Interpolation

Seminar Slides

Albert-Ludwigs-Universität Freiburg

UNI FREIBURG

Betim Musa

27^{th} June 2015

Motivation

```
        program add(int a, int b) {
         var x,i : int;
ℓ0       assume(b ≥ 0);
ℓ1       x := a;
ℓ2       i := 0;
         while(i < b) {
ℓ3           x := x + 1;
ℓ4           i := i + 1;
         }
        assert (x == a + b);
```

Motivation

	`program add(int a, int b) {`
	`var x,i : int;`
ℓ_0	`assume(b ≥ 0);`
ℓ_1	`x := a;`
ℓ_2	`i := 0;`
	`while(i < b) {`
ℓ_3	`x := x + 1;`
ℓ_4	`i := i + 1;`
	`}`
ℓ_{err}	`assert (x != a + b);`

Prove correctness (CEGAR approach)

Idea: Show that all traces from ℓ_0 to ℓ_{err} are infeasible.

Motivation

```
program add(int a, int b) {
      var x,i : int;
ℓ0    assume(b ≥ 0);
ℓ1    x := a;
ℓ2    i := 0;
      while(i < b) {
ℓ3        x := x + 1;
ℓ4        i := i + 1;
      }
ℓerr  assert (x != a + b);
```

Prove correctness (CEGAR approach)

Idea: Show that all traces from ℓ_0 to ℓ_{err} are infeasible.

1. Choose an error trace τ.
2. Show that τ is infeasible.
3. Compute interpolants for τ.

Contents

Bit of history

- W. Craig (1957), Linear reasoning. A new form of the Herbrand-Gentzen theorem

Bit of history

- W. Craig (1957), Linear reasoning. A new form of the Herbrand-Gentzen theorem
- K. L. McMillan (2003), Interpolation and SAT-Based Model Checking

Bit of history

- W. Craig (1957), Linear reasoning. A new form of the Herbrand-Gentzen theorem
- K. L. McMillan (2003), Interpolation and SAT-Based Model Checking
- A. Cimatti et al. (2007), Efficient Interpolant Generation in SMT

Contents

Interpolant

An interpolant I for the unsatisfiable pair of formulae A, B has the following properties:

Interpolant

An interpolant I for the unsatisfiable pair of formulae A, B has the following properties:

- $A \models I$

Interpolant

An interpolant I for the unsatisfiable pair of formulae A, B has the following properties:

- $A \models I$
- $I \wedge B$ is unsatisfiable

Interpolant

An interpolant I for the unsatisfiable pair of formulae A, B has the following properties:

- $A \models I$
- $I \wedge B$ is unsatisfiable
- $I \preceq A$ and $I \preceq B$ (symbol condition)

Contents

UNI FREIBURG

Ingredients

1. a pair of unsatisfiable formulae A, B
2. a resolution proof of their unsatisfiability

UNI FREIBURG

Prove unsatisfiability of $\overbrace{P \land (\neg P \lor R)}^{A} \land \overbrace{\neg R}^{B}$

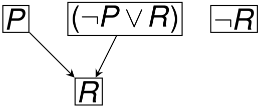
Prove unsatisfiability of $\overbrace{P \land (\neg P \lor R)}^{A} \land \overbrace{\neg R}^{B}$

$\boxed{P}$ $\boxed{(\neg P \lor R)}$ $\boxed{\neg R}$

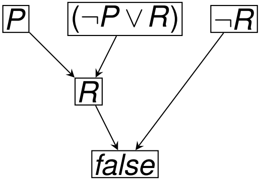
Interpolation in Propositional Logic

Resolution

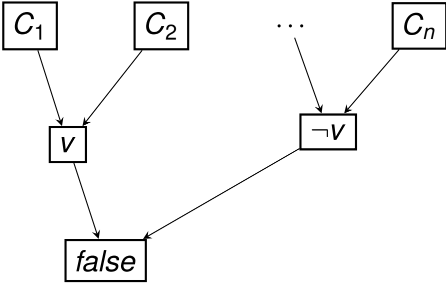
Prove unsatisfiability of $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$



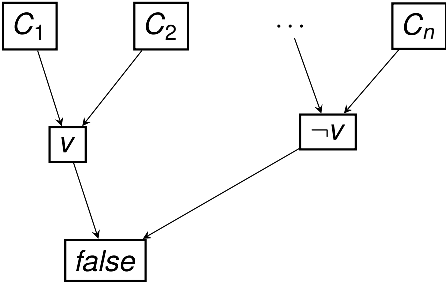
Prove unsatisfiability of $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$



Given: unsatisfiable formulae A, B and a proof of unsatisfiability.



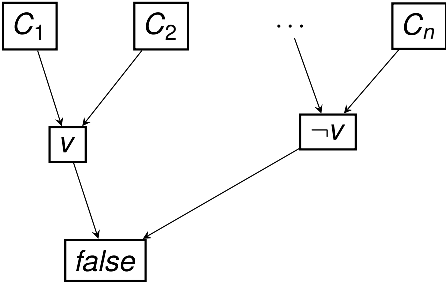
Given: unsatisfiable formulae A, B and a proof of unsatisfiability. For every vertex v of the proof define the interpolant $ITP(v)$ as follows:



Interpolation in Propositional Logic

Given: unsatisfiable formulae A, B and a proof of unsatisfiability. For every vertex v of the proof define the interpolant $ITP(v)$ as follows:

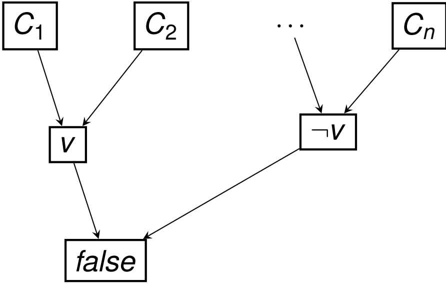
- if v is an input node



Interpolation in Propositional Logic

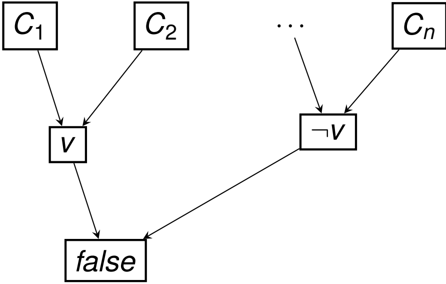
Given: unsatisfiable formulae A, B and a proof of unsatisfiability. For every vertex v of the proof define the interpolant $ITP(v)$ as follows:

- if v is an input node
  1. if $v \in A$ then **ITP**(**v**) = **global_literals**(**v**)
  2. else **ITP**(**v**) = **true**



Given: unsatisfiable formulae A, B and a proof of unsatisfiability. For every vertex v of the proof define the interpolant $ITP(v)$ as follows:

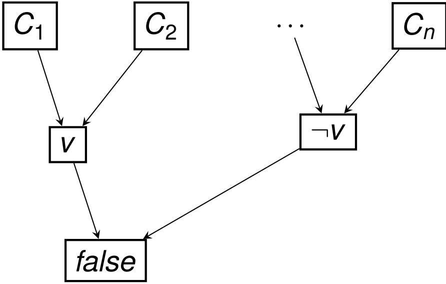
- if v is an input node
  1. if $v \in A$ then **ITP**(**v**) = **global_literals**(**v**)
  2. else **ITP**(**v**) = **true**
- else v must have two predecessors v_1, v_2 and p_v is the pivot variable.



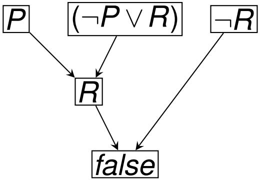
Interpolation in Propositional Logic

Given: unsatisfiable formulae A, B and a proof of unsatisfiability. For every vertex v of the proof define the interpolant $ITP(v)$ as follows:

- if v is an input node
  1. if $v \in A$ then **ITP(v) = global_literals(v)**
  2. else **ITP(v) = true**
- else v must have two predecessors v_1, v_2 and p_v is the pivot variable.
  1. if p_v is *local* to A, then **ITP(v) = ITP($\mathbf{v_1}$) $\vee$ ITP($\mathbf{v_2}$)**
  2. else **ITP(v) = ITP($\mathbf{v_1}$) $\wedge$ ITP($\mathbf{v_2}$)**



Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$

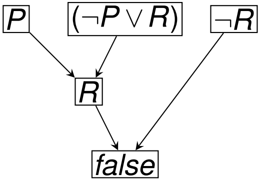


Interpolation in Propositional Logic

Example

Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$

- $ITP(P) = FALSE$

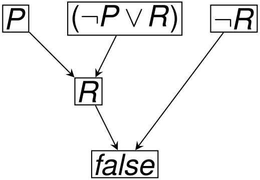


Interpolation in Propositional Logic

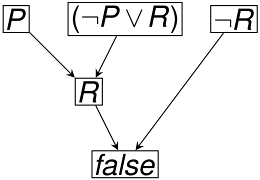
Example

Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$

- $ITP(P) = FALSE$
- $ITP(\neg P \vee R) = R$



Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$

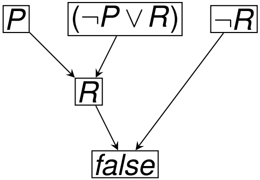


- $ITP(P) = FALSE$
- $ITP(\neg P \vee R) = R$
- $ITP(\neg R) = TRUE$

Interpolation in Propositional Logic

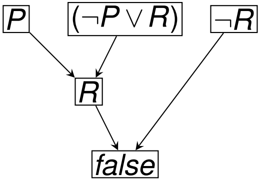
Example

Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$



- $ITP(P) = FALSE$
- $ITP(\neg P \vee R) = R$
- $ITP(\neg R) = TRUE$
- $ITP(R) =$
  $ITP(P) \vee ITP(\neg P \vee R)$

Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$

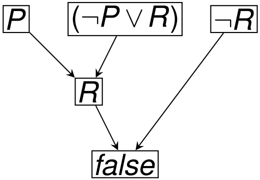


- $ITP(P) = FALSE$
- $ITP(\neg P \vee R) = R$
- $ITP(\neg R) = TRUE$
- $ITP(R) =$
  $ITP(P) \vee ITP(\neg P \vee R)$
- $ITP(false) =$
  $ITP(R) \wedge ITP(\neg R)$

Interpolation in Propositional Logic

Example

Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$

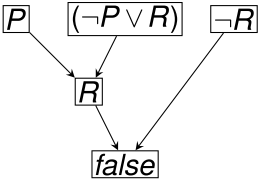


- $\color{blue}{ITP(P) = FALSE}$
- $\color{blue}{ITP(\neg P \vee R) = R}$
- $\color{blue}{ITP(\neg R) = TRUE}$
- $ITP(R) =$
  $ITP(P) \vee ITP(\neg P \vee R)$
- $ITP(false) =$
  $ITP(R) \wedge ITP(\neg R)$

Interpolation in Propositional Logic

Example

Formula: $\overbrace{P \wedge (\neg P \vee R)}^{A} \wedge \overbrace{\neg R}^{B}$



- $\color{blue}{ITP(P) = FALSE}$
- $\color{blue}{ITP(\neg P \vee R) = R}$
- $\color{blue}{ITP(\neg R) = TRUE}$
- $ITP(R) =$ $ITP(P) \vee ITP(\neg P \vee R)$
- $ITP(false) =$ $ITP(R) \wedge ITP(\neg R)$

The resulting interpolant:
$ITP(false) =$
$(FALSE \vee R) \wedge TRUE = R$

Contents

Interesting theories in practice

Interesting theories in practice

- Linear Integer Arithmetic
- Presburger Arithmetic
- Equality Theory with Uninterpreted Functions
- Theory of Arrays
- Theory of Lists

Interpolation in First-Order Logic

Overview

Interesting theories in practice

- Linear Integer Arithmetic
- Presburger Arithmetic
- Equality Theory with Uninterpreted Functions
- Theory of Arrays
- Theory of Lists

Requirements

- SAT-Solver (lazy)
- a theory solver (T-Solver)

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'
2. Assign a truth value to some variable (SAT-Solver)

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'
2. Assign a truth value to some variable (SAT-Solver)
3. Check the current assignment for consistency (T-solver)

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'
2. Assign a truth value to some variable (SAT-Solver)
3. Check the current assignment for consistency (T-solver)
   - inconsistent, T-solver returns a conflict set η, add its negation as a T-lemma

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'
2. Assign a truth value to some variable (SAT-Solver)
3. Check the current assignment for consistency (T-solver)
   - inconsistent, T-solver returns a conflict set η, add its negation as a T-lemma
   - consistent, go on with assignment of next variable

SMT: Satisfiability Modulo Theory

Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'
2. Assign a truth value to some variable (SAT-Solver)
3. Check the current assignment for consistency (T-solver)
   - inconsistent, T-solver returns a conflict set η, add its negation as a T-lemma
   - consistent, go on with assignment of next variable
4. If a truth value is assigned to all variables $\Longrightarrow$ SAT

SMT: Satisfiability Modulo Theory

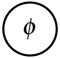
Is a given FOL-formula ϕ satisfiable with respect to the theory T?

Procedure (lazy approach)

1. Encode as a boolean formula ϕ'
2. Assign a truth value to some variable (SAT-Solver)
3. Check the current assignment for consistency (T-solver)
   - inconsistent, T-solver returns a conflict set η, add its negation as a T-lemma
   - consistent, go on with assignment of next variable
4. If a truth value is assigned to all variables $\Longrightarrow$ SAT
5. If no assignment left $\Longrightarrow$ UNSAT

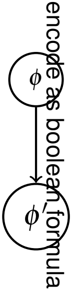
SMT-SAT (lazy approach)

Illustration



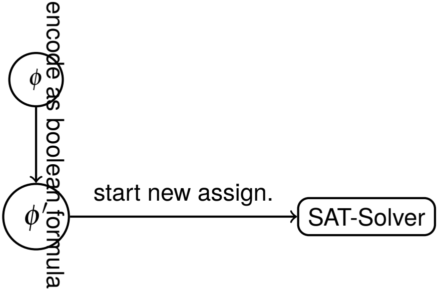
SMT-SAT (lazy approach)

Illustration



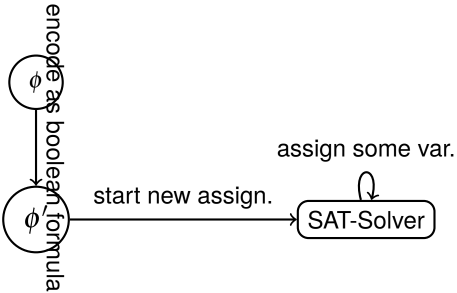
SMT-SAT (lazy approach)

Illustration



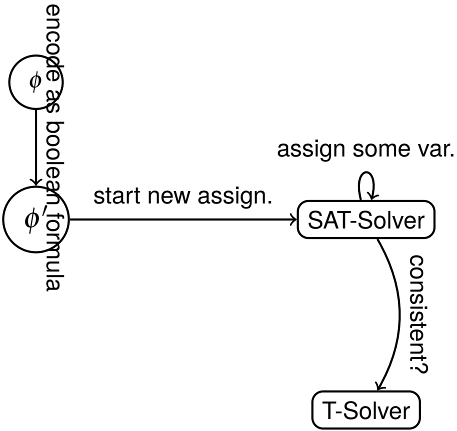
SMT-SAT (lazy approach)

Illustration



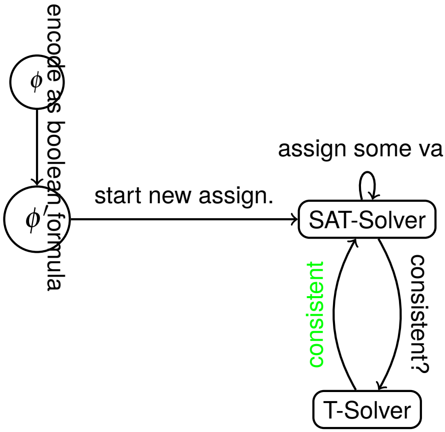
SMT-SAT (lazy approach)

Illustration



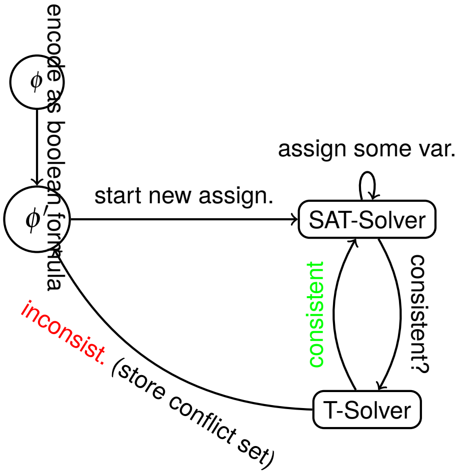
SMT-SAT (lazy approach)

Illustration



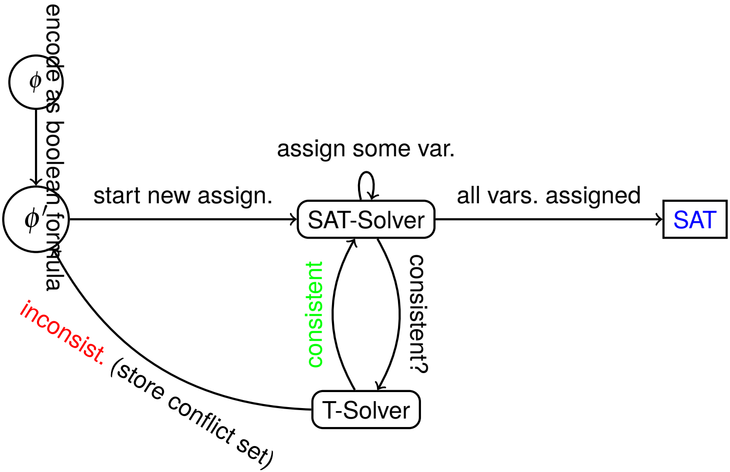
SMT-SAT (lazy approach)

Illustration



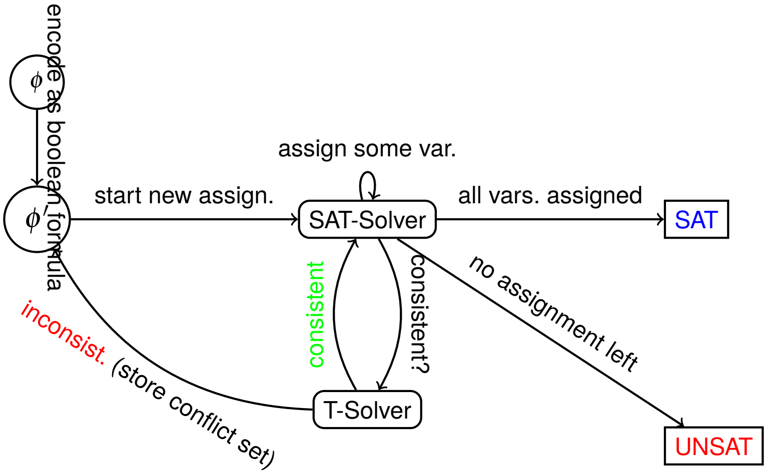
SMT-SAT (lazy approach)

Illustration



SMT-SAT (lazy approach)

Illustration



Given two formulae $c_1 = \neg x_1 \vee x_2 \vee \neg x_3$ and $c_2 = x_2 \vee x_3$

- $c_1 \downarrow c_2 = x_2 \vee \neg x_3$

Given two formulae $c_1 = \neg x_1 \vee x_2 \vee \neg x_3$ and $c_2 = x_2 \vee x_3$

- $c_1 \downarrow c_2 = x_2 \vee \neg x_3$
- $c_1 \setminus c_2 = \neg x_1$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:
  - if $C \in A$, then $I_C \equiv C \downarrow B$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - \mathit{lemma}\ \neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:
  - if $C \in A$, then $I_C \equiv C \downarrow B$
  - if $C \in B$, then $I_C \equiv \top$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:
  - if $C \in A$, then $I_C \equiv C \downarrow B$
  - if $C \in B$, then $I_C \equiv \top$
- For every inner node C of $\mathscr{P}$ obtained by resolution from $C_1 = p \vee \phi_1, C_2 = \neg p \vee \phi_2$,

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:
  - if $C \in A$, then $I_C \equiv C \downarrow B$
  - if $C \in B$, then $I_C \equiv \top$
- For every inner node C of $\mathscr{P}$ obtained by resolution from $C_1 = p \vee \phi_1, C_2 = \neg p \vee \phi_2$,
  - if $p \notin B$, then $I_C \equiv I_{C_1} \vee I_{C_2}$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:
  - if $C \in A$, then $I_C \equiv C \downarrow B$
  - if $C \in B$, then $I_C \equiv \top$
- For every inner node C of $\mathscr{P}$ obtained by resolution from $C_1 = p \vee \phi_1, C_2 = \neg p \vee \phi_2$,
  - if $p \notin B$, then $I_C \equiv I_{C_1} \vee I_{C_2}$
  - else $I_C \equiv I_{C_1} \wedge I_{C_2}$

Interpolation in SMT

Generate an interpolant for the conjunction $A \wedge B$.

- Compute a proof of unsatisfiability $\mathscr{P}$ for $A \wedge B$
- For every $T - lemma$ $\neg\eta$ in $\mathscr{P}$ compute an interpolant $I_{\neg\eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause C in $\mathscr{P}$:
  - if $C \in A$, then $I_C \equiv C \downarrow B$
  - if $C \in B$, then $I_C \equiv \top$
- For every inner node C of $\mathscr{P}$ obtained by resolution from $C_1 = p \vee \phi_1, C_2 = \neg p \vee \phi_2$,
  - if $p \notin B$, then $I_C \equiv I_{C_1} \vee I_{C_2}$
  - else $I_C \equiv I_{C_1} \wedge I_{C_2}$
- Output the interpolant at the root node, namely $I_\perp$

Conclusion

Interpolation

- an important technique in software verification

Conclusion

Interpolation

- an important technique in software verification
- available for many relevant theories (e.g. LIA, Equality with UF, Arrays, Lists)

Conclusion

Interpolation

- an important technique in software verification
- available for many relevant theories (e.g. LIA, Equality with UF, Arrays, Lists)
- research in progress for other theories

Summary

What is interpolation?

- automatically generalize formulae and preserve relevant parts

Summary

What is interpolation?

- automatically generalize formulae and preserve relevant parts
- interpolant (Craig's definition)

Summary

What is interpolation?

- automatically generalize formulae and preserve relevant parts
- interpolant (Craig's definition)

Summary

What is interpolation?

- automatically generalize formulae and preserve relevant parts
- interpolant (Craig's definition)

How does it work?

- Propositional Logic: resolution proof

Summary

What is interpolation?

- automatically generalize formulae and preserve relevant parts
- interpolant (Craig's definition)

How does it work?

- Propositional Logic: resolution proof
- First-Order Logic: Resolution proof, Theory interpolation

Future work

A theory where no efficient interpolation algorithm exists

- theory of non-linear integer arithmetic (e.g. $x^2 + y^2 = 1$)

References I

A. Cimatti, A. Griggio, R. Sebastiani.
Efficient Interpolant Generation in SMT.

K.L.McMillan.
Interpolation and SAT-based Model Checking.

Philipp Rümmer
Craig Interpolation in SAT and SMT
http://satsmt2014.forsyte.at/files/2014/01/interpolation_philipp.pdf

D. Kroening, G. Weissenbacher .
Lifting Propositional Interpolants to the Word-Level.

References II

Wikipedia
Satisfiability Modulo Theories.
https://en.wikipedia.org/wiki/Satisfiability_Modulo_Theories