Interpolation, Symbol Eliminating, and Loop Analysis

Giles Reger, Martin Suda,
Laura Kovács, and Andrei Voronkov

University of Manchester and TU Wein

Outline

Interpolation and Colored Proofs

Symbol Elimination and Loop Invariant Generation

Interpolation

Theorem

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

Then there exists a formula I such that

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

Interpolation

Theorem

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

Then there exists a formula I such that

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

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

Essentially, an interpolant is a formula that is

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

Interpolation has many uses in verification.

Interpolation

Theorem

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

Then there exists a formula I such that

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

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

Essentially, an interpolant is a formula that is

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

Interpolation has many uses in verification.

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

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

Interpolation in Verification

Interpolation Through Colors

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

Interpolation Through Colors

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

Interpolation Through Colors

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

Interpolation Through Colors

- There are three colors: blue, red and green.
- Each symbol (function or predicate) is colored in exactly one of these colors.
- We have two formulas: A and B.
- Each symbol in A is either blue or green.
- Each symbol in B is either red or green.
- We know that $\vdash A \rightarrow B$.
- Our goal is to find a green formula I such that
  1. $\vdash A \rightarrow I$;
  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 $C_1 \wedge \ldots \wedge C_1 \rightarrow C$ 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 $C_1 \wedge \ldots \wedge C_1 \rightarrow C$ holds in all models of T.
- Interpreted symbols: symbols occurring in T.
- Uninterpreted symbols: all other symbols.

Theorem

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

Then there exists a formula I such that

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

Likewise, there exists a formula I such that

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

Local Derivations

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

$$\frac{C_! \quad \cdots \quad C_n}{C}$$

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

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

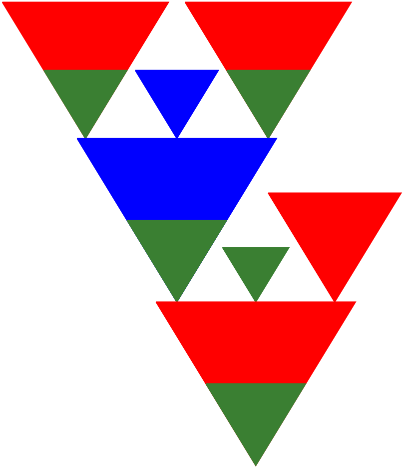
Local Derivations: Example

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

A local refutation in the superposition calculus:

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

Shape of a local derivation



Symbol Eliminating Inference

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

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

Extracting Interpolants from Local Proofs

Theorem

Let Π be a local refutation. Then one can extract from Π in linear time a reverse interpolant I of A 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 A 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 A 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

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

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 L
vampire(left_formula).
  fof(fA,axiom, q(f(a)) & ~q(f(b)) ).
vampire(end_formula).
% formula R
vampire(right_formula).
  fof(fB,conjecture, ?[V]: V != c).
vampire(end_formula).
```

Outline

Interpolation and Colored Proofs

Symbol Elimination and Loop Invariant Generation

Symbol Elimination

Colored proofs can also be used for an interesting application. Suppose that we have a set of formulas in some language and want to derive consequences of these formulas in a subset of this language.

Symbol Elimination

Colored proofs can also be used for an interesting application. Suppose that we have a set of formulas in some language and want to derive consequences of these formulas in a subset of this language.

Then we declare the symbols to be eliminated colored and ask Vampire to output symbol-eliminating inferences.

Symbol Elimination

Colored proofs can also be used for an interesting application. Suppose that we have a set of formulas in some language and want to derive consequences of these formulas in a subset of this language.

Then we declare the symbols to be eliminated colored and ask Vampire to output symbol-eliminating inferences.

This technique was used in our experiments on automatic loop invariant generation.

invgen: a Loop Analysis Tool Based On Vampire

Reasoning About Loops Using Vampire,
Laura Kovács and Simon Robillard, LPAR 2015

`https://github.com/simonr89/invgen`

`http://www.cse.chalmers.se/~simrob/downloads/loop_analysis_benchmarks.zip`

- simple fragment of programs with loops
- generation and verification mode
- supply axioms describing the loop semantics and simple properties of the scalar variables
- derive other interesting properties using symbol elimination

invgen: a Loop Analysis Tool Based On Vampire

Reasoning About Loops Using Vampire,
Laura Kovács and Simon Robillard, LPAR 2015

`https://github.com/simonr89/invgen`

`http://www.cse.chalmers.se/~simrob/downloads/loop_analysis_benchmarks.zip`

- simple fragment of programs with loops
- generation and verification mode
- supply axioms describing the loop semantics and simple properties of the scalar variables
- derive other interesting properties using symbol elimination

invgen: a Loop Analysis Tool Based On Vampire

Reasoning About Loops Using Vampire,
Laura Kovács and Simon Robillard, LPAR 2015

`https://github.com/simonr89/invgen`

`http://www.cse.chalmers.se/~simrob/downloads/loop_analysis_benchmarks.zip`

- simple fragment of programs with loops
- generation and verification mode
- supply axioms describing the loop semantics and simple properties of the scalar variables
- derive other interesting properties using symbol elimination