Symbolic Computation and Theorem Proving in Program Analysis

Laura Kovács

Chalmers

Outline

Part 1: Weakest Precondition for Program Analysis and Verification

Part 2: Polynomial Invariant Generation (TACAS'08, LPAR'10)

Part 3: Quantified Invariant Generation (FASE'09, MICAI'11)

Part 4: Invariants, Interpolants and Symbol Elimination (CADE'09, POPL'12, APLAS'12)

Outline

Invariants, Symbol Elimination, and Interpolation

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

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

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

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

end do

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

Invariants, Symbol Elimination, and Interpolation

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

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

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

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

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

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

Invariants, Symbol Elimination, and Interpolation

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

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

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

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

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

end do

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

Refutation: $A(c,d) \wedge T(c,d,c',d') \wedge \neg B(c',d')$

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

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

Invariants, Symbol Elimination, and Interpolation

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

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

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

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

$c \ := \ c+1;$

$d \ := \ d+1$

end do

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

Refutation: $A(c,d) \wedge T(c,d,c',d') \wedge \neg B(c',d')$

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

$A(c,d) \wedge T(c,d,c',d') \rightarrow I(c',d')$ and $I(c',d') \wedge \neg B(c',d') \rightarrow \bot$

Invariants, Symbol Elimination, and Interpolation

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

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

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

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

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

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

Refutation: $A(c,d) \wedge T(c,d,c',d') \wedge \neg B(c',d')$

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

$A(c,d) \wedge T(c,d,c',d') \rightarrow I(c',d')$ and $I(c',d') \wedge \neg B(c',d') \rightarrow \bot$

Taks: Compute interpolant $I(c',d')$ by eliminating symbols c, d.

Invariants, Symbol Elimination, and Interpolation

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

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

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

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

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

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

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

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

Taks: Compute interpolant $I(c',d')$ by eliminating symbols c, d.

Invariants, Symbol Elimination, and Interpolation

Reachability of B in TWO iterations: $A(c,d) \wedge T(c,d,c',d') \wedge T(c',d',c'',d'') \rightarrow B(c'',d'')$

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

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

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

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

end do

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

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

Taks: Compute interpolant $I(c'',d'')$ by eliminating symbols c, d, c', d'.

Invariants, Symbol Elimination, and Interpolation

Reachability of B in TWO iterations: $A(c,d) \wedge T(c,d,c',d') \wedge T(c',d',c'',d'') \rightarrow B(c'',d'')$

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

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

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

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

end do

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

$$I(c',d') \equiv (\forall k) 0 \leq k < c' \rightarrow C[k] = D[k]$$

$$I(c'',d'') \equiv (\forall k) 0 \leq k < c'' \rightarrow C[k] = D[k]$$

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

Outline

Symbol Elimination and Interpolation

What is an Interpolant?

Computing Interpolants

- Local Derivations
- Symbol Eliminations
- Building Interpolants from Proof

Summary: Invariants, Symbol Elimination, Interpolants

Notation

- First-order predicate logic with equality.
- $\top$: always true,
  $\bot$: always false.
- $\forall A$: universal closure of A.
- Symbols:
  - predicate symbols;
  - function symbols;
  - constants.

  Equality is part of the language $\rightarrow$ equality is not a symbol.
- $\mathcal{L}_A$: the language of A: the set of all formulas built from the symbols occurring in A.

What is an Interpolant?

Let A, B be closed formulas such that $A \rightarrow B$.

Theorem (Craig's Interpolation Theorem)

There exists a closed formula $I \in \mathcal{L}_A \cap \mathcal{L}_B$ such that

$$A \rightarrow I \quad \textit{and} \quad I \rightarrow B.$$

I is an interpolant of A and B.

Note: if A and B are ground, they also have a ground interpolant.

What is an Interpolant?

Let A, B be closed formulas such that $A \rightarrow B$.

Theorem (Craig's Interpolation Theorem)

There exists a closed formula $I \in \mathcal{L}_A \cap \mathcal{L}_B$ *such that*

$$A \rightarrow I \quad \textit{and} \quad I \rightarrow B.$$

I is an interpolant of A and B.

$\boxed{\text{Reverse interpolant of } A \text{ and } B\text{:}}$ any formula I such that

$$A \rightarrow I \quad \text{and} \quad I, \neg B \rightarrow \bot.$$

Interpolation with Theories

- Theory T: any set of closed formulas.
- $C_1, \ldots, C_n \rightarrow_T C$ means 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 $A \rightarrow_T B$.

Then there exists a formula I such that

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

Interpolation with Theories

- Theory T: any set of closed formulas.
- $C_1, \ldots, C_n \rightarrow_T C$ means 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 $A \rightarrow_T B$.

Then there exists a formula I such that

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

Computing Interpolants using Inference Systems

- Inference Rule:

$$\frac{A_1 \quad \dots \quad A_n}{A}$$

- Inference system: a set of inference rules.
- Axiom: an inference rule with 0 premises.
- Derivation of A: tree with the root A built from inferences.

Interpolants and Local AB-Derivations

AB-derivation

Let $\mathcal{L} = \mathcal{L}_A \cap \mathcal{L}_B$.

A derivation Π is an AB-derivation if

(AB1) For every leaf C of Π one of following conditions holds:

1. $A \rightarrow_T \forall C$ and $C \in \mathcal{L}_A$ or
2. $B \rightarrow_T \forall C$ and $C \in \mathcal{L}_B$.

(AB2) For every inference

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

of Π we have $\forall C_1, \ldots, \forall C_n \rightarrow_T \forall C$.

We will refer to property (AB2) as soundness.

Interpolants and Local AB-Derivations

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

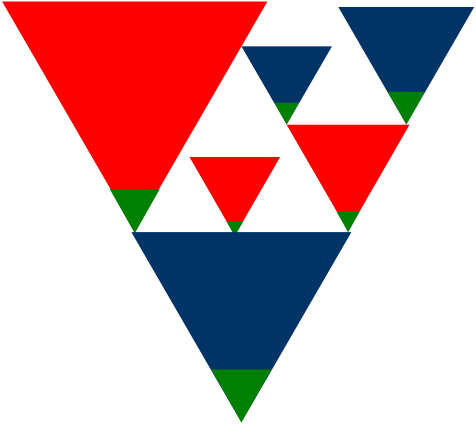
This inference is local if the following two conditions hold:

(L1) Either $\{C_1, \ldots, C_n, C\} \subseteq \mathcal{L}_A$ or $\{C_1, \ldots, C_n, C\} \subseteq \mathcal{L}_B$.

(L2) If all of the formulas $C_1, \ldots, C_n$ are colorless, then C is colorless, too.

A derivation is called local if so is every inference of this derivation.

Shape of local derivations for $A \to B$



Local Derivations: Example $A \rightarrow B$

[demo]

- $A := \forall x(x = c)$
- $B := a = b$
- Universal interpolant I: $\forall x \forall y(x = y)$

A local refutation of in the superposition calculus:

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

Local Derivations: Example $A \to B$

[demo]

- A := $\forall x(x = c)$
- B := $a = b$
- Universal interpolant I: $\forall x \forall y(x = y)$

A local refutation of in the superposition calculus:

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

Interpolants and Symbol Eliminating Inference

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

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

Interpolant $\forall x \forall y (x = y)$: conclusion of a symbol-eliminating inference.

Interpolants and Symbol Eliminating Inference

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

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

Interpolant $\forall x \forall y (x = y)$: conclusion of a symbol-eliminating inference.

Extracting Interpolants from Local Proofs

Theorem (CADE'09)

Let Π be a **closed local** AB**-refutation**.

Then:

- *A reverse interpolant I of A and B can be extracted from Π in linear time.*
- *I is ground if all formulas in Π are ground.*
- *I is a boolean combination of conclusions of symbol-eliminating inferences of Π.*

NOTE:

- No restriction on the calculus (only soundness required)
  – can be used with theories.
- Can generate interpolants in theories where no good interpolation algorithms exist.
- Shift of interest: what matters are symbol-eliminating inferences.

Extracting Interpolants from Local Proofs

Theorem (CADE'09)

Let Π be a **closed local** ***AB*-refutation**.

Then:

- *A reverse interpolant I of A and B can be extracted from Π in linear time.*
- *I is ground if all formulas in Π are ground.*
- *I is a boolean combination of conclusions of symbol-eliminating inferences of Π.*

NOTE:

- No restriction on the calculus (only soundness required)
  – can be used with theories.
- Can generate interpolants in theories where no good interpolation algorithms exist.
- Shift of interest: what matters are symbol-eliminating inferences.

Extracting Interpolants from Local Proofs

Theorem (CADE'09)

Let Π be a **closed local AB-refutation***.*

Then:

- *A reverse interpolant I of A and B can be extracted from Π in linear time.*
- *I is ground if all formulas in Π are ground.*
- *I is a boolean combination of conclusions of symbol-eliminating inferences of Π.*

NOTE:

- No restriction on the calculus (only soundness required)
  – can be used with theories.
- Can generate interpolants in theories where no good interpolation algorithms exist.
- Shift of interest: what matters are symbol-eliminating inferences.

Building Interpolants from Proofs

- Problem: generation of proofs giving interpolants.
  - Idea 1: look for local refutations only;
  - Idea 2: find calculi that guarantee that local proofs exist.
    - LASCA: Superposition + Linear Arithmetic;
    - Separating orderings (colored symbols are the greatest).

Theorem (CADE'09)

If $\succ$ is separating, then every AB-derivation in LASCA is local.

First-order interpolation implemented in Vampire.

Building Interpolants from Proofs

- Problem: generation of proofs giving interpolants.
  - Idea 1: look for local refutations only;
  - Idea 2: find calculi that guarantee that local proofs exist.
    - LASCA: Superposition + Linear Arithmetic;
    - Separating orderings (colored symbols are the greatest).

Theorem (CADE'09)

If $\succ$ is separating, then every AB-derivation in LASCA is local.

First-order interpolation implemented in Vampire.

Building Interpolants from Proofs

- Problem: generation of proofs giving interpolants.
  - Idea 1: look for local refutations only;
  - Idea 2: find calculi that guarantee that local proofs exist.
    - LASCA: Superposition + Linear Arithmetic;
    - Separating orderings (colored symbols are the greatest).

Theorem (CADE'09)

If $\succ$ is separating, then every AB-derivation in LASCA is local.

First-order interpolation implemented in Vampire.

Building Interpolants from Proofs

- Problem: generation of proofs giving interpolants.
  - Idea 1: look for local refutations only;
  - Idea 2: find calculi that guarantee that local proofs exist.
    - LASCA: Superposition + Linear Arithmetic;
    - Separating orderings (colored symbols are the greatest).

Theorem (CADE'09)

If $\succ$ is separating, then every AB-derivation in LASCA is local.

First-order interpolation implemented in Vampire.

Formulas	Coloring	Reverse Interpolant
$L: \; z < 0 \wedge x \leq z \wedge y \leq x$ $R: \; y \leq 0 \wedge x + y \geq 0$	left: z right: -	$y \leq x \wedge x < 0$
$L: \; g(a) = c + 5 \wedge f(g(a)) \geq c + 1$ $R: \; h(b) = d + 4 \wedge d = c + 1 \wedge f(h(b)) < c + 1$	left: g, a right: h, b	$c + 1 \leq f(c + 5)$
$L: \; p \leq c \wedge c \leq q \wedge f(c) = 1$ $R: \; q \leq d \wedge d \leq p \wedge f(d) = 0$	left: c right: d	$p \leq q \wedge (q > p \vee f(p) = 1)$
$L: \; f(x_1) + x_2 = x_3 \wedge f(y_1) + y_2 = y_3 \wedge y_1 \leq x_1$ $R: \; x_2 = g(b) \wedge y_2 = g(b) \wedge x_1 \leq y_1 \wedge x_3 < y_3$	left: f right: g, b	$x_1 > y_1 \vee x_2 \neq y_2 \vee x_3 = y_3$
$L: \; c_2 = car(c_1) \wedge c_3 = cdr(c_1) \wedge \neg(atom(c_1))$ $R: \; c_1 \neq cons(c_2, c_3)$	left: $car, cons$ right: -	$\neg atom(c_1) \wedge c_1 = cons(c_2, c_3)$
$L: \; Q(f(a)) \wedge \neq Q(f(b))$ $R: \; f(V) = c$	left: Q, a, b right: c	$\exists x, y : f(x) \neq f(y)$
$L: \; a = c \wedge f(c) = a$ $R: \; c = b \wedge \neq (b = f(c))$	left: a right: b	$c = f(c)$
$L: \; True \wedge a'[x'] = y \wedge x' = x \wedge y' = y + 1 \wedge z' = x'$ $R: \; \neg(y' = a'[z'] + 1)$	left: x, y right: -	$1 + a'[x'] = y' \wedge x' = z'$

Table : Interpolation with Vampire, within 1 second time limit.

Symbol Elimination and Interpolation

Interpolation Through Colors in Vampire

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

Interpolation Through Colors in Vampire

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

Interpolation Through Colors in Vampire

- 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 in Vampire

- 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 $\rightarrow A \rightarrow B$.
- Our goal is to find a green formula I such that
  1. $\rightarrow A \rightarrow I$;
  2. $\rightarrow I \rightarrow B$.

Interpolation Example in Vampire

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

Interpolation Example 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).
```

Symbol Elimination and Interpolation

Given: a problem (*an interpolation problem*)
Generate: a formula (*an interpolant*)

$$
\begin{aligned}
&\text{-}1 + a + \text{-}a = \text{-}1 \wedge \\
&\forall x(\neg(x \leq 5) \vee \text{-}6 + x \leq \text{-}1) \wedge \\
&\text{-}(\text{-}1 + \text{-}1 + a) = \text{-}1 \wedge \\
&\forall x((1 \leq x + \text{--}(\text{-}1 + a) \vee \neg(\text{-}1 \leq x))) \wedge \\
&(a \leq 6 \vee 1 \leq a + \text{-}1) \wedge \\
&\forall x(\neg(\text{-}1 \leq x) \vee \ \neg(x \leq \text{-}2)) \wedge \\
&\forall x(\text{-}1 \leq x + \text{-}a \vee \neg(\text{-}1 + a \leq x)) \wedge \\
&\forall x(\text{-}1 + x = 1 + \text{-}2 + x) \wedge \\
&\text{-}a + \text{-}1 + a = \text{-}1 \wedge \\
&\forall x(\neg(\text{--}(\text{-}1 + a) \leq x) \vee 1 \leq x + \text{-}1) \wedge \\
&\forall x((\neg(x \leq 4) \vee \text{-}5 + x \leq \text{-}1)) \wedge \\
&\forall x(x + \text{-}3 \leq \text{-}1 \vee \neg(x \leq 2)) \wedge \\
&\forall x(\neg(x \leq 3) \vee \text{-}4 + x \leq \text{-}1) \wedge \\
&\forall x(x + \text{-}a \leq \text{-}1 \vee \neg(x \leq \text{-}1 + a)) \wedge \\
&\forall x(\text{-}1 + x = \text{-}1 + \text{-}1 + a + \text{-}(\text{-}1 + a) + x) \wedge \\
&6 \leq b
\end{aligned}
$$

Given: a problem (*an interpolation problem*)
Generate: a formula (*an interpolant*)

$$
\begin{aligned}
&\text{-}1 + a + \text{-}a = \text{-}1 \wedge \\
&\forall x(\neg(x \leq 5) \vee \text{-}6 + x \leq \text{-}1) \wedge \\
&\text{-}(\text{-}1 + \text{-}1 + a) = \text{-}1 \wedge \\
&\forall x((1 \leq x + \text{--}(\text{-}1 + a) \vee \neg(\text{-}1 \leq x))) \wedge \\
&(a \leq 6 \vee 1 \leq a + \text{-}1) \wedge \\
&\forall x(\neg(\text{-}1 \leq x) \vee \ \neg(x \leq \text{-}2)) \wedge \\
&\forall x(\text{-}1 \leq x + \text{-}a \vee \neg(\text{-}1 + a \leq x)) \wedge \\
&\forall x(\text{-}1 + x = 1 + \text{-}2 + x) \wedge \\
&\text{-}a + \text{-}1 + a = \text{-}1 \wedge \\
&\forall x(\neg(\text{--}(\text{-}1 + a) \leq x) \vee 1 \leq x + \text{-}1) \wedge \\
&\forall x((\neg(x \leq 4) \vee \text{-}5 + x \leq \text{-}1)) \wedge \\
&\forall x(x + \text{-}3 \leq \text{-}1 \vee \neg(x \leq 2)) \wedge \\
&\forall x(\neg(x \leq 3) \vee \text{-}4 + x \leq \text{-}1) \wedge \\
&\forall x(x + \text{-}a \leq \text{-}1 \vee \neg(x \leq \text{-}1 + a)) \wedge \\
&\forall x(\text{-}1 + x = \text{-}1 + \text{-}1 + a + \text{-}(\text{-}1 + a) + x) \wedge \\
&6 \leq b
\end{aligned}
$$

or

$$
\begin{aligned}
&\neg(a \leq 6) \wedge \\
&\text{-}a \leq \text{-}1 \wedge \\
&\neg(\text{-}1 \leq \text{-}a) \wedge \\
&a = 3 \wedge \\
&1 \leq \text{-}1 + a \wedge \\
&\neg(2 + a \leq 6) \wedge \\
&\neg(\text{-}1 + a \leq 1) \wedge \\
&(a \neq 6 \vee \neg(b \leq 6))
\end{aligned}
$$

Given: a problem (*an interpolation problem*)
Generate: a formula (*an interpolant*) which is small

$$
\begin{aligned}
&\text{-}1 + a + \text{-}a = \text{-}1 \wedge \\
&\forall x(\neg(x \leq 5) \vee \text{-}6 + x \leq \text{-}1) \wedge \\
&\text{-}(\text{-}1 + \text{-}1 + a) = \text{-}1 \wedge \\
&\forall x((1 \leq x + \text{--}(\text{-}1 + a) \vee \neg(\text{-}1 \leq x))) \wedge \\
&(a \leq 6 \vee 1 \leq a + \text{-}1) \wedge \\
&\forall x(\neg(\text{-}1 \leq x) \vee \neg(x \leq \text{-}2)) \wedge \\
&\forall x(\text{-}1 \leq x + \text{-}a \vee \neg(\text{-}1 + a \leq x)) \wedge \\
&\forall x(\text{-}1 + x = 1 + \text{-}2 + x) \wedge \\
&\text{-}a + \text{-}1 + a = \text{-}1 \wedge \\
&\forall x(\neg(\text{--}(\text{-}1 + a) \leq x) \vee 1 \leq x + \text{-}1) \wedge \\
&\forall x((\neg(x \leq 4) \vee \text{-}5 + x \leq \text{-}1)) \wedge \\
&\forall x(x + \text{-}3 \leq \text{-}1 \vee \neg(x \leq 2)) \wedge \\
&\forall x(\neg(x \leq 3) \vee \text{-}4 + x \leq \text{-}1) \wedge \\
&\forall x(x + \text{-}a \leq \text{-}1 \vee \neg(x \leq \text{-}1 + a)) \wedge \\
&\forall x(\text{-}1 + x = \text{-}1 + \text{-}1 + a + \text{-}(\text{-}1 + a) + x) \wedge \\
&6 \leq b
\end{aligned}
$$

or

$$
\begin{aligned}
&\neg(a \leq 6) \wedge \\
&\text{-}a \leq \text{-}1 \wedge \\
&\neg(\text{-}1 \leq \text{-}a) \wedge \\
&a = 3 \wedge \\
&1 \leq \text{-}1 + a \wedge \\
&\neg(2 + a \leq 6) \wedge \\
&\neg(\text{-}1 + a \leq 1) \wedge \\
&(a \neq 6 \vee \neg(b \leq 6))
\end{aligned}
$$

Given: a problem (*an interpolation problem*)
Generate: a formula (*an interpolant*) which is small

$$
\begin{array}{l}
\text{-}1 + a + \text{-}a = \text{-}1 \wedge \\
\forall x(\neg(x \leq 5) \vee \text{-}6 + x \leq \text{-}1) \wedge \\
\text{-}(\text{-}1 + \text{-}1 + a) = \text{-}1 \wedge \\
\forall x((1 \leq x + \text{--}(\text{-}1 + a) \vee \neg(\text{-}1 \leq x))) \wedge \\
(a \leq 6 \vee 1 \leq a + \text{-}1) \wedge \\
\forall x(\neg(\text{-}1 \leq x) \vee \neg(x \leq \text{-}2)) \wedge \\
\forall x(\text{-}1 \leq x + \text{-}a \vee \neg(\text{-}1 + a \leq x)) \wedge \\
\forall x(\text{-}1 + x = 1 + \text{-}2 + x) \wedge \\
\text{-}a + \text{-}1 + a = \text{-}1 \wedge \\
\forall x(\neg(\text{--}(\text{-}1 + a) \leq x) \vee 1 \leq x + \text{-}1) \wedge \\
\forall x((\neg(x \leq 4) \vee \text{-}5 + x \leq \text{-}1)) \wedge \\
\forall x(x + \text{-}3 \leq \text{-}1 \vee \neg(x \leq 2)) \wedge \\
\forall x(\neg(x \leq 3) \vee \text{-}4 + x \leq \text{-}1) \wedge \\
\forall x(x + \text{-}a \leq \text{-}1 \vee \neg(x \leq \text{-}1 + a)) \wedge \\
\forall x(\text{-}1 + x = \text{-}1 + \text{-}1 + a + \text{-}(\text{-}1 + a) + x) \wedge \\
6 \leq b
\end{array}
$$

or

$$
\begin{array}{l}
\neg(a \leq 6) \wedge \\
\text{-}a \leq \text{-}1 \wedge \\
\neg(\text{-}1 \leq \text{-}a) \wedge \\
a = 3 \wedge \\
1 \leq \text{-}1 + a \wedge \\
\neg(2 + a \leq 6) \wedge \\
\neg(\text{-}1 + a \leq 1) \wedge \\
(a \neq 6 \vee \neg(b \leq 6))
\end{array}
$$

What is a good interpolant?

- logical strength [Jhala07, D'Silva09, McMillan08];
- small size [Kroening10, Brillout11, Griggio11].

How to Make Interpolants Smaller/Nicer?

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

How to Make Interpolants Smaller/Nicer?

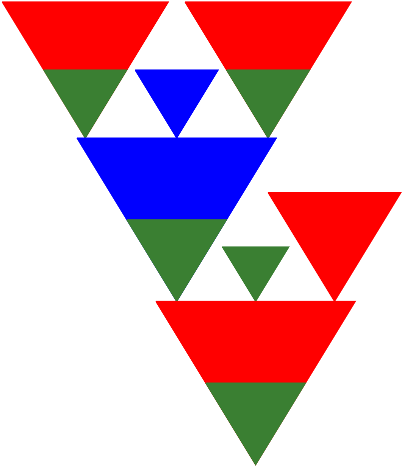
- in size;
- in weight;
- in the number of quantifiers;
- ...

Revised Interpolation Problem:

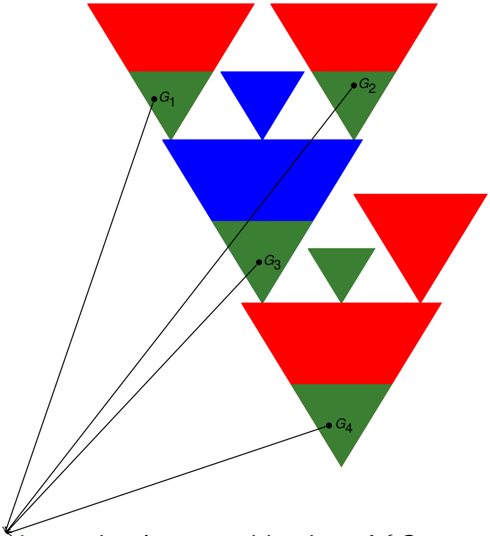
Given $\rightarrow R \rightarrow B$, find a green formula I:

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

Extracting Interpolants from Local Proofs



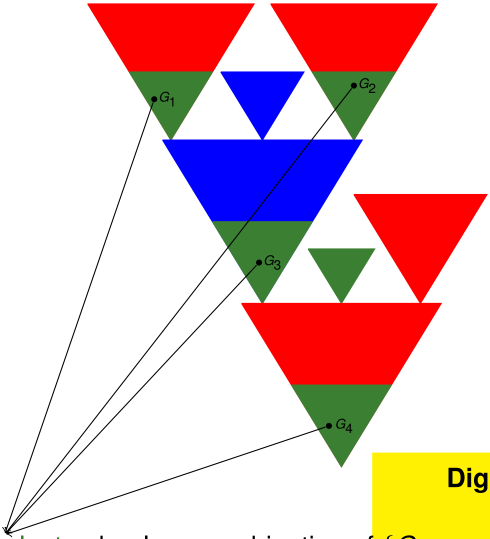
Extracting Interpolants from Local Proofs



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

[McMillan05, KV09]

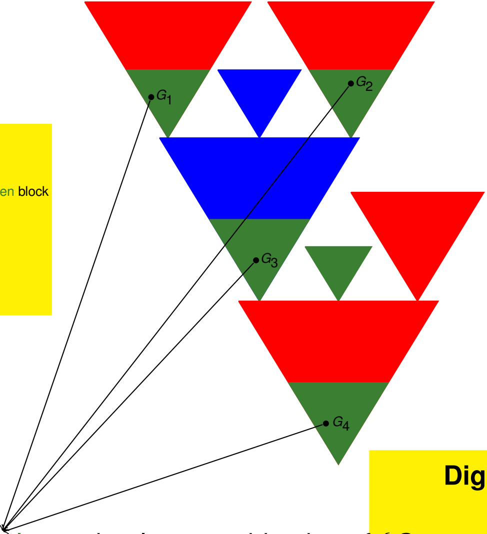
Extracting Interpolants from Local Proofs



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

Digest

Extracting Interpolants from Local Proofs



G is in the digest:

- comes from a red block

- followed by a blue or green block

Digest

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

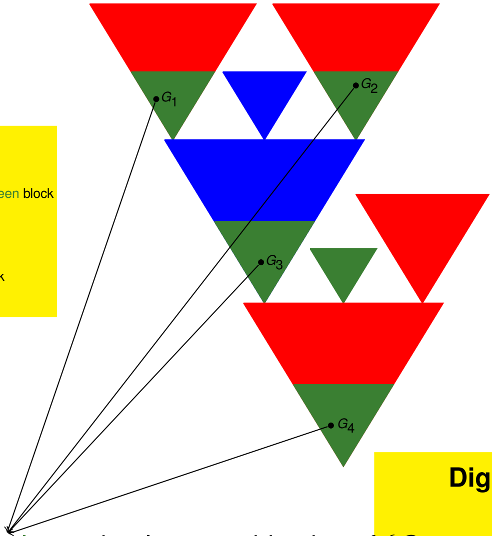
Extracting Interpolants from Local Proofs

G is in the digest:

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

or

- comes from a blue block
- followed by a red



Digest

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

How to Make Interpolants Smaller/Nicer?

Task: minimise interpolants = minimise digest

How to Make Interpolants Smaller/Nicer?

Task: minimise interpolants = minimise digest

Idea: Change the green areas of the local proof

How to Make Interpolants Smaller/Nicer?

Task: minimise interpolants = minimise digest

Idea: Change the green areas of the local proof

Slicing off formulas

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

How to Make Interpolants Smaller/Nicer?

Task: minimise interpolants = minimise digest

Idea: Change the green areas of the local proof

Slicing off formulas

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

If A is green: Green slicing

How to Make Interpolants Smaller/Nicer?

Task: minimise interpolants = minimise digest

Idea: Change the green areas of the local proof

Slicing off formulas

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

How to Make Interpolants Smaller/Nicer?

Task: minimise interpolants = minimise digest

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

Slicing off formulas

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

How to Make Interpolants Smaller/Nicer?

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

How to Make Interpolants Smaller/Nicer?

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

Digest: $\{G_4, G_7\}$

Reverse interpolant: $G_4 \to G_7$

How to Make Interpolants Smaller/Nicer?

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

How to Make Interpolants Smaller/Nicer?

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

Digest: $\{G_5, G_7\}$

Reverse interpolant: $G_5 \rightarrow G_7$

How to Make Interpolants Smaller/Nicer?

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

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

How to Make Interpolants Smaller/Nicer?

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

$$\cfrac{R_3 \qquad\qquad \cfrac{}{G_6}}{\cfrac{R_4}{\cfrac{G_7}{\bot}}}$$

Digest: $\{G_6, G_7\}$

Reverse interpolant: $G_6 \rightarrow G_7$

How to Make Interpolants Smaller/Nicer?

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

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

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

How to Make Interpolants Smaller/Nicer?

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

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

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

Digest: $\{G_6\}$

Reverse interpolant: $\neg G_6$

How to Make Interpolants Smaller/Nicer?

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

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

How to Make Interpolants Smaller/Nicer?

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

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

- There is no obvious logical relation between $G_4 \rightarrow G_7$ 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.

How to Make Interpolants Smaller/Nicer?

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

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

How to Make Interpolants Smaller/Nicer?

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

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

How to Make Interpolants Smaller/Nicer?

Solution:

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

How to Make Interpolants Smaller/Nicer?

Solution:

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

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

How to Make Interpolants Smaller/Nicer?

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.

How to Make Interpolants Smaller/Nicer?

Solution:

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

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

Some predicates on green 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;
- green(G): the trace of G contains only green formulas;
- digest(G): G belongs to the digest.

How to Make Interpolants Smaller/Nicer?

Solution:

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

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

$$\neg\mathrm{sliced}(G_1) \to \mathrm{Green}(G_1)$$
$$\mathrm{sliced}(G_1) \to \mathrm{red}(G_1)$$

Some predicates on green formulas:

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

How to Make Interpolants Smaller/Nicer?

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 green 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;
- green(G): the trace of G contains only green formulas;
- digest(G): G belongs to the digest.

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

How to Make Interpolants Smaller/Nicer?

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 green 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;
- green(G): the trace of G contains only green formulas;
- digest(G): G belongs to the digest.

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

How to Make Interpolants Smaller/Nicer?

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 green 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;
- green(G): the trace of G contains only green formulas;
- digest(G): G belongs to the digest.

$$\neg\text{sliced}(G_1) \rightarrow \text{Green}(G_1)$$
$$\text{sliced}(G_1) \rightarrow \text{red}(G_1)$$
$$\neg\text{sliced}(G_3) \rightarrow \text{Green}(G_3)$$
$$\text{sliced}(G_3) \rightarrow (\text{Green}(G_3) \leftrightarrow \text{Green}(G_1) \wedge \text{Green}(G_2))$$
$$\text{sliced}(G_3) \rightarrow (\text{red}(G_3) \leftrightarrow \text{red}(G_1) \vee \text{red}(G_2))$$
$$\text{sliced}(G_3) \rightarrow (\text{blue}(G_3) \leftrightarrow \text{blue}(G_1) \vee \text{blue}(G_2))$$
$$\text{digest}(G_1) \rightarrow \neg\text{sliced}(G_1)$$
$$\ldots$$

How to Make Interpolants Smaller/Nicer?

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)

How to Make Interpolants Smaller/Nicer?

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

How to Make Interpolants Smaller/Nicer?

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/green formula

  rc(G)/bc(G)

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

  bf(G)/rf(G)

How to Make Interpolants Smaller/Nicer?

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/green formula

  rc(G)/bc(G)

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

  bf(G)/rf(G)

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

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

How to Make Interpolants Smaller/Nicer?

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/green formula

  rc(G)/bc(G)

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

  bf(G)/rf(G)

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

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

$\neg\text{sliced}(G_3) \rightarrow \text{Green}(G_3)$

$\text{sliced}(G_3) \rightarrow (\text{Green}(G_3) \leftrightarrow \text{Green}(G_1) \wedge \text{Green}(G_2))$

$\text{sliced}(G_3) \rightarrow (\text{red}(G_3) \leftrightarrow \text{red}(G_1) \vee \text{red}(G_2))$

$\text{sliced}(G_3) \rightarrow (\text{blue}(G_3) \leftrightarrow \text{blue}(G_1) \vee \text{blue}(G_2))$

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

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

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

$\ldots$

How to Make Interpolants Smaller/Nicer?

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/green formula

  rc(G)/bc(G)

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

  bf(G)/rf(G)

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

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

$\neg\text{sliced}(G_3) \rightarrow \text{Green}(G_3)$

$\text{sliced}(G_3) \rightarrow (\text{Green}(G_3) \leftrightarrow \text{Green}(G_1) \wedge \text{Green}(G_2))$

$\text{sliced}(G_3) \rightarrow (\text{red}(G_3) \leftrightarrow \text{red}(G_1) \vee \text{red}(G_2))$

$\text{sliced}(G_3) \rightarrow (\text{blue}(G_3) \leftrightarrow \text{blue}(G_1) \vee \text{blue}(G_2))$

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

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

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

$\ldots$

How to Make Interpolants Smaller/Nicer?

Solution:

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

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

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

How to Make Interpolants Smaller/Nicer?

Solution:

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

How to Make Interpolants Smaller/Nicer?

Solution:

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

How to Make Interpolants Smaller/Nicer?

Solution:

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

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

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

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

How to Make Interpolants Smaller/Nicer?

Solution:

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

- 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;
    - minimal interpolants were generated for 2123 SMT examples.

Experiments with Minimising Interpolants

- 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;
    - minimal interpolants were generated for 2123 SMT examples.

Experiments with Minimising Interpolants

- More realistic benchmarks:
  - 4048 problems coming from CPAchecker;
  - we used Vampire to generate local proofs;
  - minimal interpolants were generated for 1903 CPAchecker examples:
    - for 296 examples the size of the interpolant has decreased by a factor of 5;
    - for 6 examples the size of the interpolant has decreased by a factor of 500.

Symbol Elimination and Interpolation

Summary: Invariant Generation, Interpolation, Symbol Elimination

Given the proof obligation $A \to B$:

1. Run a theorem prover and eliminate extra symbols;
2. Generate a (reverse) interpolant from a refutation;
3. Interpolant is a boolean combination of consequences of **symbol-eliminating inferences**.

Given a loop:

1. Express loop properties in a language containing extra symbols;
2. Every logical consequence of these properties is a valid loop property, but not an invariant;
3. Run a theorem prover for eliminating extra symbols;
4. Every derived formula in the language of the loop is a loop invariant;
5. Invariants are consequences of **symbol-eliminating inferences**.

Summary: Invariant Generation, Interpolation, Symbol Elimination

Given the proof obligation $A \rightarrow B$:

1. Run a theorem prover and eliminate extra symbols;
2. Generate a (reverse) interpolant from a refutation;
3. Interpolant is a boolean combination of consequences of **symbol-eliminating inferences**.

Given a loop:

1. Express loop properties in a language containing extra symbols;
2. Every logical consequence of these properties is a valid loop property, but not an invariant;
3. Run a theorem prover for eliminating extra symbols;
4. Every derived formula in the language of the loop is a loop invariant;
5. Invariants are consequences of **symbol-eliminating inferences**.

End of Session 4

Slides for session 4 ended here . . .