Decision Procedures

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

Summer 2012

Craig Interpolation

Given an unsatisfiable formula of the form:

$$F \wedge G$$

Can we find a "smaller" formula that explains the conflict?

I.e., a formula implied by F that is inconsistent with G?

Under certain conditions, there is an interpolant I with

- $F \Rightarrow I$.
- $I \wedge G$ is unsatisfiable.
- I contains only symbols common to F and G.

Craig Interpolation

A craig interpolant I for an unsatisfiable formula $F \land G$ is

- $F \Rightarrow I$.
- $I \land G$ is unsatisfiable.
- I contains only symbols common to F and G.

Craig interpolants exists in many theories and fragments:

- First-order logic.
- Quantifier-free FOL.
- Quantifier-free fragment of T_{E}.
- Quantifier-free fragment of $T_{\mathbb{Q}}$.
- Quantifier-free fragment of $\widehat{T_{\mathbb{Z}}}$ (augmented with divisibility).

However, QF fragment of $T_{\mathbb{Z}}$ does not allow Craig interpolation.

Program correctness

Consider this path through LinearSearch:

Single Static Assignment (SSA) replaces assignments by assumes:

$@\mathrm{pre}\ 0 \leq \ell \wedge u < \lvert a \rvert$	$@\mathrm{pre}\ 0 \leq \ell \wedge u < \lvert a \rvert$
$i := \ell$	$\texttt{assume}\ i_1 = \ell$
$\texttt{assume}\ i \leq u$	$\texttt{assume}\ i_1 \leq u$
$\texttt{assume}\ a[i] \neq e$	$\texttt{assume}\ a[i_1] \neq e$
$i := i + 1$	$\texttt{assume}\ i_2 = i_1 + 1$
$\texttt{assume}\ i \leq u$	$\texttt{assume}\ i_2 \leq u$
$@\ 0 \leq i \wedge i < \lvert a \rvert$	$@\ 0 \leq i_2 \wedge i_2 < \lvert a \rvert$

If program contains only assumes, the VC looks like

$$VC : P \rightarrow (F_1 \rightarrow (F_2 \rightarrow (F_3 \rightarrow \ldots (F_n \rightarrow Q) \ldots)))$$

Using $\neg(F \rightarrow G) \Leftrightarrow F \land \neg G$ compute negation:

$$\neg VC : P \land F_1 \land F_2 \land F_3 \land \cdots \land F_n \land Q$$

If verification condition is valid $\neg VC$ is unsatisfiable. We can compute interpolants for any program point, e.g. for

$$P \land F_1 \land F_2 \land F_3 \land \cdots \land F_n \land \neg Q$$

Verification Condition and interpolants

Consider the path through LinearSearch:

$$@\text{pre } 0 \leq \ell \land u < |a|$$

`assume` $i_1 = \ell$

`assume` $i_1 \leq u$

`assume` $a[i_1] \neq e$

`assume` $i_2 = i_1 + 1$

`assume` $i_2 \leq u$

$@\ 0 \leq i_2 \land i_2 < |a|$

The negated VC is unsatisfiable:

$$0 \leq \ell \land u < |a| \land i_1 = \ell$$
$$\land\ i_1 \leq u \land a[i_1] \neq e \land i_2 = i_1 + 1$$
$$\land\ i_2 \leq u \land (0 > i_2 \lor i_2 \geq |a|)$$

The interpolant I for the red and blue part is

$$i_1 \geq 0 \land u < |a|$$

This is actually the loop invariant needed to prove the assertion.

Computing Interpolants

Suppose $F_1 \wedge \ldots \wedge F_m \wedge G_1 \wedge \ldots \wedge G_n$ is unsat.
How can we compute an interpolant?

- The algorithm is dependent on the theory and the fragment.
- We will show an algorithm for
  - Quantifier-free conjunctive fragment of T_{E}.
  - Quantifier-free conjunctive fragment of $T_{\mathbb{Q}}$.

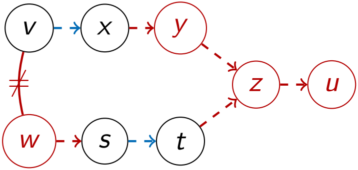
$$F_1 \wedge \cdots \wedge F_m \wedge G_1 \wedge \cdots \wedge G_n \text{ is unsat.}$$

Let us first consider the case without function symbols.
The congruence closure algorithm returns unsat. Hence,

- there is a disequality $v \neq w$ and
- v,w have the same representative.

Example:

$$v \neq w \wedge x = y \wedge y = z \wedge z = u \wedge w = s \wedge t = z \wedge s = t \wedge v = x$$



The Interpolant "summarizes" the red edges: $I : v \neq s \wedge x = t$

Computing Interpolants for T_E

Given conjunctive formula:

$$F_1 \land \cdots \land F_n \land G_1 \land \cdots \land G_m$$

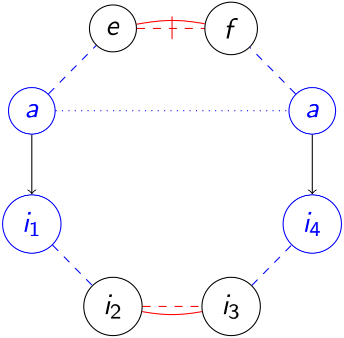
The following algorithm can be used unless there is a congruence edge:

- Build the congruence closure graph. Edges F_i are colored red, Edges G_j are colored blue.
- Add (colored) disequality edge. Find circle and remove all other edges.
- Combine maximal red paths, remove blue paths.
- The F paths start and end at shared symbols.
  Interpolant is the conjunction of the corresponding equalities.

Handling Congruence Edges (Case 1)

Both sides of the congruence edge belong to G.

$$i_3 = i_2 \land e \neq f \land a(i_1) = e \land a(i_4) = f \land i_1 = i_2 \land i_3 = i_4$$



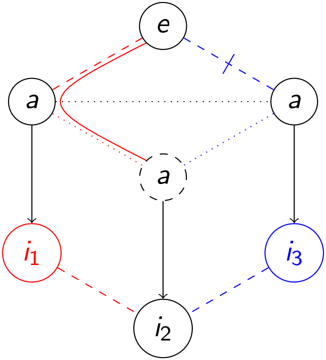
Interpolant:
$i_2 = i_3 \land e \neq f$

- Follow the path that connects the arguments.
- Also add summarized edges for that path.
- Treat the congruence edge as blue edge (ignore it).
- Interpolant is conjunction of all summarized paths.

Handling Congruence Edges (Case 2)

Both side of the congruence edge belong to different formulas.

$$a(i_1) = e \land i_2 = i_1 \land i_3 = i_2 \land a(i_3) \neq e$$



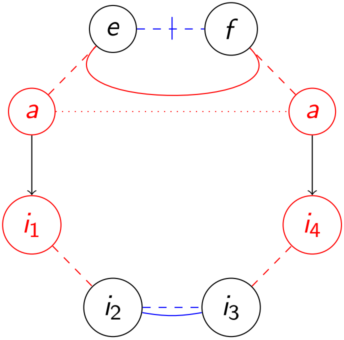
Interpolant: $e = a(i_2)$.

- Function symbol a must be shared.
- Follow the path that connects the arguments.
- Find first change from red to blue.
- Lift function application on that term.
- Summarize $e = a(i_1) \land i_1 = i_2$ by $e = a(i_2)$.
- Compute remaining interpolant as usual.

Handling Congruence Edges (Case 3)

Both side of the congruence edge belong to F.

$$a(i_1) = e \wedge a(i_4) = f \wedge i_1 = i_2 \wedge i_3 = i_4 \wedge i_3 = i_2 \wedge e \neq f$$



Interpolant:
$i_2 = i_3 \rightarrow e = f$

- Follow the path that connects the arguments.
- Find the first and last terms i_2, i_3 where color changes.
- Treat congruence edge as red edge and summarize path.
- The summary only holds under $i_2 = i_3$,i.e., add $i_2 = i_3 \rightarrow e = f$ to interpolants.
- Summarize remaining path segments as usual.

Computing Interpolants for $T_{\mathbb{Q}}$

First apply Dutertre/de Moura algorithm.

- Non-basic variables $x_1, \ldots, x_n$.
- Basic variables $y_1, \ldots, y_m$.
- $y_i = \sum a_{ij} x_j$
- Conjunctive formula
  $y_1 \leq b_1 \ldots y_{m'} \leq b_{m'} \wedge y_{m'+1} \leq b_{m'+1} \ldots y_m \leq b_m$.

The algorithm returns unsatisfiable if and only if there is a line:

	x	$\cdots$	x	y	$\cdots$	y	y	$\cdots$	y
$\vdots$									
y_i/y_i	0	$\cdots$	0	$-/0$	$\cdots$	$-/0$	$-/0$	$\cdots$	$-/0$
$\vdots$									

$y_i = \sum -a'_k y_k$, $a'_k \geq 0$ and $\sum -a'_k b_k > b_i$
(the constraint $y_i \leq b_i$ is not satisfied)

Computing Interpolants for $T_{\mathbb{Q}}$

The conflict is:

$$b_i \geq y_i = \sum -a'_k y_k \geq \sum -a'_k b_k > b_i$$

or

$$0 = y_i + \sum a'_k y_k \leq b_i + \sum a'_k b_k < 0$$

We split the y variables into blue and red ones:

$$0 = \sum_{k=1}^{m'} a_{ik} y_k + \sum_{k=m'+1}^{m} a_{ik} y_k \leq \sum_{k=1}^{m'} a_{ik} b_k + \sum_{k=m'+1}^{m} a_{ik} b_k < 0$$

where $a'_k \geq 0, (a'_i = 1)$. The interpolant I is the red part:

$$\sum_{k=1}^{m'} a_{ik} y_k \leq \sum_{k=1}^{m'} a_{ik} b_k$$

where the basic variables y_k are replaced by their definition.

Example

$$x_1 + x_2 \leq 3 \land x_1 - x_2 \leq 1 \land x_3 - x_1 \leq 1 \land x_3 \geq 4$$

$$y_1 := x_1 + x_2 \qquad b_1 := 3 \qquad y_3 := -x_1 + x_3 \qquad b_3 := 1$$
$$y_2 := x_1 - x_2 \qquad b_1 := 1 \qquad y_4 := -x_3 \qquad b_4 := -4$$

Algorithm ends with the tableaux

	1	1	-4	
	y_2	y_3	y_4	β
y_1	-1	-2	-2	5
x_1	0	-1	-1	3
x_2	-1	-1	-1	2
x_3	0	0	-1	4

Conflict is $0 = y_1 + y_2 + 2y_3 + 2y_4 \leq 3 + 1 + 2 - 8 = -2$.
Interpolant is: $y_1 + y_2 \leq 3 + 1$
or (substituting non-basic vars): $2x_1 \leq 4$.

Correctness

$$F_k \,:\, y_k := \sum_{j=0}^{n} a_{kj}x_j \le b_k, {\scriptstyle (k=1,\dots,m)} \qquad G_k \,:\, y_k := \sum_{j=0}^{n} a_{kj}x_j \le b_k, {\scriptstyle (k=m',\dots,m)}$$

Conflict is $0 = \sum_{k=1}^{m'} a'_k y_k + \sum_{k=m'+1}^{m} a'_k y_k \le \sum_{k=1}^{m'} a'_k b_k + \sum_{k=m'+1}^{m} a'_k b_k < 0$

After substitution the red part $\sum_{k=1}^{m'} a'_k y_k \le \sum_{k=1}^{m'} a'_k b_k$ becomes

$$I \,:\, \sum_{j=1}^{n} \left(\sum_{k=1}^{m'} a'_k a_{kj} \right) x_j \le \sum_{k=1}^{m'} a'_k b_k.$$

- $F \Rightarrow I$ (sum up the inequalities in F with factors a'_k).
- $I \wedge G \Rightarrow \bot$ (sum up I and G with factors a'_k to get $0 \le \sum_{k=1}^{m} a'_k b_k < 0$).
- Only shared symbols in I: $0 = \sum_{k=1}^{m'} a_{kj} a'_k x_j + \sum_{k=m'+1}^{m} a_{kj} a'_k x_j$.
  If the left sum is not zero, the right sum is not zero and x_j appears in F and G.

Computing Interpolants for DPLL(T)

Key Idea: Compute Interpolants for conflict clauses:
Split C into C_F and C_G (if literal appear in F and G put it in C_G).

The conflict clause follows from the original formula:

$$F \land G \Rightarrow C_F \lor C_G$$

Hence, the following formula is unsatisfiable.

$$F \land \neg C_F \land G \land \neg C_G$$

An interpolant I_C for C is the interpolant of the above formula. I_C contains only symbols shared between F and G.

Computing Interpolants for Conflict Clauses

There are several points where conflict clauses are returned:

- Conflict clause is returned by TCHECK.
  Then theory must give an interpolant.
- Conflict clause comes from F.
  Then $F \Rightarrow C_F \vee C_G$.
  Hence, $(F \wedge \neg C_F) \Rightarrow C_G$. Also, $C_G \wedge G \wedge \neg C_G$ is unsatisfiable
  Interpolant is C_G.
- Conflict clause comes from G.
  Then $C_G = C$, $G \Rightarrow C_G$.
  Hence, $(G \wedge \neg C_G)$ is unsatisfiable. Interpolant is $\top$.
- Conflict clause comes from resolution on ℓ.
  Then there is a unit clause $U = \ell \vee U'$ with interpolant I_U
  and conflict clause $C = \neg\ell \vee C'$ with interpolant I_C.

  If $\ell \in F$, set $I_{U' \vee C'} = I_U \vee I_C$
  If $\ell \in G$, set $I_{U' \vee C'} = I_U \wedge I_C$

Computing Interpolants for DPLL(T)

The previous algorithm can compute interpolant for each conflict clause.
The final conflict clause returned is $\bot$.
$I_\bot$ is an interpolant of $F \wedge G$.

Unfortunately, it is not that easy...
... because equalities shared by Nelson-Oppen can contain red and blue symbols simultaneously.

Interpolating in theory combination is still ongoing research.