Interpolation
Seminar Slides

Betim Musa
27th June 2015
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);
}

Prove correctness (CEGAR approach)
Idea: Show that all traces from ℓ₀ to ℓ₄ are infeasible.

Choose an error trace τ.
Show that τ is infeasible.
Compute interpolants for τ.
Motivation

program add(int a, int b) {
    var x, i : int;

    ℓ₀ assume(b ≥ 0);
    ℓ₁ x := a;
    ℓ₂ i := 0;
    ℓ₃ while(i < b) {
        ℓ₃ x := x + 1;
        ℓ₄ i := i + 1;
    }
    ℓ₆ assert (x != a + b);

Prove correctness (CEGAR approach)

Idea: Show that all traces from ℓ₀ to ℓ₆ 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;
    }
    ℓ₆ err assert (x != a + b);
}

Prove correctness (CEGAR approach)

Idea: Show that all traces from ℓ₀ to ℓ₆ are infeasible.

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

A bit of history

Interpolation
  What is an interpolant?
  Interpolation in Propositional Logic
  Interpolation in First-Order Logic

Conclusion

References
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
A bit of history

Interpolation

What is an interpolant?
Interpolation in Propositional Logic
Interpolation in First-Order Logic

Conclusion

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

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

- $A \models I$
- $I \land B$ is unsatisfiable
An interpolant $I$ for the unsatisfiable pair of formulae $A, B$ has the following properties:

- $A \models I$
- $I \land B$ is unsatisfiable
- $I \preceq A$ and $I \preceq B$ (symbol condition)
A bit of history

Interpolation
  What is an interpolant?
  Interpolation in Propositional Logic
  Interpolation in First-Order Logic

Conclusion

References
## Interpolation in Propositional Logic

### Ingredients

1. A pair of unsatisfiable formulae $A, B$
2. A resolution proof of their unsatisfiability
Interpolation in Propositional Logic

Resolution

Prove unsatisfiability of

\[ A = P \land (\lnot P \lor R) \land \lnot R \]

\[ B = \lnot P \lor R \land \lnot R \]
Interpolation in Propositional Logic

Resolution

Prove unsatisfiability of

\[
A = \overbrace{P \land (\neg P \lor R)} \land \overbrace{\neg R} 
\]

\[
P \quad (\neg P \lor R) \quad \neg R
\]
Prove unsatisfiability of $P \land (\neg P \lor R) \land \neg R$.
Prove unsatisfiability of $P \land (\neg P \lor R) \land \neg R$.
Interpolation in Propositional Logic

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

For every vertex $v$ of the proof define the interpolant $\text{ITP}(v)$ as follows:

1. If $v$ is an input node
   - If $v \in A$ then $\text{ITP}(v) = \text{global} \_ \text{literals}(v)$
   - Else $\text{ITP}(v) = \text{true}$

2. Else $v$ must have two predecessors $v_1, v_2$ and $p_v$ is the pivot variable.
   - If $p_v$ is local to $A$, then $\text{ITP}(v) = \text{ITP}(v_1) \lor \text{ITP}(v_2)$
   - Else $\text{ITP}(v) = \text{ITP}(v_1) \land \text{ITP}(v_2)$
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:

$$
\begin{align*}
ITP(v) &= \text{global literals } v \\
\text{else if } p_v \text{ is local to } A, &\quad \text{then } ITP(v) = ITP(v_1) \lor ITP(v_2) \\
\text{else, } v \text{ must have two predecessors } v_1, v_2 &\quad \text{and } p_v \text{ is the pivot variable.}
\end{align*}
$$

$$
\begin{align*}
ITP(v_1) &= ITP(v) \\
ITP(v_2) &= ITP(v)
\end{align*}
$$
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
  - if \( p_v \) is local to \( A \), then
    \[
    ITP(v) = ITP(v_1) \lor ITP(v_2)
    \]
  - else
    \[
    ITP(v) = ITP(v_1) \land ITP(v_2)
    \]
- if \( v \) has two predecessors \( v_1, v_2 \) and \( p_v \) is the pivot variable.
  - if \( p_v \) is local to \( A \), then
    \[
    ITP(v) = ITP(v_1) \lor ITP(v_2)
    \]
  - else
    \[
    ITP(v) = ITP(v_1) \land ITP(v_2)
    \]
Interpolation in Propositional Logic

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

1. If $v$ is an input node
   - If $v \in A$ then $ITP(v) = \text{global_literals}(v)$
   - Else $ITP(v) = \text{true}$
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 $ITP(v) = \text{global\_literals}(v)$
  2. else $ITP(v) = \text{true}$

- 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 $ITP(v) = ITP(v_1) \lor ITP(v_2)$
  2. else $ITP(v) = ITP(v_1) \land ITP(v_2)$
Interpolation in Propositional Logic

Example

Formula: $P \land (\neg P \lor R) \land \neg R$

The resulting interpolant:

$\text{ITP}(\text{false}) = (\text{FALSE} \lor R) \land \text{TRUE} = R$
Interpolation in Propositional Logic

Example

Formula:

\[ P \land (\neg P \lor R) \land \neg R \]

- \(\text{ITP}(P) = \text{FALSE}\)

- \(\text{ITP}(\neg P \lor R) = R\)

- \(\text{ITP}(\neg R) = \text{TRUE}\)

- \(\text{ITP}(\text{false}) = (\text{FALSE} \lor R) \land \text{TRUE} = R\)
Interpolation in Propositional Logic

Example

Formula: $P \land (\neg P \lor R) \land \neg R$

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

The resulting interpolant:

$ITP(false) = (FALSE \lor R) \land TRUE = R$
Interpolation in Propositional Logic

Example

Formula: \( P \land (\neg P \lor R) \land \neg R \)

- \( ITP(P) = FALSE \)
- \( ITP(\neg P \lor R) = R \)
- \( ITP(\neg R) = TRUE \)
Interpolation in Propositional Logic

Example

Formula: \( P \land (\neg P \lor R) \land \neg R \)

- \( ITP(P) = FALSE \)
- \( ITP(\neg P \lor R) = R \)
- \( ITP(\neg R) = TRUE \)
- \( ITP(R) = ITP(P) \lor ITP(\neg P \lor R) \)
Interpolation in Propositional Logic

Example

Formula: $P \land (\neg P \lor R) \land \neg R$

- $ITP(P) = \text{FALSE}$
- $ITP(\neg P \lor R) = R$
- $ITP(\neg R) = \text{TRUE}$
- $ITP(R) = ITP(P) \lor ITP(\neg P \lor R)$
- $ITP(\text{false}) = ITP(R) \land ITP(\neg R)$
Interpolation in Propositional Logic

Example

Formula: $P \land (\neg P \lor R) \land \neg R$

- $\text{ITP}(P) = \text{FALSE}$
- $\text{ITP}(\neg P \lor R) = R$
- $\text{ITP}(\neg R) = \text{TRUE}$
- $\text{ITP}(R) = \text{ITP}(P) \lor \text{ITP}(\neg P \lor R)$
- $\text{ITP}(\text{false}) = \text{ITP}(R) \land \text{ITP}(\neg R)$
Interpolation in Propositional Logic

Example

Formula: $P \land (\neg P \lor R) \land \neg R$

- $\text{ITP}(P) = \text{FALSE}$
- $\text{ITP}(\neg P \lor R) = R$
- $\text{ITP}(\neg R) = \text{TRUE}$
- $\text{ITP}(R) = \text{ITP}(P) \lor \text{ITP}(\neg P \lor R)$
- $\text{ITP}(\text{false}) = \text{ITP}(R) \land \text{ITP}(\neg R)$

The resulting interpolant:
$\text{ITP}(\text{false}) = (\text{FALSE} \lor R) \land \text{TRUE} = R$
Contents

A bit of history

Interpolation
  What is an interpolant?
  Interpolation in Propositional Logic
  Interpolation in First-Order Logic

Conclusion

References
Interesting theories in practice
Interesting theories in practice

- Linear Integer Arithmetic
- Presburger Arithmetic
- Equality Theory with Uninterpreted Functions
- Theory of Arrays
- Theory of Lists
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 $\phi$ satisfiable with respect to the theory $T$?
SMT: Satisfiability Modulo Theory

Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
2. Assign a truth value to some variable (SAT-Solver)
SMT: Satisfiability Modulo Theory

Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
2. Assign a truth value to some variable (SAT-Solver)
3. Check the current assignment for consistency ($T$-solver)
Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
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 $\eta$, add its negation as a $T$-lemma
4. If a truth value is assigned to all variables $\Rightarrow$ SAT
5. If no assignment left $\Rightarrow$ UNSAT
SMT: Satisfiability Modulo Theory

Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
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 $\eta$, 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 $\Rightarrow$ SAT
5. If no assignment left $\Rightarrow$ UNSAT

27th June 2015 
Betim Musa – Interpolation
SMT: Satisfiability Modulo Theory

Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
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 $\eta$, 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 $\iff$ SAT
SMT: Satisfiability Modulo Theory

Is a given FOL-formula $\phi$ satisfiable with respect to the theory $T$?

Procedure (lazy approach)

1. Encode as a boolean formula $\phi'$
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 $\eta$, 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 $\iff$ SAT
5. If no assignment left $\iff$ UNSAT
SMT-SAT (lazy approach)

Illustration

\[ \phi \]
SMT-SAT (lazy approach)

Illustration

- Encode as boolean formula

$\phi$
SMT-SAT (lazy approach)

Illustration

\[ \phi \] encode as boolean formula

\[ \phi \] start new assign.

SAT-Solver
SMT-SAT (lazy approach)

Illustration

encode as boolean formula

\( \phi \)

\( \phi' \)

start new assign.

assign some var.

\( \phi \)

SAT-Solver

inconsist. (store conflict set)

consistent

all vars. assigned

no assignment left
SMT-SAT (lazy approach)

Illustration

1. Encode as boolean formula
2. Start new assign.
3. Assign some var.
4. SAT-Solver
5. Consistent?
6. T-Solver
7. No assignment left
8. Inconsistent (store conflict set)
9. All vars. assigned
SMT-SAT (lazy approach)

Illustration

\[ \phi \]

- Start new assignment
- Assign some variable
- Encode as boolean formula
- SAT-Solver consistent?
- T-Solver consistent?
SMT-SAT (lazy approach)

Illustration

- encode as boolean formula
- start new assign.
- assign some var.
- consistent?
- inconsistent (store conflict set)

SAT-Solver

T-Solver
SMT-SAT (lazy approach)

Illustration

\( \phi \)

- **Encode as boolean formula**
- **Start new assign.**
- **Assign some var.**
- **SAT-Solver**
- **T-Solver**
- **SAT**

- **Consistent?**
- **Consistent**
- **Inconsistent** (store conflict set)
- **All vars. assigned**
- **No assignment left**
SMT-SAT (lazy approach)

Illustration

1. Encode as boolean formula
2. Start new assignment
3. Assign some variable
4. Check consistency
5. If consistent, continue; otherwise, store conflict set and backtrack
6. If all variables are assigned and consistent, SAT solver returns SAT
7. If no assignment left, T-solver returns UNSAT
Given two formulae $c_1 = \neg x_1 \lor x_2 \lor \neg x_3$ and $c_2 = x_2 \lor x_3$

\[ c_1 \downarrow c_2 = x_2 \lor \neg x_3 \]
Given two formulae $c_1 = \lnot x_1 \lor x_2 \lor \lnot x_3$ and $c_2 = x_2 \lor x_3$

- $c_1 \downarrow c_2 = x_2 \lor \lnot x_3$
- $c_1 \setminus c_2 = \lnot x_1$
Interpolation in SMT

Generate an interpolant for the conjunction $A \land B$. 
Interpolation in SMT

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

Compute a proof of unsatisfiability $\mathcal{P}$ for $A \land B$. 
Interpolation in SMT

Generate an interpolant for the conjunction $A \land B$.
- Compute a proof of unsatisfiability $\mathcal{P}$ for $A \land B$
- For every $T - lemma$ $\neg \eta$ in $\mathcal{P}$ compute an interpolant $I_{\neg \eta}$ for $(\eta \setminus B, \eta \downarrow B)$
Interpolation in SMT

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

- Compute a proof of unsatisfiability $\mathcal{P}$ for $A \land B$
- For every $T$–lemma $\neg \eta$ in $\mathcal{P}$ compute an interpolant $I_{\neg \eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause $C$ in $\mathcal{P}$:
Generate an interpolant for the conjunction $A \land B$.

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

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

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

- Compute a proof of unsatisfiability $\mathcal{P}$ for $A \land B$
- For every $T$ – lemma $\neg \eta$ in $\mathcal{P}$ compute an interpolant $I_{\neg \eta}$ for $(\eta \setminus B, \eta \downarrow B)$
- For every input clause $C$ in $\mathcal{P}$:
  - 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 $\mathcal{P}$ obtained by resolution from $C_1 = p \lor \phi_1, C_2 = \neg p \lor \phi_2$, output the interpolant at the root node, namely $I_{\bot}$
Generate an interpolant for the conjunction $A \land B$.

- Compute a proof of unsatisfiability $\mathcal{P}$ for $A \land B$

- For every $T$-lemma $\neg \eta$ in $\mathcal{P}$ compute an interpolant $I_{\neg \eta}$ for $(\eta \setminus B, \eta \downarrow B)$

- For every input clause $C$ in $\mathcal{P}$:
  - 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 $\mathcal{P}$ obtained by resolution from $C_1 = p \lor \phi_1, C_2 = \neg p \lor \phi_2$,
  - if $p \notin B$, then $I_C \equiv I_{C_1} \lor I_{C_2}$
Generate an interpolant for the conjunction $A \land B$.

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

Output the interpolant at the root node, namely $I_{\bot}$. 

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

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

- automatically generalize formulae and preserve relevant parts
## What is interpolation?

- automatically generalize formulae and preserve relevant parts
- interpolant (Craig’s definition)
### 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
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. $x^2 + y^2 = 1$)
A. Cimatti, A. Griggio, R. Sebastiani. Efficient Interpolant Generation in SMT.


Philipp Rümmer Craig Interpolation in SAT and SMT

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

Satisfiability Modulo Theories.