$A_1 \rightarrow I_1$$
$$A_2 \wedge I_4 \wedge I_3 \rightarrow I_2$$
$$A_0 \wedge I_2 \wedge I_1 \rightarrow \bot$$

Tree interpolation

$$A_0 \wedge A_1 \wedge A_2 \wedge A_3 \wedge A_4 \rightarrow \bot$$



node $\wedge$ child Itps $\rightarrow$ Itp

$$A_4 \rightarrow I_4$$
$$A_3 \rightarrow I_3$$
$$A_1 \rightarrow I_1$$
$$A_2 \wedge I_4 \wedge I_3 \rightarrow I_2$$
$$A_0 \wedge I_2 \wedge I_1 \rightarrow \bot$$

Language restrictions

$$I_4 \in \mathcal{L}(A_4) \cap \mathcal{L}(A_3, A_2, A_1, A_0)$$
$$I_3 \in \mathcal{L}(A_3) \cap \mathcal{L}(A_4, A_2, A_1, A_0)$$
$$I_2 \in \mathcal{L}(A_4, A_3, A_2) \cap \mathcal{L}(A_1, A_0)$$
$$I_1 \in \mathcal{L}(A_1) \cap \mathcal{L}(A_3, A_2, A_1, A_0)$$

Related Work

Solving recursion-free Horn clauses
[Gupta, Popeea, Rybalchenko POPL'11]

Interpolants for procedure summarization
[McMillan, Rybalchenko MSR-TR'13]

Generalized property directed reachability
[Hoder, Bjørner SAT'12]

Interpolation and Horn Clauses
[Hojjat, Rümmer, Kuncak CAV'13]

Nested Interpolants
[Heizmann, Hoenicke, Podelski POPL'10]

and many more ...

Important questions

- Do interpolants always exist?
  Yes, in first-order logic (also with respect to a theory)
- Is a logic closed under interpolation? (e.g. quantifier free fragments)
  Not necessarily, consider $a = 2b + 1 \wedge a = 2c$ over $\mathbb{Z}$

Important questions

- Do interpolants always exist?
  Yes, in first-order logic (also with respect to a theory)
- Is a logic closed under interpolation? (e.g. quantifier free fragments)
  Not necessarily, consider $a = 2b + 1 \wedge a = 2c$ over $\mathbb{Z}$
- How to interpolate efficiently?
- How to obtain "good" interpolants?

Proof-based interpolation

- Refutations (should) capture the cause of unsatisfiability
- Extensive literature on interpolant extraction for various theories
  - ☞ Notion of *local proof*

Proof-based interpolation

- Refutations (should) capture the cause of unsatisfiability
- Extensive literature on interpolant extraction for various theories
  - ☞ Notion of *local proof*

Consider $a = b \wedge b = c$ and $c = d \wedge a \neq d$

Proof-based interpolation

- Refutations (should) capture the cause of unsatisfiability
- Extensive literature on interpolant extraction for various theories
  - ☞ Notion of *local proof*

Consider $a = b \wedge b = c$ and $c = d \wedge a \neq d$

$$\dfrac{\dfrac{a = b \qquad \dfrac{b = c \qquad c = d}{b = d}}{a = b} \qquad a \neq d}{\bot}$$

Proof-based interpolation

- Refutations (should) capture the cause of unsatisfiability
- Extensive literature on interpolant extraction for various theories
  - ☞ Notion of *local proof*

Consider $a = b \wedge b = c$ and $c = d \wedge a \neq d$

$$\frac{\dfrac{a = b \qquad \dfrac{b = c \qquad c = d}{b = d}}{a = b} \qquad a \neq d}{\bot} \qquad\qquad \frac{\dfrac{\dfrac{a = b \qquad b = c}{a = c} \qquad c = d}{a = d} \qquad a \neq d}{\bot}$$

Proof-based interpolation

- Refutations (should) capture the cause of unsatisfiability
- Extensive literature on interpolant extraction for various theories
  - ☞ Notion of *local proof*

Consider $a = b \wedge b = c$ and $c = d \wedge a \neq d$

$$\frac{\dfrac{a = b \qquad \dfrac{b = c \qquad c = d}{b = d}}{a = b} \qquad a \neq d}{\bot}$$

$$\frac{\dfrac{\dfrac{a = b \qquad b = c}{a = c} \qquad c = d}{a = d} \qquad a \neq d}{\bot}$$

interpolant: $a = c$

Vampire

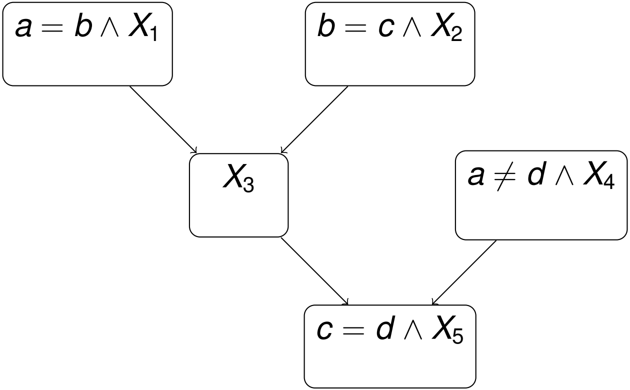
- Vampire is one of the best first-order theorem provers

- Recent developments/extensions:
  - Invariant generation [FASE'09, MICAI'11]
  - Interpolation and Symbol Elimination [CADE'09, IJCAR'10]
  - Interpolant minimization &
    Theory independent proof localization [POPL'12]
  - Incremental tree interpolation [today]

Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

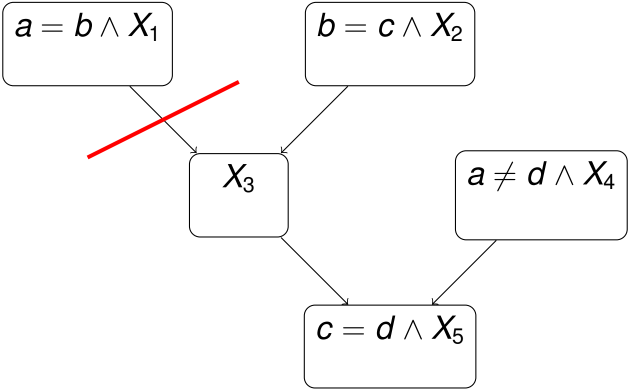
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

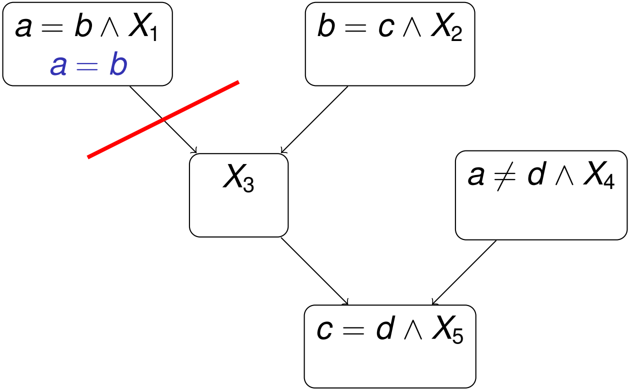
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

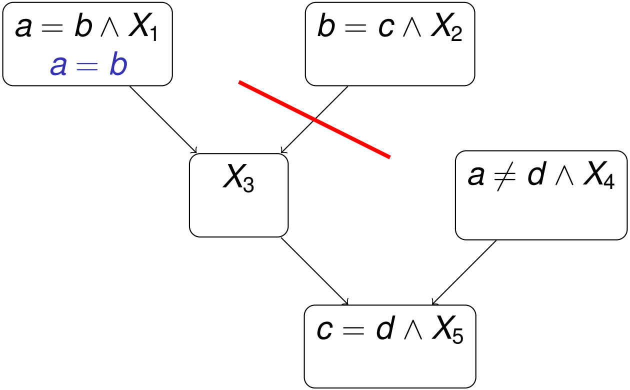
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

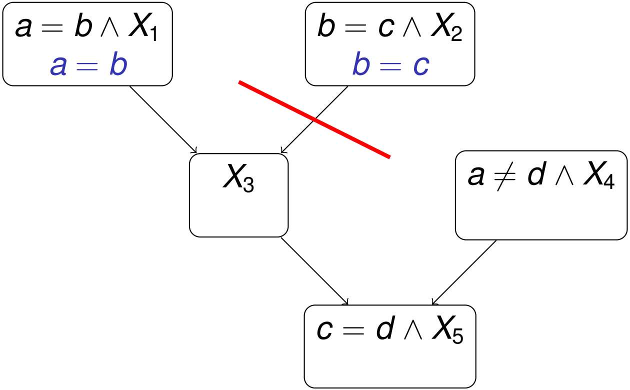
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

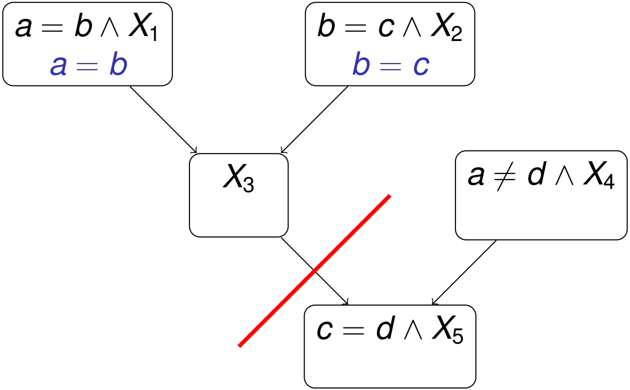
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

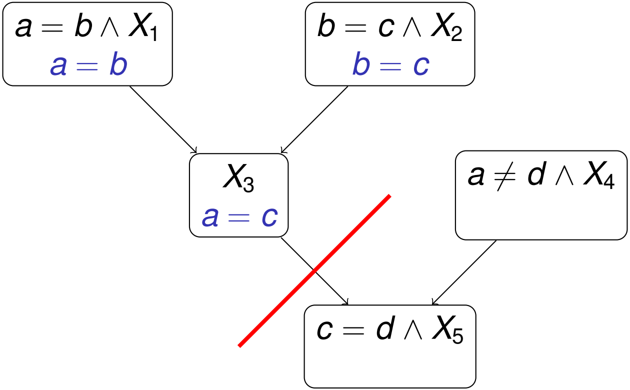
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

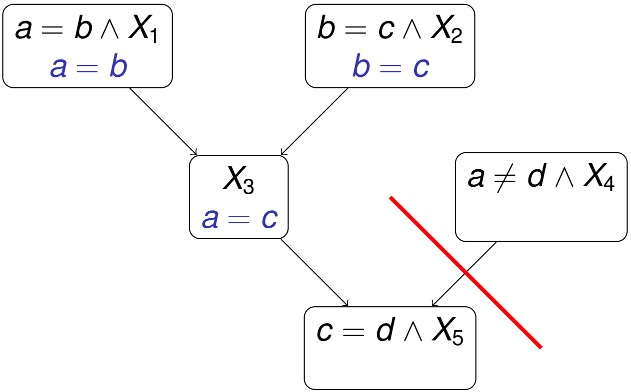
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

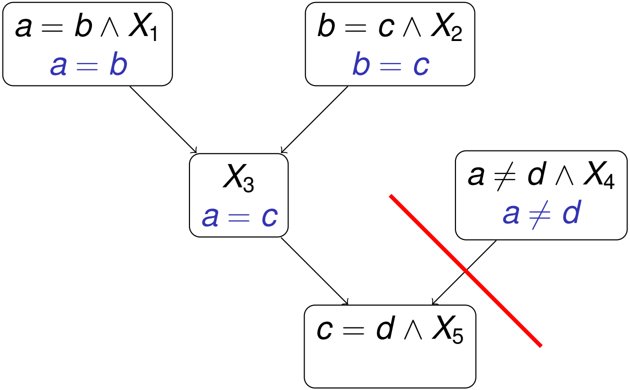
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

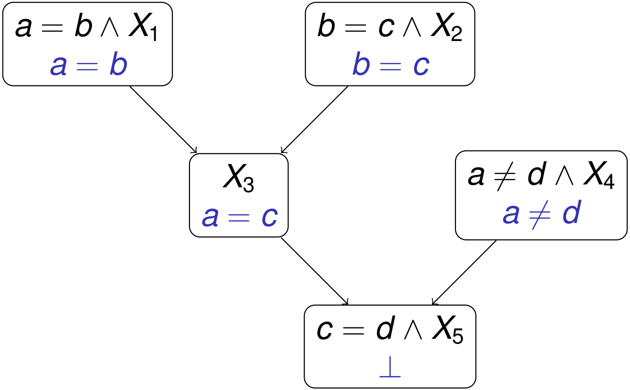
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



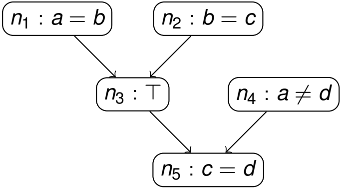
Incremental tree interpolation

- Visit tree nodes in *topological order*
- Per node: *partition* the tree and compute *binary interpolant*
- Crucial: reuse previously computed interpolants

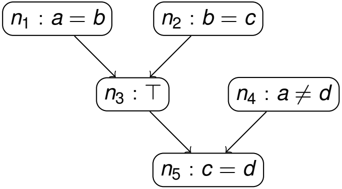
$\mathcal{L}(X_1, X_2, X_3, X_4, X_5) \cap \{a, b, c, d\} = \emptyset$



Tool usage



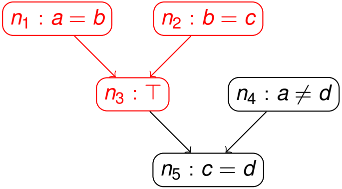
Tool usage



Tree interpolation problem in SMT-LIB 1.2 syntax using iZ3 convention

```
   ...
:assumption (implies (and
    (= a b)
  ) n1 )
:assumption (implies (and
    (= b c)
  ) n2 )
:assumption (implies (and
    n1
    n2
    true
  ) n3 )
:assumption (implies (and
    (not (= a d))
  ) n4 )
:formula (implies (and
    n3
    n4
    (= c d)
  ) false )
```

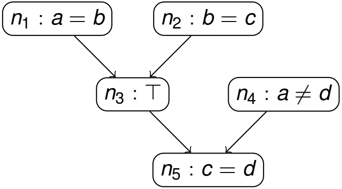
Tool usage



Tree interpolation problem in SMT-LIB 1.2 syntax using iZ3 convention

```
   ...
:assumption (implies (and
    (= a b)
  ) n1 )
:assumption (implies (and
    (= b c)
  ) n2 )
:assumption (implies (and
    n1
    n2
    true
  ) n3 )
:assumption (implies (and
    (not (= a d))
  ) n4 )
:formula (implies (and
    n3
    n4
    (= c d)
  ) false )
```

Tool usage




```
> vampire --show_interpolant tree x.smt
Parsing SMTLIB file: x.smt
Parsing terminated.
Building Tree.
Building Tree terminated.

n1: (= a b)
n2: (= b c)
n3: (= a c)
n4: (not (= a d))
```

Tree interpolation problem in SMT-LIB 1.2 syntax using iZ3 convention

```
   ...
:assumption (implies (and
    (= a b)
  ) n1 )
:assumption (implies (and
    (= b c)
  ) n2 )
:assumption (implies (and
    n1
    n2
    true
  ) n3 )
:assumption (implies (and
    (not (= a d))
  ) n4 )
:formula (implies (and
    n3
    n4
    (= c d)
  ) false )
```

Evaluation

Quantifier-free benchmarks

175 QF_AUFLIA problems from model checking Windows device drivers (90 nodes on average)

Quantified benchmarks

4 small AUFLIA problems

Evaluation

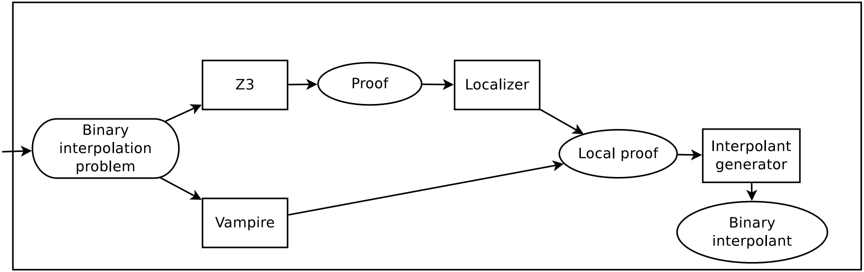
Quantifier-free benchmarks

175 QF_AUFLIA problems from model checking Windows device drivers (90 nodes on average)

Vampire/Vampire	101	141
Vampire/Z3	113	
iZ3	175	

Quantified benchmarks

4 small AUFLIA problems



Evaluation

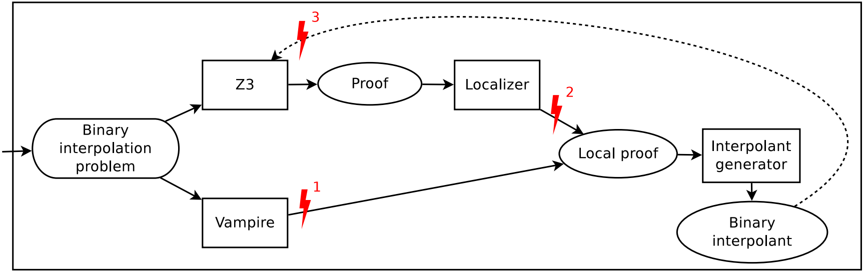
Quantifier-free benchmarks

175 QF_AUFLIA problems from model checking Windows device drivers (90 nodes on average)

<table>
<tr><td>Vampire/Vampire</td><td>101</td><td rowspan="2">141</td></tr>
<tr><td>Vampire/Z3</td><td>113</td></tr>
<tr><td>iZ3</td><td>175</td><td></td></tr>
</table>

Quantified benchmarks

4 small AUFLIA problems



Evaluation

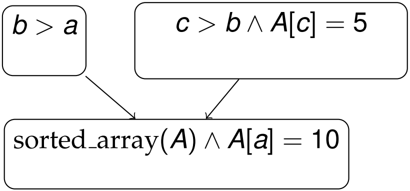
Quantifier-free benchmarks

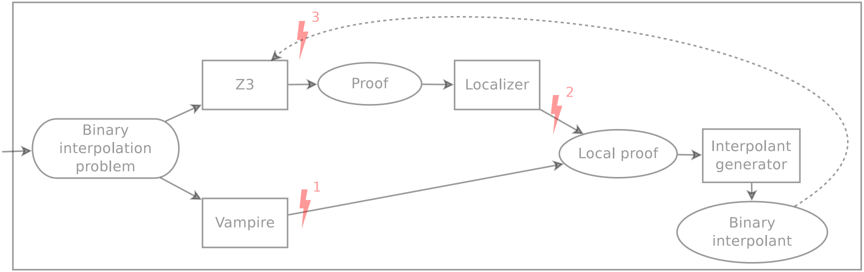
175 QF_AUFLIA problems from model checking Windows device drivers (90 nodes on average)

Vampire/Vampire	101	141
Vampire/Z3	113	
iZ3	175	

Quantified benchmarks

4 small AUFLIA problems





Evaluation

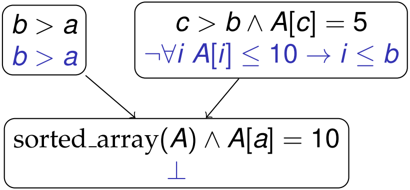
Quantifier-free benchmarks

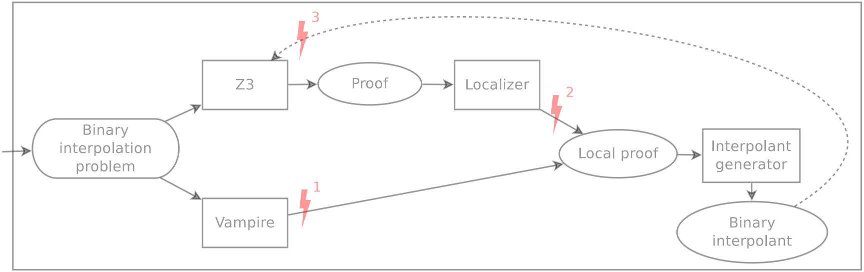
175 QF_AUFLIA problems from model checking Windows device drivers (90 nodes on average)

Vampire/Vampire	101	141
Vampire/Z3	113	
iZ3	175	

Quantified benchmarks

4 small AUFLIA problems





Evaluation

Quantifier-free benchmarks

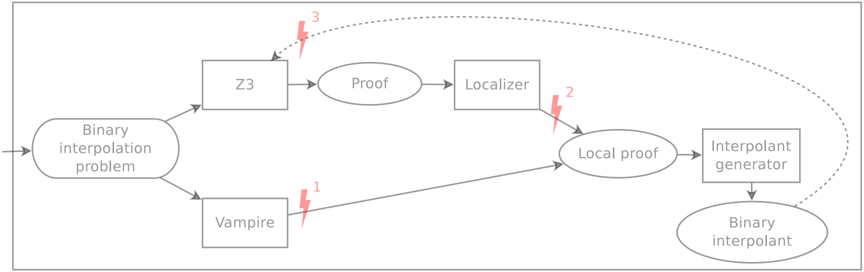
175 QF_AUFLIA problems from model checking Windows device drivers (90 nodes on average)

Vampire/Vampire	101	141
Vampire/Z3	113	
iZ3	175	

Quantified benchmarks

4 small AUFLIA problems

Vampire/Vampire	4
iZ3	1



Conclusion

- Tree interpolation in Vampire
- Strength: reasoning with quantifiers
- Challenges: Theory specific reasoning
- Visit, try, utilize!
  `http://vprover.org/tree_itp`

Conclusion

- Tree interpolation in Vampire
- Strength: reasoning with quantifiers
- Challenges: Theory specific reasoning
- Visit, try, utilize!
  http://vprover.org/tree_itp

Thank you!