First-Order Interpolation

Laura Kovács

TU WIEN FAKULTÄT FÜR !NFORMATIK Faculty of Informatics

CHALMERS

- Interpolation: Craig Interpolation
- Use of interpolation in software verification thanks to K. McMillan

- Interpolation: Craig Interpolation
- Use of interpolation in software verification thanks to K. McMillan

Interpolation in Software Verification

$$
\begin{array}{l}
\textbf{while}\ (c < N)\ \textbf{do} \\
\quad C[c] \;:=\; D[d]; \\
\\
\quad c \;:=\; c+1; \\
\quad d \;:=\; d+1 \\
\textbf{od}
\end{array}
$$

Interpolation in Software Verification

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k) \ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c, d)$

while $(c < N)$ **do**

$C[c] \ := \ D[d];$

$c \ := \ c + 1;$
$d \ := \ d + 1$
od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c, d)$

Interpolation in Software Verification

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c, d)$

while $(c < N)$ **do**

$C[c] \ := \ D[d];$

Loop Invariant?

$c \ := \ c + 1;$
$d \ := \ d + 1$
od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c, d)$

Interpolation in Software Verification

Reachability of B in ONE iteration: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow B(c',d')$

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c,d)$

while $(c < N)$ **do**

$C[c] \ := \ D[d];$

$c \ := \ c + 1;$
$d \ := \ d + 1$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c',d')$

Interpolation in Software Verification

Reachability of B in ONE iteration: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow B(c',d')$

$\{c = d = 0 \;\wedge N > 0 \;\wedge\; (\forall k)\;(0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c,d)$

while $(c < N)$ **do**

$C[c] \;:=\; D[d];$ $\qquad \underbrace{c < N \wedge C[c] = D[d] \wedge c' = c+1 \wedge d' = d+1}_{T(c,d,c',d')}$

$c \;:=\; c+1;$

$d \;:=\; d+1$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c',d')$

Interpolation in Software Verification

Reachability of B in ONE iteration: $R(c,d) \land T(c,d,c',d') \land c' \geq N \rightarrow B(c',d')$

$\{c = d = 0 \ \land N > 0 \ \land \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c,d)$

while $(c < N)$ **do**

$C[c] \ := \ D[d];$

$c \ := \ c + 1;$
$d \ := \ d + 1$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c',d')$

Refutation: $R(c,d) \land T(c,d,c',d') \land c' \geq N \land \neg B(c',d')$

- The formula is of 2 states (c, d, c', d').
- Need a state formula $I(c',d')$ such that: (Jhala and McMillan)

$R(c,d) \land T(c,d,c',d') \land c' \geq N \rightarrow I(c',d')$ and $I(c',d') \land \neg B(c',d') \rightarrow \bot$

Interpolation in Software Verification

Reachability of B in ONE iteration: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow B(c',d')$

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c,d)$

while $(c < N)$ **do**

$C[c] \ := \ D[d];$

$c \ := \ c + 1;$
$d \ := \ d + 1$
od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c',d')$

Refutation: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \wedge \neg B(c',d')$

- The formula is of 2 states (c, d, c', d').
- Need a state formula $I(c',d')$ such that: (Jhala and McMillan)

$R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow I(c',d')$ and $I(c',d') \wedge \neg B(c',d') \rightarrow \perp$

Interpolation in Software Verification

Reachability of B in ONE iteration: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow B(c',d')$

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c,d)$

while $(c < N)$ **do**

$C[c] \ := \ D[d];$

$c \ := \ c + 1;$
$d \ := \ d + 1$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c',d')$

Refutation: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \wedge \neg B(c',d')$

- The formula is of 2 states (c, d, c', d').
- Need a state formula $I(c',d')$ such that: (Jhala and McMillan)

$R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow I(c',d')$ and $I(c',d') \wedge \neg B(c',d') \rightarrow \perp$

Task: Compute interpolant $I(c',d')$ from refutation by eliminating symbols c, d.

Interpolation in Software Verification

Reachability of B in ONE iteration: $R(c,d) \wedge T(c,d,c',d') \wedge c' \geq N \rightarrow B(c',d')$

$\{c = d = 0 \;\wedge N > 0 \;\wedge\; (\forall k)\;(0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c,d)$

while $(c < N)$ **do**

$C[c] \;:=\; D[d];$

$c \;:=\; c + 1;$
$d \;:=\; d + 1$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c',d')$

$$I(c',d') \equiv 0 < c' = 1 \wedge C[0] = D[0]$$

$$I(c'',d'') \equiv 0 < c'' = 2 \wedge C[0] = D[0] \wedge C[1] = D[1]$$

Task: Compute interpolant $I(c',d')$ from refutation by eliminating symbols c, d.

Interpolation in Software Verification

Reachability of B in TWO iterations

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c, d)$

while $(c < N)$ **do**

$$\begin{aligned} C[c] &:= D[d]; \\ c &:= c + 1; \\ d &:= d + 1 \end{aligned}$$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c', d')$

$$\begin{aligned} I(c', d') &\equiv 0 < c' = 1 \wedge C[0] = D[0] \\ I(c'', d'') &\equiv 0 < c'' = 2 \wedge C[0] = D[0] \wedge C[1] = D[1] \end{aligned}$$

Task: Compute interpolant $I(c'', d'')$ from refutation by eliminating c, d, c', d'.

Interpolation in Software Verification

Reachability of B in TWO iterations

$\{c = d = 0 \ \wedge N > 0 \ \wedge \ (\forall k)\ (0 \leq k < N \rightarrow D[k] = 0)\}$ precondition $R(c, d)$

while $(c < N)$ **do**

$$\begin{array}{lcl} C[c] & := & D[d]; \\ c & := & c + 1; \\ d & := & d + 1 \end{array}$$

od

$\{(\forall k)(0 \leq k < N \rightarrow C[k] = 0)\}$ postcondition $B(c', d')$

$$\begin{array}{lcl} I(c', d') & \equiv & (\forall k) 0 \leq k < c' \rightarrow C[k] = D[k] \\ I(c'', d'') & \equiv & (\forall k) 0 \leq k < c'' \rightarrow C[k] = D[k] \end{array}$$

Task: Compute interpolant $I(c'', d'')$ implying invariant in any state.

Interpolation in Software Verification

Tasks:

- **Proving:** Refute reachability properties
- **Extracting:** Compute interpolants from proofs

Outline

Interpolation

Theorem

Let R, B be closed formulas and let $R \vdash B$.

Then there exists a formula I such that

1. $R \vdash I$ *and* $I \vdash B$;
2. *every symbol of* I *occurs both in* R *and* B;

Interpolation

Theorem

Let R, B be closed formulas and let $R \vdash B$.

Then there exists a formula I such that

1. $R \vdash I$ *and* $I \vdash B$*;*
2. *every symbol of* I *occurs both in* R *and* B*;*

Any formula I with this property is called an interpolant of R and B.

Essentially, an interpolant is a formula that is

1. intermediate in power between R and B;
2. Uses only common symbols of R and B.

Interpolation

Theorem

Let R, B be closed formulas and let $R \vdash B$.

Then there exists a formula I such that

1. $R \vdash I$ *and* $I \vdash B$*;*
2. *every symbol of I occurs both in R and B;*

Any formula I with this property is called an interpolant of R and B.

Essentially, an interpolant is a formula that is

1. intermediate in power between R and B;
2. Uses only common symbols of R and B.

When we deal with refutations rather than proofs and have an unsatisfiable set $\{R, B\}$, it is convenient to use reverse interpolants of R and B, that is, a formula I such that

1. $R \vdash I$ and $\{I, B\}$ is unsatisfiable;
2. every symbol of I occurs both in R and B;

Interpolation Through Colors

- There are three colors: red, blue and grey.

Interpolation Through Colors

- There are three colors: red, blue and grey.
- Each symbol (function or predicate) is colored in exactly one of these colors.

Interpolation Through Colors

- There are three colors: red, blue and grey.
- Each symbol (function or predicate) is colored in exactly one of these colors.
- We have two formulas: R and B.
- Each symbol in R is either red or grey.
- Each symbol in B is either blue or grey.

Interpolation Through Colors

- There are three colors: red, blue and grey.
- Each symbol (function or predicate) is colored in exactly one of these colors.
- We have two formulas: R and B.
- Each symbol in R is either red or grey.
- Each symbol in B is either blue or grey.
- We know that $\vdash R \rightarrow B$.
- Our goal is to find a grey formula I such that:
  1. $\vdash R \rightarrow I$;
  2. $\vdash I \rightarrow B$.

Interpolation with Theories

- Theory T: any set of closed green formulas.
- $C_1, \ldots, C_n \vdash_T C$ denotes that the formula $C_1 \land \ldots \land C_1 \rightarrow C$ holds in all models of T.
- Interpreted symbols: symbols occurring in T.
- Uninterpreted symbols: all other symbols.

Interpolation with Theories

- Theory T: any set of closed green formulas.
- $C_1, \ldots, C_n \vdash_T C$ denotes that the formula $C_1 \land \ldots \land C_1 \rightarrow C$ holds in all models of T.
- Interpreted symbols: symbols occurring in T.
- Uninterpreted symbols: all other symbols.

Theorem

Let R, B be formulas and let $R \vdash_T B$.

Then there exists a formula I such that

1. $R \vdash_T I$ *and* $I \vdash B$*;*
2. *every uninterpreted symbol of I occurs both in R and B;*
3. *every interpreted symbol of I occurs in B.*

Likewise, there exists a formula I such that

1. $R \vdash I$ *and* $I \vdash_T B$*;*
2. *every uninterpreted symbol of I occurs both in R and B;*
3. *every interpreted symbol of I occurs in R.*

Local Derivations

A derivation is called local (well-colored) if each inference in it

$$\frac{C_1 \quad \cdots \quad C_n}{C}$$

either has no blue symbols or has no red symbols.
That is, one cannot mix blue and red in the same inference.

Local Derivations: Example

- R := $\forall x(x = a)$
- B := $c = b$
- Interpolant: $\forall x \forall y(x = y)$ (note: universally quantified!)

Local Derivations: Example

- R := $\forall x(x = a)$
- B := $c = b$
- Interpolant: $\forall x \forall y(x = y)$ (note: universally quantified!)

$$\dfrac{\dfrac{\dfrac{x = a}{c = a} \qquad \dfrac{x = a}{b = a}}{c = b} \qquad c \neq b}{\perp}$$

Local Derivations: Example

- R := $\forall x(x = a)$
- B := $c = b$
- Interpolant: $\forall x \forall y(x = y)$ (note: universally quantified!)

Non-local proof

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

Local Derivations: Example

- R := $\forall x (x = a)$
- B := $c = b$
- Interpolant: $\forall x \forall y (x = y)$ (note: universally quantified!)

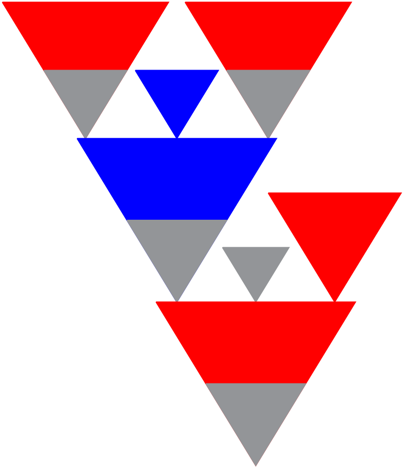
Non-local proof

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

Local Proof

$$\dfrac{\dfrac{\dfrac{x=a \quad y=a}{x=y} \qquad c \neq b}{y \neq b}}{\bot}$$

Shape of a local derivation

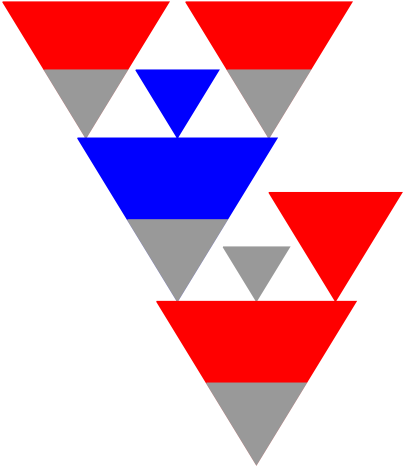


Symbol Eliminating Inference

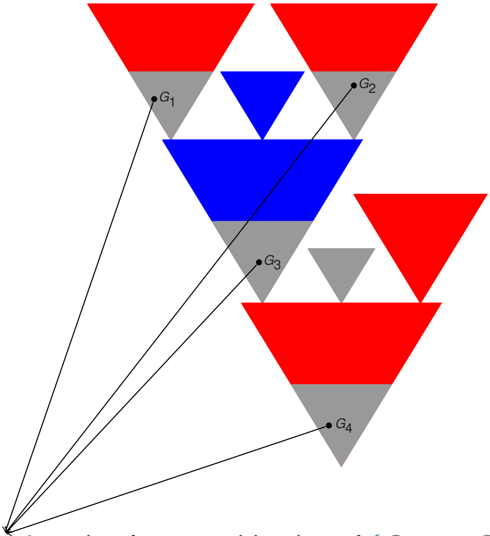
- At least one of the premises is not grey.
- The conclusion is grey.

$$\dfrac{\boxed{\dfrac{x = a \quad y = a}{x = y}} \qquad c \neq b}{\boxed{\dfrac{y \neq b}{\perp}}}$$

Extracting Interpolants from Local Proofs



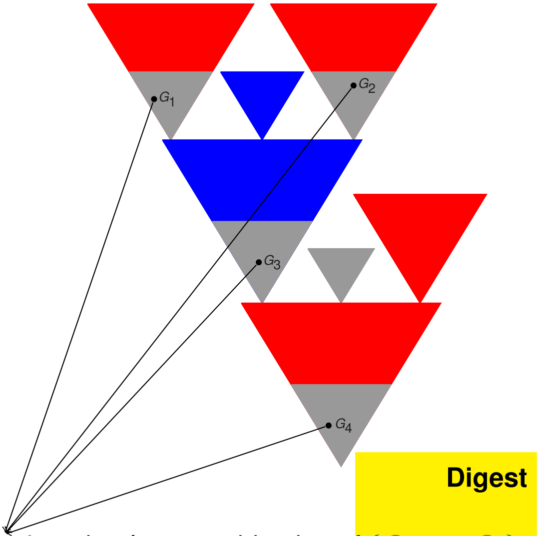
Extracting Interpolants from Local Proofs



Interpolant: boolean combination of $\{G_1, \ldots, G_4\}$

[McMillan05, KV09]

Extracting Interpolants from Local Proofs

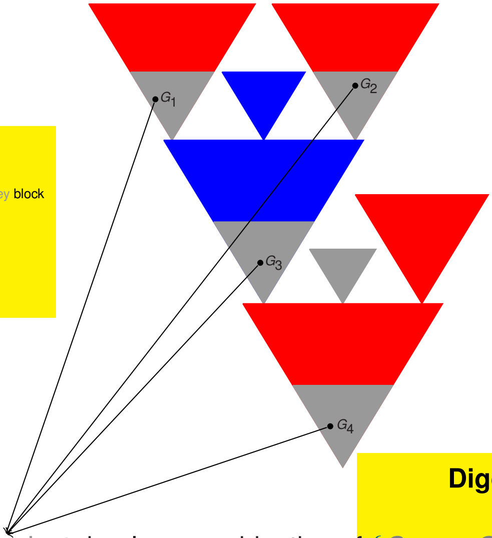


Interpolant: boolean combination of $\{G_1, \ldots, G_4\}$

Extracting Interpolants from Local Proofs

G is in the digest:

- comes from a red block
- followed by a blue or grey block



Digest

Interpolant: boolean combination of $\{G_1, \ldots, G_4\}$

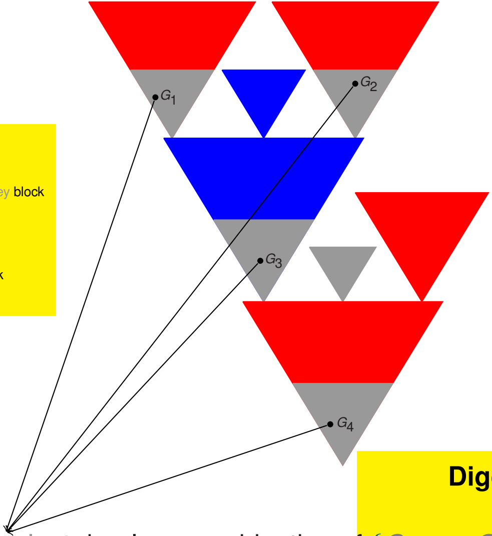
Extracting Interpolants from Local Proofs

G is in the digest:

- comes from a red block
- followed by a blue or grey block

or

- comes from a blue block
- followed by a red



Digest

Interpolant: boolean combination of $\{G_1, \ldots, G_4\}$

Extracting Interpolants from Local Proofs

Theorem

Let Π be a local refutation. Then one can extract from Π in linear time a reverse interpolant I of R and B. This interpolant is ground if all formulas in Π are ground.

Extracting Interpolants from Local Proofs

Theorem

Let Π be a local refutation. Then one can extract from Π in linear time a reverse interpolant I of R and B. This interpolant is ground if all formulas in Π are ground. This reverse interpolant is a boolean combination of conclusions of symbol-eliminating inferences of Π.

Extracting Interpolants from Local Proofs

Theorem

Let Π be a local refutation. Then one can extract from Π in linear time a reverse interpolant I of R and B. This interpolant is ground if all formulas in Π are ground. This reverse interpolant is a boolean combination of conclusions of symbol-eliminating inferences of Π.

What is remarkable in this theorem:

- No restriction on the calculus (only soundness required) – can be used with theories.
- Can generate interpolants in theories where no good interpolation algorithms exist.

Interpolation: Examples in Vampire

Our running example:

Local proof and interpolant: `vampire interpol1.p`

Non-local proof: `vampire interpol2.p`

What is Vampire?

An automated theorem prover for first-order logic and theories.

https://vprover.github.io/download.html

What is Vampire?

An automated theorem prover for first-order logic and theories.

`https://vprover.github.io/download.html`

- Completely automatic: once you started a proof attempt, it can only be interrupted by terminating the process.

What is Vampire?

An automated theorem prover for first-order logic and theories.

`https://vprover.github.io/download.html`

- Completely automatic: once you started a proof attempt, it can only be interrupted by terminating the process.
- Champion of the CASC world-cup in first-order theorem proving: won CASC $>$45 times.

Vampire:

▷ It produces detailed proofs but also supports finding finite models

▷ In normal operation it is saturation-based - it saturates a clausal form with respect to an inference system

▷ It is portfolio-based - it works best when you allow it to try lots of strategies

▷ It supports lots of extra features and options

Vampire:

- ▷ It produces detailed proofs but also supports finding finite models
- ▷ It competes with SMT solvers on their problems (thanks to our FOOL logic and AVATAR)
- ▷ In normal operation it is saturation-based - it saturates a clausal form with respect to an inference system
- ▷ It is portfolio-based - it works best when you allow it to try lots of strategies
- ▷ It supports lots of extra features and options helpful for program analysis by symbol elimination

Interpolation: Examples in Vampire

Our running example:

Local proof and interpolant: `vampire interpol1.p`

Non-local proof: `vampire interpol2.p`

Interpolation: Examples in Vampire

```
fof(fA,axiom, q(f(a)) & ~q(f(b)) ).
fof(fB,conjecture, ?[V]: V != c).
```

Non-local proof: `vampire interpol4.p`

Interpolation: Examples in Vampire

```
% request to generate an interpolant
vampire(option,show_interpolant,on).
% symbol coloring
vampire(symbol,predicate,q,1,left).
vampire(symbol,function,f,1,left).
vampire(symbol,function,a,0,left).
vampire(symbol,function,b,0,left).
vampire(symbol,function,c,0,right).
% formula R
vampire(left_formula).
  fof(fA,axiom, q(f(a)) & ~q(f(b)) ).
vampire(end_formula).
% formula B
vampire(right_formula).
  fof(fB,conjecture, ?[V]: V != c).
vampire(end_formula).
```

Local proof and interpolant: `vampire interpol3.p`

Outline

Localizing Proofs

Task: eliminate non-local inferences

Localizing Proofs

Task: eliminate non-local inferences

Idea: quantify away colored symbols

$\downarrow$

colored symbols replaced by logical variables.

Localizing Proofs

Task: eliminate non-local inferences

Idea: quantify away colored symbols

$\downarrow$

colored symbols replaced by logical variables.

Given $R(a) \vdash B$ where a is an uninterpreted constant not occurring in B.

Then, $R(a) \vdash (\exists x)R(x)$ and $(\exists x)R(x) \vdash B$.

Localizing Proofs

Task: eliminate non-local inferences

Idea: quantify away colored symbols

$\downarrow$

colored symbols replaced by logical variables.

Given $R(a) \vdash B$ where a is an uninterpreted constant not occurring in B.

Then, $R(a) \vdash (\exists x)R(x)$ and $(\exists x)R(x) \vdash B$.

$$\dfrac{\dfrac{R_1(a)}{R_2(a)} \quad B}{A} \qquad \Bigg\| \qquad \dfrac{\dfrac{R_1(a)}{(\exists x)R_2(x)} \quad B}{A}$$

Localizing Proofs

Task: eliminate non-local inferences

Idea: quantify away colored symbols

$\downarrow$

colored symbols replaced by logical variables.

Cons: Comes at the cost of using extra quantifiers.

Given $R(a) \vdash B$ where a is an uninterpreted constant not occurring in B.

Then, $R(a) \vdash (\exists x) R(x)$ and $(\exists x) R(x) \vdash B$.

$$\dfrac{\dfrac{R_1(a)}{R_2(a)} \quad B}{A} \qquad \Bigg\| \qquad \dfrac{\dfrac{R_1(a)}{(\exists x) R_2(x)} \quad B}{A}$$

Localizing Proofs

Task: eliminate non-local inferences

Idea: quantify away colored symbols

$\downarrow$

colored symbols replaced by logical variables.

Cons: Comes at the cost of using extra quantifiers.

But we can **minimise** the number of quantifiers in the interpolant.

Given $R(a) \vdash B$ where a is an uninterpreted constant not occurring in B.

Then, $R(a) \vdash (\exists x)R(x)$ and $(\exists x)R(x) \vdash B$.

$$\dfrac{\dfrac{R_1(a)}{R_2(a)} \quad B}{A} \qquad \Bigg\| \qquad \dfrac{\dfrac{R_1(a)}{(\exists x)R_2(x)} \quad B}{A}$$

Outline

Minimizing Interpolants

Our Interest: Small Interpolants

- in size;
- in weight;
- in the number of quantifiers;
- ...

Minimizing Interpolants

Our Interest: Small Interpolants

- in size;
- in weight;
- in the number of quantifiers;
- ...

Given $\vdash R \rightarrow B$, find a grey formula I:

- $\vdash R \rightarrow I$;
- $\vdash I \rightarrow B$;
- I is small.

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Hercule Poirot:

It is the little GREY CELLS*, mon ami, on which one must rely.*

Mon Dieu, mon ami, but use your little GREY CELLS*!*

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Idea: Change the grey areas of the local proof

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Idea: Change the grey areas of the local proof

Slicing off formulas

$$\dfrac{A_1 \quad \cdots \quad A_n \quad \dfrac{A_{n+1} \quad \cdots \quad A_m}{A}}{A_0} \quad \xrightarrow[\text{slicing off } A]{} \quad \dfrac{A_1 \quad \cdots \quad A_n \quad A_{n+1} \quad \cdots \quad A_m}{A_0}$$

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Idea: Change the grey areas of the local proof

Slicing off formulas

$$\dfrac{A_1 \quad \cdots \quad A_n \quad \dfrac{A_{n+1} \quad \cdots \quad A_m}{A}}{A_0} \quad \xrightarrow[\text{slicing off } A]{} \quad \dfrac{A_1 \quad \cdots \quad A_n \quad A_{n+1} \quad \cdots \quad A_m}{A_0}$$

If A is grey: Grey slicing

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Idea: Change the grey areas of the local proof

Slicing off formulas

$$\dfrac{B_0 \qquad \dfrac{R_0}{G_1}}{G_0} \quad \xrightarrow[\text{slicing off } G_1]{} \quad \dfrac{B_0 \qquad R_0}{G_0}$$

Minimizing Interpolant

Task: minimise interpolants = minimise digest

Idea: Change the grey areas of the local proof, but preserve locality!

Slicing off formulas

$$\dfrac{B_0 \qquad \dfrac{R_0}{G_1}}{G_0} \quad \xrightarrow{\text{slicing off } G_1} \quad \dfrac{B_0 \qquad R_0}{G_0}$$

Minimizing Interpolant

$$\dfrac{\dfrac{\dfrac{R_3 \qquad \dfrac{\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{G_4}}{G_5}}{G_6}}{R_4}}{G_7}}{\bot}$$

Minimizing Interpolant

$$\dfrac{\dfrac{\dfrac{R_3 \qquad \dfrac{\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{G_4}}{G_5}}{G_6}}{R_4}}{G_7}}{\bot}$$

Digest: $\{G_4, G_7\}$

Reverse interpolant: $G_4 \rightarrow G_7$

Minimizing Interpolant

$$\dfrac{\dfrac{\dfrac{R_3 \qquad \dfrac{\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{}}{G_5}}{G_6}}{R_4}}{G_7}}{\bot}$$

Minimizing Interpolant

$$\dfrac{\dfrac{\dfrac{R_3 \qquad \dfrac{\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{}}{G_5}}{G_6}}{R_4}}{G_7}}{\bot}$$

Digest: $\{G_5, G_7\}$

Reverse interpolant: $G_5 \to G_7$

Minimizing Interpolant

$$\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{}}{}$$

$$\dfrac{\dfrac{\dfrac{R_3 \qquad\qquad \overline{G_6}}{R_4}}{G_7}}{\bot}$$

Minimizing Interpolant

$$\dfrac{\dfrac{R_1 \qquad G_1}{G_3} \qquad \dfrac{B_1 \qquad G_2}{}}{}$$

$$\dfrac{\dfrac{\dfrac{R_3 \qquad\qquad \dfrac{}{G_6}}{R_4}}{G_7}}{\perp}$$

Digest: $\{G_6, G_7\}$

Reverse interpolant: $G_6 \rightarrow G_7$

Minimizing Interpolant

$$\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{}}{}$$

$$\dfrac{R_3 \qquad \dfrac{}{G_6}}{\dfrac{R_4}{}}$$

$$\dfrac{}{\bot}$$

Minimizing Interpolant

$$\cfrac{\cfrac{R_1 \quad G_1}{G_3} \qquad \cfrac{B_1 \quad G_2}{}}{}$$

$$\cfrac{\cfrac{R_3 \qquad \cfrac{}{G_6}}{R_4}}{}$$

$$\cfrac{}{\bot}$$

Digest: $\{G_6\}$

Reverse interpolant: $\neg G_6$

Minimizing Interpolant

$$\dfrac{\dfrac{\dfrac{R_3 \qquad \dfrac{\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{G_4}}{G_5}}{G_6}}{R_4}}{G_7}}{\bot}$$

Note that the interpolant has changed from $G_4 \rightarrow G_7$ to $\neg G_6$.

Minimizing Interpolant

$$\dfrac{\dfrac{\dfrac{R_3 \qquad \dfrac{\dfrac{\dfrac{R_1 \quad G_1}{G_3} \qquad \dfrac{B_1 \quad G_2}{G_4}}{G_5}}{G_6}}{R_4}}{G_7}}{\perp}$$

Note that the interpolant has changed from $G_4 \to G_7$ to $\neg G_6$.

- There is no obvious logical relation between $G_4 \to G_7$ and $\neg G_6$, for example none of these formulas implies the other one;
- These formulas may even have no common atoms or no common symbols.

Minimizing Interpolant

If grey slicing gives us very different interpolants, we can use it for finding small interpolants.

Problem: if the proof contains n grey formulas, the number of possible different slicing off transformations is 2^n.

Minimizing Interpolant

If grey slicing gives us very different interpolants, we can use it for finding small interpolants.

Problem: if the proof contains n grey formulas, the number of possible different slicing off transformations is 2^n.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

G_3, and at most one of G_1, G_2 can be sliced off.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

Some predicates on grey formulas:

- sliced(G): G was sliced off;
- red(G): the trace of G contains a red formula;
- blue(G): the trace of G contains a blue formula;
- grey(G): the trace of G contains only grey formulas;
- digest(G): G belongs to the digest.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\frac{\dfrac{R}{G_1} \quad \dfrac{B}{G_2}}{G_3}$$

Some predicates on grey formulas:

- $\text{sliced}(G)$: G was sliced off;
- $\text{red}(G)$: the trace of G contains a red formula;
- $\text{blue}(G)$: the trace of G contains a blue formula;
- $\text{grey}(G)$: the trace of G contains only grey formulas;
- $\text{digest}(G)$: G belongs to the digest.

$$\neg\text{sliced}(G_1) \rightarrow \text{grey}(G_1)$$
$$\text{sliced}(G_1) \rightarrow \text{red}(G_1)$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \quad \dfrac{B}{G_2}}{G_3}$$

Some predicates on grey formulas:

- sliced(G): G was sliced off;
- red(G): the trace of G contains a red formula;
- blue(G): the trace of G contains a blue formula;
- grey(G): the trace of G contains only grey formulas;
- digest(G): G belongs to the digest.

$$\neg \mathrm{sliced}(G_3) \rightarrow \mathrm{grey}(G_3)$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{grey}(G_3) \leftrightarrow \mathrm{grey}(G_1) \wedge \mathrm{grey}(G_2))$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{red}(G_3) \leftrightarrow \mathrm{red}(G_1) \vee \mathrm{red}(G_2))$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{blue}(G_3) \leftrightarrow \mathrm{blue}(G_1) \vee \mathrm{blue}(G_2))$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

Some predicates on grey formulas:

- sliced(G): G was sliced off;
- red(G): the trace of G contains a red formula;
- blue(G): the trace of G contains a blue formula;
- grey(G): the trace of G contains only grey formulas;
- digest(G): G belongs to the digest.

$$\text{digest}(G_1) \rightarrow \neg\text{sliced}(G_1)$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

Some predicates on grey formulas:

- $\mathsf{sliced}(G)$: G was sliced off;
- $\mathsf{red}(G)$: the trace of G contains a red formula;
- $\mathsf{blue}(G)$: the trace of G contains a blue formula;
- $\mathsf{grey}(G)$: the trace of G contains only grey formulas;
- $\mathsf{digest}(G)$: G belongs to the digest.

$$
\begin{array}{l}
\neg\mathsf{sliced}(G_1) \rightarrow \mathsf{grey}(G_1) \\
\mathsf{sliced}(G_1) \rightarrow \mathsf{red}(G_1) \\
\neg\mathsf{sliced}(G_3) \rightarrow \mathsf{grey}(G_3) \\
\mathsf{sliced}(G_3) \rightarrow (\mathsf{grey}(G_3) \leftrightarrow \mathsf{grey}(G_1) \land \mathsf{grey}(G_2)) \\
\mathsf{sliced}(G_3) \rightarrow (\mathsf{red}(G_3) \leftrightarrow \mathsf{red}(G_1) \lor \mathsf{red}(G_2)) \\
\mathsf{sliced}(G_3) \rightarrow (\mathsf{blue}(G_3) \leftrightarrow \mathsf{blue}(G_1) \lor \mathsf{blue}(G_2)) \\
\mathsf{digest}(G_1) \rightarrow \neg\mathsf{sliced}(G_1) \\
\ldots
\end{array}
$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \quad \dfrac{B}{G_2}}{G_3}$$

Express digest(G)

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

Express digest(G)

by considering the possibilities:

- G comes from a red/ blue/ grey formula
- G is followed by a red/ blue/ grey formula

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\cfrac{\cfrac{R}{G_1} \quad \cfrac{B}{G_2}}{G_3}$$

Express digest(G)

by considering the possibilities:

- G comes from a red/ blue/ grey formula

  rc(G)/bc(G)

- G is followed by a red/ blue/ grey formula

  bf(G)/rf(G)

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \quad \dfrac{B}{G_2}}{G_3}$$

Express digest(G)

by considering the possibilities:

- G comes from a red/ blue/ grey formula

  rc(G)/bc(G)

- G is followed by a red/ blue/ grey formula

  bf(G)/rf(G)

$$\text{digest}(G_3) \leftrightarrow (\text{rc}(G_3) \wedge \text{rf}(G_3)) \vee (\text{bc}(G_3) \wedge \text{bf}(G_3))$$

$$\text{rc}(G_3) \leftrightarrow (\neg\text{sliced}(G_3) \wedge (\text{red}(G_1) \vee \text{red}(G_2))$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \qquad \dfrac{B}{G_2}}{G_3}$$

Express digest(G)

by considering the possibilities:

- G comes from a red/ blue/ grey formula

  rc(G)/bc(G)

- G is followed by a red/ blue/ grey formula

  bf(G)/rf(G)

$$\neg\mathrm{sliced}(G_1) \rightarrow \mathrm{grey}(G_1)$$
$$\mathrm{sliced}(G_1) \rightarrow \mathrm{red}(G_1)$$
$$\neg\mathrm{sliced}(G_3) \rightarrow \mathrm{grey}(G_3)$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{grey}(G_3) \leftrightarrow \mathrm{grey}(G_1) \wedge \mathrm{grey}(G_2))$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{red}(G_3) \leftrightarrow \mathrm{red}(G_1) \vee \mathrm{red}(G_2))$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{blue}(G_3) \leftrightarrow \mathrm{blue}(G_1) \vee \mathrm{blue}(G_2))$$
$$\mathrm{digest}(G_1) \rightarrow \neg\mathrm{sliced}(G_1)$$

$$\mathrm{digest}(G_3) \leftrightarrow (\mathrm{rc}(G_3) \wedge \mathrm{rf}(G_3)) \vee (\mathrm{bc}(G_3) \wedge \mathrm{bf}(G_3))$$
$$\mathrm{rc}(G_3) \leftrightarrow (\neg\mathrm{sliced}(G_3) \wedge (\mathrm{red}(G_1) \vee \mathrm{red}(G_2))$$
$$\ldots$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT
- solutions encode all slicing off transformations

$$\dfrac{\dfrac{R}{G_1} \quad \dfrac{B}{G_2}}{G_3}$$

Express digest(G)

by considering the possibilities:

- G comes from a red/ blue/ grey formula

  rc(G)/bc(G)

- G is followed by a red/ blue/ grey formula

  bf(G)/rf(G)

$$\neg\mathrm{sliced}(G_1) \rightarrow \mathrm{grey}(G_1)$$
$$\mathrm{sliced}(G_1) \rightarrow \mathrm{red}(G_1)$$
$$\neg\mathrm{sliced}(G_3) \rightarrow \mathrm{grey}(G_3)$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{grey}(G_3) \leftrightarrow \mathrm{grey}(G_1) \wedge \mathrm{grey}(G_2))$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{red}(G_3) \leftrightarrow \mathrm{red}(G_1) \vee \mathrm{red}(G_2))$$
$$\mathrm{sliced}(G_3) \rightarrow (\mathrm{blue}(G_3) \leftrightarrow \mathrm{blue}(G_1) \vee \mathrm{blue}(G_2))$$
$$\mathrm{digest}(G_1) \rightarrow \neg\mathrm{sliced}(G_1)$$

$$\mathrm{digest}(G_3) \leftrightarrow (\mathrm{rc}(G_3) \wedge \mathrm{rf}(G_3)) \vee (\mathrm{bc}(G_3) \wedge \mathrm{bf}(G_3))$$
$$\mathrm{rc}(G_3) \leftrightarrow (\neg\mathrm{sliced}(G_3) \wedge (\mathrm{red}(G_1) \vee \mathrm{red}(G_2))$$
$$\ldots$$

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT;
- solutions encode all slicing off transformations;
- compute small interpolants: smallest digest of grey formulas;

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}} \left(\sum_{G_i} \text{digest}(G_i) \right)$$

- use a pseudo-boolean optimisation tool or an SMT solver to minimise interpolants;
- minimising interpolants is an NP-complete problem.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT;
- solutions encode all slicing off transformations;
- compute small interpolants: smallest digest of grey formulas;

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}}\left(\sum_{G_i}\text{digest}(G_i)\right)$$

- use a pseudo-boolean optimisation tool or an SMT solver to minimise interpolants;
- minimising interpolants is an NP-complete problem.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT;
- solutions encode all slicing off transformations;
- compute small interpolants: smallest digest of grey formulas;

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}}\left(\sum_{G_i}\mathrm{digest}(G_i)\right)$$

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}}\left(\sum_{G_i}\mathrm{quantifier\_number}(\mathrm{G_i})\ \mathrm{digest}(G_i)\right)$$

- use a pseudo-boolean optimisation tool or an SMT solver to minimise interpolants;
- minimising interpolants is an NP-complete problem.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT;
- solutions encode all slicing off transformations;
- compute small interpolants: smallest digest of grey formulas;

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}} \left(\sum_{G_i} \mathrm{digest}(G_i) \right)$$

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}} \left(\sum_{G_i} \mathrm{quantifier\_number}(\mathrm{G_i})\ \mathrm{digest}(G_i) \right)$$

- use a pseudo-boolean optimisation tool or an SMT solver to minimise interpolants;
- minimising interpolants is an NP-complete problem.

Minimizing Interpolant

Solution:

- encode all sequences of transformations as an instance of SAT;
- solutions encode all slicing off transformations;
- compute small interpolants: smallest digest of grey formulas;

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}}\left(\sum_{G_i}\mathrm{digest}(G_i)\right)$$

$$\mathtt{min}_{\{G_{i_1},\ldots,G_{i_n}\}}\left(\sum_{G_i}\mathrm{quantifier\_number}(\mathrm{G_i})\;\mathrm{digest}(G_i)\right)$$

- use a pseudo-boolean optimisation tool or an SMT solver to minimise interpolants;
- minimising interpolants is an NP-complete problem.

Experiments with Small Interpolants

- Implemented in Vampire;
- We used Yices for solving pseudo-boolean constraints;
- Experimental results:
  - 9632 first-order examples from the TPTP library:

    for example, for 2000 problems the size of the interpolants became 20-49 times smaller;
  - 4347 SMT examples:
    - we used Z3 for proving SMT examples;
    - Z3 proofs were localised in Vampire;
    - small interpolants were generated for 2123 SMT examples.

Experiments with Small Interpolants

- Implemented in Vampire;
- We used Yices for solving pseudo-boolean constraints;
- Experimental results:
  - 9632 first-order examples from the TPTP library:

    for example, for 2000 problems the size of the interpolants became 20-49 times smaller;
  - 4347 SMT examples:
    - we used Z3 for proving SMT examples;
    - Z3 proofs were localised in Vampire;
    - small interpolants were generated for 2123 SMT examples.

Experiments with Small Interpolants

- Implemented in Vampire;
- We used Yices for solving pseudo-boolean constraints;
- Experimental results:
  - 9632 first-order examples from the TPTP library:

    for example, for 2000 problems the size of the interpolants became 20-49 times smaller;
  - 4347 SMT examples:
    - we used Z3 for proving SMT examples;
    - Z3 proofs were localised in Vampire;
    - small interpolants were generated for 2123 SMT examples.

Outline

Quantifier Complexity of Interpolants

Local Proofs Do Not Always Exist

- R: $(\forall x)p(r, x)$
- B: $(\forall y)\neg p(y, b)$
- Reverse interpolant I of R and B: $(\exists y)(\forall x)p(y, x)$.
- Note: R and B contain no quantifier alternations, yet I contains quantifier alternations. One can prove that every interpolant of this formula must have at least one quantifier alternation.
- There is no local refutation of R, B in the resolution/superposition calculus.
- There is a non-local one:

$$\frac{p(r, x) \qquad \neg p(y, b)}{\bot}$$

Quantifier Complexity of Interpolants

Local Proofs Do Not Always Exist

- R: $(\forall x)p(r, x)$
- B: $(\forall y)\neg p(y, b)$
- Reverse interpolant I of R and B: $(\exists y)(\forall x)p(y, x)$.
- Note: R and B contain no quantifier alternations, yet I contains quantifier alternations. One can prove that every interpolant of this formula must have at least one quantifier alternation.
- There is no local refutation of R, B in the resolution/superposition calculus.
- There is a non-local one:

$$\frac{p(r, x) \quad \neg p(y, b)}{\bot}$$

Quantifier Complexity of Interpolants

Local Proofs Do Not Always Exist

- R: $(\forall x)p(r, x)$
- B: $(\forall y)\neg p(y, b)$
- Reverse interpolant I of R and B: $(\exists y)(\forall x)p(y, x)$.
- Note: R and B contain no quantifier alternations, yet I contains quantifier alternations. One can prove that every interpolant of this formula must have at least one quantifier alternation.
- There is no local refutation of R, B in the resolution/superposition calculus.
- There is a non-local one:

$$\frac{p(r, x) \quad \neg p(y, b)}{\bot}$$

Quantifier Complexity of Interpolants

Theorem There is no lower bound on the number of quantifier alternations in interpolants of universal sentences.

That is, for every positive integer n there exist universal sentences R, B such that $\{R, B\}$ is unsatisfiable and every reverse interpolant of R and B has at least n quantifier alternations.

Quantifier Complexity of Interpolants

Example

Take the formula A: $\forall x_1 \exists y_1 \forall x_1 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots)$ and let R be obtained by skolemizing A and B be obtained by skolemizing $\neg A$:

$$
\begin{aligned}
R &= \forall x_1 \forall x_2 \ldots p(x_1, r_1(x_1), x_2, r_2(x_1, x_2), \ldots) \\
B &= \forall y_1 \forall y_2 \ldots \neg p(b_1, y_1, b_2(y_1), y_2, \ldots) \\
I &= \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots)
\end{aligned}
$$

There is no reverse interpolant with a smaller number of quantifier alternations.

The resolution refutation consists of a single step deriving the empty clause from R and B.

Quantifier Complexity of Interpolants

Example

Take the formula A: $\forall x_1 \exists y_1 \forall x_1 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots)$ and let R be obtained by skolemizing A and B be obtained by skolemizing $\neg A$:

$$\begin{aligned} R &= \forall x_1 \forall x_2 \ldots p(x_1, r_1(x_1), x_2, r_2(x_1, x_2), \ldots) \\ B &= \forall y_1 \forall y_2 \ldots \neg p(b_1, y_1, b_2(y_1), y_2, \ldots) \\ I &= \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots) \end{aligned}$$

There is no reverse interpolant with a smaller number of quantifier alternations.

The resolution refutation consists of a single step deriving the empty clause from R and B.

Quantifier Complexity of Interpolants

Example

Take the formula A: $\forall x_1 \exists y_1 \forall x_1 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots)$ and let R be obtained by skolemizing A and B be obtained by skolemizing $\neg A$:

$$\begin{aligned}
R &= \forall x_1 \forall x_2 \ldots p(x_1, r_1(x_1), x_2, r_2(x_1, x_2), \ldots) \\
B &= \forall y_1 \forall y_2 \ldots \neg p(b_1, y_1, b_2(y_1), y_2, \ldots) \\
I &= \forall x_1 \exists y_1 \forall x_2 \exists y_2 \ldots p(x_1, y_1, x_2, y_2, \ldots)
\end{aligned}$$

There is no reverse interpolant with a smaller number of quantifier alternations.

The resolution refutation consists of a single step deriving the empty clause from R and B.

Quantifier Complexity of Interpolants

Bad News for Local Proof Systems

Let S be an inference system with the following property: for every red formula R and blue formula B, if $\{R, B\}$ is unsatisfiable, then there is a local refutation of R, B in S.

Then the number of quantifier alternations in refutations of universal formulas of S is not bound by any positive integer.

Quantifier Complexity of Interpolants

- There is no bound on the number of quantifier alternations in reverse interpolants of universal formulas.
- Any complete local proof system for first-order predicate logic must have inferences dealing with formulas of an arbitrary quantifier complexity, even if the input formulas have no quantifier alternations.
- There is no simple modification of the superposition calculus for logic with/without equality in which every unsatisfiable formula has a local refutation.

Quantifier Complexity of Interpolants

- There is no bound on the number of quantifier alternations in reverse interpolants of universal formulas.
- Any complete local proof system for first-order predicate logic must have inferences dealing with formulas of an arbitrary quantifier complexity, even if the input formulas have no quantifier alternations.
- There is no simple modification of the superposition calculus for logic with/without equality in which every unsatisfiable formula has a local refutation.

Quantifier Complexity of Interpolants

- There is no bound on the number of quantifier alternations in reverse interpolants of universal formulas.
- Any complete local proof system for first-order predicate logic must have inferences dealing with formulas of an arbitrary quantifier complexity, even if the input formulas have no quantifier alternations.
- There is no simple modification of the superposition calculus for logic with/without equality in which every unsatisfiable formula has a local refutation.