Abstract Conflict Driven Clause Learning

Leopold Haller

Joint work with
Vijay D’Silva, Alberto Griggio, Michael Tautschnig, Daniel Kroening
<table>
<thead>
<tr>
<th>What the British say</th>
<th>What the British mean</th>
<th>What others understand</th>
</tr>
</thead>
<tbody>
<tr>
<td>That's not bad.</td>
<td>That's good</td>
<td>Could be better.</td>
</tr>
<tr>
<td>Oh, by the way ...</td>
<td>The primary purpose of our discussion is ...</td>
<td>It's not very important, but ...</td>
</tr>
<tr>
<td>What an Abs. Int. person says</td>
<td>What they might mean</td>
<td>What others understand</td>
</tr>
<tr>
<td>-------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Isn’t this an instance of abstract interpretation?</td>
<td>I think there is a simple top-down characterisation of this in the language of algebra, fixed points and abstraction.</td>
<td>This is a trivial consequence of abstract interpretation.</td>
</tr>
<tr>
<td>Technique X computes an abstract fixed point.</td>
<td>There is a view of X that allows for the application of a rich body of results.</td>
<td>Details are unimportant.</td>
</tr>
</tbody>
</table>
“Everything is Abstract Interpretation ...”

<table>
<thead>
<tr>
<th>What an Abs. Int. person says</th>
<th>What they might mean</th>
<th>What others understand</th>
</tr>
</thead>
<tbody>
<tr>
<td>Isn’t this an instance of abstract interpretation?</td>
<td>I think there is a simple top-down characterisation of this in the language of algebra, fixed points and abstraction.</td>
<td>This is a trivial consequence of abstract interpretation.</td>
</tr>
<tr>
<td>Technique X computes an abstract fixed point.</td>
<td>There is a view of X that allows for the application of a rich body of results.</td>
<td>Details are unimportant.</td>
</tr>
</tbody>
</table>

... including SAT solvers

(Satisfiability Solvers are Static Analysers. D’Silva, Haller, Kroening, SAS 2012)
Propositional Satisfiability (SAT)

Given a propositional formula $\varphi$, is there a propositional truth assignment $\sigma$ such that $\sigma \models \varphi$.

- Solvers are based on Conflict Driven Clause Learning (CDCL)
- Basis of modern Satisfiability Module Theory (SMT) solvers
- Critical components of program verification techniques
Work on CDCL has resulted in an exponential decrease in runtimes.

Can we lift this success to other domains?
SMT via DPLL(T)

Solve satisfiability for (QF) first order formula with background theory

\[
\begin{align*}
(p \lor q) \land (r \lor s) \\
(p \lor q) \land (2x - y \geq 1) \land (x = 5 \lor y = x)
\end{align*}
\]

CDCL enumerates candidate propositional truth assignments, theory solver checks consistency.

DPLL(T) is a mathematical recipe and implementation framework for building SMT decision procedures!
DPLL(T) can be viewed to partition the space of potential models using the structure of the formula. Measures have to be taken to avoid enumeration behaviour.

\[(x = 0 \lor x = 2 \lor x = 4 \lor \ldots \lor x = 2k) \land (y = 0 \lor y = 2 \lor y = 4 \lor \ldots \lor y = 2k) \land (x + y = 2c + 1)\]

DPLL(T) explores truth assignments to predicates

Full even / odd partitioning
CDCL \rightarrow \text{Natural Domain} \rightarrow SMT
Natural Domain SMT

Propositional CDCL

DPLL(T)

Model Search

Confl. Analysis

Theory Solver

"Theory" Learning

Theory Model Search

Theory Conflict Analysis

Conflict

Natural Domain SMT

Monday, 23 July 12
Abstract Interpretation
Abstract Interpretation by Example: Intervals

Track possible range for variable

```c
int a = 5;
int b;

if(*)
    b = 3;
else
    b = -3;

a += b;

assert(a == 0);
```
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

```
int a = 5;
int b;

if(*)
    b = 3;
else
    b = -3;

a += b;

assert(a == 0);
```
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

```
    T
    int a = 5;
    int b;

    if(*)
        b = 3;
    else
        b = -3;
    a += b;

    assert(a == 0);
```
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

<table>
<thead>
<tr>
<th>T</th>
<th>int a = 5; int b;</th>
</tr>
</thead>
<tbody>
<tr>
<td>a (\mapsto) [5, 5]</td>
<td>if(*) b = 3; else b = -3;</td>
</tr>
<tr>
<td></td>
<td>a += b;</td>
</tr>
<tr>
<td></td>
<td>assert(a == 0);</td>
</tr>
</tbody>
</table>
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

\[ T \]

\[
\begin{array}{ll}
\text{int } a = 5; \\
\text{int } b;
\end{array}
\]

Imprecise OA:

\[
\begin{array}{ll}
\text{if}(\star) & b = 3; \\
\text{else} & b = -3;
\end{array}
\]

\[
\begin{array}{l}
a += b; \\
\end{array}
\]

\[
\begin{array}{l}
\text{assert}(a == 0); \\
\end{array}
\]
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

\[
\begin{array}{ll}
\top & \text{int } a = 5; \\
& \text{int } b; \\
\hline
a \mapsto [5, 5] & \text{if}(*) \\
& b = 3; \\
& \text{else} \\
& b = -3; \\
\hline
\text{Imprecise OA:} & a += b; \\
\hline
a \mapsto [5, 5], b \mapsto [-3, 3] & \text{assert}(a == 0); \\
\hline
a \mapsto [2, 8], b \mapsto [-3, 3] & \\
\end{array}
\]
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

\[
\begin{align*}
T \\
\text{int } a &= 5; \quad \text{int } b; \\
\text{if}(*) &\quad \text{else} \quad b = 3; \\
a &\mapsto [5, 5], b &\mapsto [-3, 3] \\
a &\mapsto [2, 8], b &\mapsto [-3, 3]
\end{align*}
\]

Underapproximate Analysis with weakest precondition

\[
\begin{align*}
\text{assert}(a == 0);
\end{align*}
\]
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

\[
\begin{array}{ll}
T \\
\hline
a \mapsto [5, 5] \\
\end{array}
\]

int a = 5;
int b;

Underapproximate Analysis with weakest precondition

Imprecise OA:

\[
\begin{array}{ll}
\hline
\text{if(*)} & \\
b = 3; & \text{else} \\
b = -3; & \\
\end{array}
\]

\[
\begin{array}{ll}
a \mapsto [5, 5], b \mapsto [-3, 3] \\
\end{array}
\]

a += b;
\{ a \mapsto [-\infty, -1], a \mapsto [1, \infty] \}
assert(a == 0);
Abstract Interpretation by Example: Intervals

Track possible range for variable

<table>
<thead>
<tr>
<th>Overapproximate Analysis with strongest postcondition</th>
<th>Underapproximate Analysis with weakest precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td></td>
</tr>
<tr>
<td>( a \mapsto [5, 5] )</td>
<td>( \text{int } a = 5; ) ( \text{int } b; )</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Imprecise OA:</td>
<td></td>
</tr>
<tr>
<td>( a \mapsto [5, 5], b \mapsto [-3, 3] )</td>
<td>( \text{if}(*) ) ( b = 3; ) ( \text{else} ) ( b = -3; ) ( {(a \mapsto [4, \infty], b \mapsto [-3, 3])} )</td>
</tr>
<tr>
<td>( a \mapsto [2, 8], b \mapsto [-3, 3] )</td>
<td></td>
</tr>
<tr>
<td>( a += b; )</td>
<td>( {a \mapsto [-\infty, -1], a \mapsto [1, \infty]} )</td>
</tr>
<tr>
<td>assert( (a == 0); )</td>
<td></td>
</tr>
</tbody>
</table>
Abstract Interpretation by Example: Intervals

Track possible range for variable

Overapproximate Analysis with strongest postcondition

\[ T \]

\[
\begin{align*}
a &\mapsto [5, 5] \\
\text{int } a &= 5; \\
\text{int } b; &\mapsto [4, \infty]
\end{align*}
\]

Underapproximate Analysis with weakest precondition

\[
\begin{align*}
\text{if}(*) & \text{ b } = 3; \\
\text{else} & \text{ b } = -3; & (a \mapsto [4, \infty], b \mapsto [-3, 3])
\end{align*}
\]

Imprecise OA:

\[
\begin{align*}
a &\mapsto [5, 5], b \mapsto [-3, 3] \\
a &\mapsto [2, 8], b \mapsto [-3, 3] \\
\text{a } &\text{ += b}; \\
\text{assert}(\text{a } \text{ == 0});
\end{align*}
\]

UA "guess":

\[
\begin{align*}
\{(a \mapsto [4, \infty], b \mapsto [-3, 3])\}
\end{align*}
\]
## Abstract Interpretation by Example: Intervals

Track possible range for variable

<table>
<thead>
<tr>
<th>Overapproximate Analysis with strongest postcondition</th>
<th>Underapproximate Analysis with weakest precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a \mapsto [5, 5] )</td>
<td>( { a \mapsto [4, \infty] } )</td>
</tr>
<tr>
<td>( \text{int } a = 5; ) ( \text{int } b; )</td>
<td></td>
</tr>
<tr>
<td>( \text{if}(*) ) ( b = 3; ) ( \text{else} ) ( b = -3; )</td>
<td>( { (a \mapsto [4, \infty], b \mapsto [-3, 3]) } )</td>
</tr>
<tr>
<td>( \text{Imprecise OA:} ) ( a \mapsto [5, 5], b \mapsto [-3, 3] )</td>
<td></td>
</tr>
<tr>
<td>( a \mapsto [2, 8], b \mapsto [-3, 3] ) ( a += b; )</td>
<td>( { a \mapsto [-\infty, -1], a \mapsto [1, \infty] } )</td>
</tr>
<tr>
<td></td>
<td>\text{assert}(a == 0);</td>
</tr>
</tbody>
</table>
Abstract Interpretation by Example: Intervals

Track possible range for variable

<table>
<thead>
<tr>
<th>Overapproximate Analysis with strongest postcondition</th>
<th>Underapproximate Analysis with weakest precondition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a \mapsto [5, 5]$</td>
<td>${a \mapsto [4, \infty]}$</td>
</tr>
<tr>
<td>$\text{int } a = 5;$</td>
<td></td>
</tr>
<tr>
<td>$\text{int } b;$</td>
<td></td>
</tr>
<tr>
<td>$\text{if}(\ast){\text{else} \quad b = -3;}$</td>
<td></td>
</tr>
<tr>
<td>$a \mapsto [5, 5], b \mapsto [-3, 3]$</td>
<td>${(a \mapsto [4, \infty], b \mapsto [-3, 3])}$</td>
</tr>
<tr>
<td>$a \mapsto [2, 8], b \mapsto [-3, 3]$</td>
<td>${a \mapsto [-\infty, -1], a \mapsto [1, \infty]}$</td>
</tr>
<tr>
<td>$a += b;$</td>
<td></td>
</tr>
<tr>
<td>$\text{assert}(a == 0);$</td>
<td></td>
</tr>
</tbody>
</table>

Imprecise OA: $\{a \mapsto [4, \infty]\}$

Sound, but incomplete
Abstract Interpretation

Concrete Lattice

\((\wp(States), \subseteq, \cap, \cup)\)

Galois connection:

\[
\begin{array}{c}
\gamma \\
\downarrow \\
\alpha
\end{array}
\]

Abstract Lattice

\((Intervals, \sqsubseteq, \cap, \cup)\)
Abstract Interpretation

Concrete Lattice
$(\varnothing(States), \subseteq, \cap, \cup)$

Galois connection:

Abstract Lattice
$(Intervals, \subseteq, \cap, \cup)$

Abstraction and concretisation function

$\alpha(\{x \mapsto 3, x \mapsto 1, x \mapsto 9\}) = x \mapsto [1, 9]$  
$\gamma(x \mapsto [4, 6]) = \{x \mapsto 4, x \mapsto 5, x \mapsto 6\}$
Abstract Interpretation

Concrete Lattice
$(\wp(\text{States}), \subseteq, \cap, \cup)$

Galois connection:
$\begin{array}{ccc}
\alpha & \gamma & \beta \\
\downarrow & & \downarrow \\
\wp(\text{States}) & \subseteq & \wp(\text{States})
\end{array}$

Abstract Lattice
$(\text{Intervals}, \subseteq, \cap, \cup)$

Abstraction and concretisation function
\[
\alpha(\{x \mapsto 3, x \mapsto 1, x \mapsto 9\}) = x \mapsto [1, 9] \\
\gamma(x \mapsto [4, 6]) = \{x \mapsto 4, x \mapsto 5, x \mapsto 6\}
\]

Concrete transformer

Sound abstr. transformer

post : $\wp(\text{States}) \rightarrow \wp(\text{States})$

$\hat{\text{post}} : \text{Intervals} \rightarrow \text{Intervals}$

$\text{post} \circ \gamma \subseteq \gamma \circ \hat{\text{post}}$
Approximating Fixed Points

Fixed points can be computed in the abstract

\[ \text{lfp } X. I \cup \text{post}(X) \subseteq \gamma(\text{lfp } X. \alpha(I) \sqcup \text{post}(X)) \]
Accelerating Fixed Point Computations

```c
x = 0;
while(x < 1000)
    x++;
```

Fixed point computations might take a long time (or fail to terminate):

\[
F_0 : x \mapsto [0, 0] \quad F_1 : x \mapsto [0, 1] \quad F_2 : x \mapsto [0, 2] \quad F_3 : x \mapsto [0, 3] \quad \ldots
\]
Accelerating Fixed Point Computations

\[
x = 0;\\while(x < 1000)\\x++;\]

Fixed point computations might take a long time (or fail to terminate):
\[
F_0 : x \mapsto [0, 0] \quad F_1 : x \mapsto [0, 1] \quad F_2 : x \mapsto [0, 2] \quad F_3 : x \mapsto [0, 3] \quad \ldots
\]

Widening
(jumps above least fixed point)

Narrowing
(stay above greatest fixed point)
Abstract Interpretation

\[\downarrow\]

Interpreting Logic
Abstractly Interpreting Logic

Check satisfiability of the following formula

\[ \varphi = p \land (\neg p \lor q) \land (\neg p \lor \neg q) \]

Prove the following program safe

```c
int main()
{
    if(p)
        if(!p || q)
            if(!p || !q)
                assert(false);
}
```
Abstractly Interpreting Logic

Constants analysis

\[
\begin{array}{cccc}
p : t & q : t & q : f & p : f \\
\hline
\top & p : t, q : t & p : t, q : f & p : f, q : t & p : f, q : f \\
\hline
\end{array}
\]

```
int main()
{
    \top
    if(p) { p : t }
    if(!p || q) { p : t, q : t }
    if(!p || !q) { \bot }
    assert(false);
}
```
Abstractly Interpreting Logic

Set of formulae $Form$

Set of structures $Struct$

Semantic entailment relation $\models \in \varphi(Struct \times Form)$

Concrete Domain $(\varphi(Struct), \subseteq, \cap, \cup)$
Abstractly Interpreting Logic

Set of formulae
\[ \text{Form} \]

Set of structures
\[ \text{Struct} \]

Semantic entailment relation
\[ \models \in \wp(\text{Struct} \times \text{Form}) \]

Concrete Domain
\[ (\wp(\text{Struct}), \subseteq, \cap, \cup) \]

E.g., propositional logic:
\[ \text{Lit} = \{p, \neg p \mid p \in \text{Props}\} \]
\[ \text{Form} = \wp(\text{Clauses}) \]
\[ \text{Clauses} = \wp(\text{Lit}) \]
\[ \text{Struct} = \text{Props} \rightarrow \{t, f\} \]

\[ \sigma \models \varphi \text{ iff } \]
\[ \forall C \in \varphi. \exists l \in C. \ (l = p \land \sigma(p) = t) \lor (l = \neg p \land \sigma(p) = f) \]
Abstract Satisfaction

Structure transformers;

\[ \text{mods}_\varphi(S) = \{ \sigma \mid \sigma \in S \land \sigma \models \varphi \} \]
\[ \text{confs}_\varphi(S) = \{ \sigma \mid \sigma \in S \lor \sigma \not\models \varphi \} \]

\[ \text{mods}_\varphi = \text{post\_assume}(\varphi) \]
\[ \text{confs}_\varphi = \text{pre\_assume}(\varphi) \]
Abstract Satisfaction

Structure transformers;

\[
\text{mods}_\varphi(S) = \{\sigma \mid \sigma \in S \land \sigma \models \varphi\} \quad \text{confs}_\varphi(S) = \{\sigma \mid \sigma \in S \lor \sigma \not\models \varphi\}
\]

\[
\text{mods}_\varphi = \text{post}_{\text{assume}}(\varphi) \quad \text{confs}_\varphi = \text{pre}_{\text{assume}}(\varphi)
\]

Overapproximation \(\text{amods}_\varphi\) of \(\text{mods}_\varphi\)

Underapproximation \(\text{aconfs}_\varphi\) of \(\text{confs}_\varphi\)

\[
\text{gfp} \quad \text{amods}_\varphi = \bot \quad \text{or}
\]

\[
\text{lfp} \quad \text{aconfs}_\varphi = \top \quad \implies \varphi \text{ is unsatisfiable}
\]
Natural Domain

SMT

CDCL as
Natural Domain SMT
for propositional Logic

Model Search
w. Abstraction

Interpreting Logic

Conflict Analysis
w. Abstraction
Model Search

Find either a satisfying assignment or a conflicting partial assignment

Deduction: Infer variable assignments
Decisions: Guess variable assignments
Partial Assignments are an Abstract Domain

```c
#define l_True (lbool ((uint8_t)0))
#define l_False (lbool ((uint8_t)1))
#define l.Undef (lbool ((uint8_t)2))

class lbool {
    [...] }

class Solver {
    [...]  
    bool okay () const;
    vec<lbool> assigns; // The current assignments.
    // Enqueue a literal. Assumes value of literal is undefined.
```

\[
\varphi(Props \rightarrow \{t,f\}) \xleftarrow{\alpha} \xrightarrow{\gamma} \\
\]

\[
\begin{array}{cccc}
\text{p:}t & \text{q:}t & \text{q:f} & \text{p:f} \\
\text{p:}t, q: t & p: t, q: f & p: f, q: t & p: f, q: f \\
\end{array}
\]
Deduction Computes a Greatest Fixed Point

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

\[ \text{gfp } \text{mods}_\varphi \]
Deduction Computes a Greatest Fixed Point

\[-p \land (p \lor q) \land (\neg q \lor r)\]

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

\[\text{gfp } mods_{\varphi}\]
Deduction Computes a Greatest Fixed Point

\[
\neg p \land (p \lor q) \land (\neg q \lor r)
\]

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

\[
gfp \text{ mods}_\varphi
\]
Deduction Computes a Greatest Fixed Point

\[ \neg p \land (p \lor q) \land (\neg q \lor r) \]

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

\[ \text{gfp } \text{mods}_\varphi \]
Deduction Computes a Greatest Fixed Point

\[ \neg p \land (p \lor q) \land (\neg q \lor r) \]

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

\[ \text{gfp } \text{mods}_\varphi \]
Deduction Computes a Greatest Fixed Point

\[ \neg p \land (p \lor q) \land (\neg q \lor r) \]

\[ (p:f) \]

\[ (p:f, q:t) \]

\[ (p:f, q:t, r:t) \]

\[ bcp(\pi) = \text{gfp } X. \, \text{unit}(\pi \sqcap X) \]

The unit rule overapproximates the model transformer, BCP abstractly computes the fixed point:

\[ \text{gfp } \text{mods}_\varphi \]
Decision Making is Dual Widening

Once no more new facts can be deduced, a solver heuristically picks a truth value for an unassigned variable

\[ p \land (q \lor r) \land (q \lor \neg r) \]

Deduction

\[ \top \]

\[ p : t \]
Decision Making is Dual Widening

Once no more new facts can be deduced, a solver heuristically picks a truth value for an unassigned variable

\[ p \land (q \lor r) \land (q \lor \neg r) \]

Decision \( q : f \)

\[ p : t, q : f \]
Decision Making is Dual Widening

Once no more new facts can be deduced, a solver heuristically picks a truth value for an unassigned variable

\[ p \land (q \lor r) \land (q \lor \neg r) \]

Diagram:

- Decision: \( q: f \)
- Deduction:
  - \( p: t \)
  - \( p: t, q: f \)
  - \( p: t, q: f, r: t \)
  - \( \bot \)
Decision Making is Dual Widening

Once no more new facts can be deduced, a solver heuristically picks a truth value for an unassigned variable

\[ p \land (q \lor r) \land (q \lor \neg r) \]

Recall: Widenings jump over a least fixed point
Decisions jump under a greatest fixed point (unusual: unsound!)
Conflicts Analysis

Abduction:
Find possible generalisations of conflict

Heuristic Choice:
Choose one generalisation
CDCL solvers record deductions in data structure called implication graph

\[(\neg p \lor q) \land (\neg p \lor \neg r) \land (\neg q \lor r \lor \neg s) \land (s \lor t) \land (s \lor \neg t)\]
CDCL solvers record deductions in data structure called implication graph

\[(\neg p \lor q) \land (\neg p \lor \neg r) \land (\neg q \lor r \lor \neg s) \land (s \lor t) \land (s \lor \neg t)\]

Conflict abduction is performed by obtaining cuts through the graph

Original conflict \[\pi = (p:t, q:t, r:f, s:f, t:t)\]

Possible generalisations from cuts \[cut(\{\pi\}) = \{(p:t), (q:t, r:f), (s:f)\}\]
Abduction computes a least fixed point

\[(p: t, q: t) \rightarrow (r: f, s: t)\]  generalisation from clause minimisation

\[(r: f, s: t) \rightarrow (r: f)\]  generalisation from graph cuts

Original conflict
Abduction computes a least fixed point

\[
\begin{align*}
(p:t, q:t) & \quad (r:f, s:t) \\
(r:f) & \quad (s:t) \quad \text{generalisation from clause minimisation}
\end{align*}
\]

\[
\begin{align*}
(p:t, q:t, r:f, s:t) & \quad \text{generalisation from graph cuts}
\end{align*}
\]

Original conflict

\[
\{(p:t, q:t), (r:f), (s:t)\} \quad \text{generalisation from clause minimisation}
\]

\[
\{ (p:t, q:t), (r:f, s:t) \}
\]

\[
\{ (p:t, q:t, r:f, s:t) \}
\]

Collecting all conflicts
Abduction computes a least fixed point

\[
\begin{align*}
(p:t, q:t) & \rightarrow (r:f, s:t) \\
(p:t, q:t, r:f, s:t) & \rightarrow (r:f)
\end{align*}
\]

Original conflict

\[
\{(p:t, q:t), (r:f), (s:t)\}
\]

Collecting all conflicts

\[
\{(p:t, q:t), (r:f, s:t)\}
\]

\[
\{(p:t, q:t, r:f, s:t)\}
\]

Abduction underapproximately computes the fixed point \( \text{lfp } confs \varphi \)
Heuristic Choice is Dual Narrowing

\{(p:t, q:t), (r:f), (s:t)\} →

\{(p:t, q:t), (r:f, s:t)\}

\{(p:t, q:t, r:f, s:t)\}

Collecting all conflicts
Heuristic Choice is Dual Narrowing

\[
\{(p:t, q:t), (r:f), (s:t)\} \Rightarrow
\]

\[
\{(p:t, q:t), (r:f, s:t)\}
\]

Collecting all conflicts

\[
\{(r:f)\} \Rightarrow
\]

\[
\{(r:f, s:t)\}
\]

SAT Solvers choose one reason
Heuristic Choice is Dual Narrowing

Recall that narrowing is used to converge above a greatest fixed point. Heuristic choice of conflict reasons leads to convergence below a least fixed point!
ACDCL: A recipe for deriving natural domain SMT solvers from abstract domains

Search

\text{gfp}(amod_\varphi)

Dual widen

\text{SAT} ?

Prove

\text{lfp}(aconf_\varphi)

Dual narrow

\text{UNSAT}

Learned transformer

Conflict

Overapproximating domain $O$

Underapproximating domain $U$
Model Search and Conflict Analysis with Abstract Domains

\[ \text{Struct} \]

\[ \varphi \]
Model Search and Conflict Analysis with Abstract Domains

\[ \text{Struct} \]

\[ \varphi \]
Model Search and Conflict Analysis with Abstract Domains

\[ \text{Struct} \]

\[ \varphi \]

Deduction
Model Search and Conflict Analysis with Abstract Domains
Model Search and Conflict Analysis with Abstract Domains

\[ \text{Struct} \]

Deduction
Decision
Deduction

\( \varphi \)
Model Search and Conflict Analysis with Abstract Domains
Model Search and Conflict Analysis with Abstract Domains

\textit{Struct}

Deduction
Decision
Deduction
Conflict
Abduction
Model Search and Conflict Analysis with Abstract Domains

Struct

Deduction
Decision
Deduction
Conflict
Abduction
Choice
Model Search and Conflict Analysis with Abstract Domains

\textit{Struct}

\begin{itemize}
  \item Deduction
  \item Decision
  \item Deduction
  \item Conflict
  \item Abduction
  \item Choice
  \item Abduction
\end{itemize}
Model Search and Conflict Analysis with Abstract Domains
CDCL as Natural Domain SMT for propositional Logic

Model Search w. Abstraction

Conflict Analysis w. Abstraction

Abstract Learning
Tabu Learning

Simple but weak form of learning:
When the conflict region is reentered immediately deduce conflict
Tabu Learning

Simple but weak form of learning:
When the conflict region is reentered immediately deduce conflict
Tabu Learning

Simple but weak form of learning:
When the conflict region is reentered immediately deduce conflict

No lattice theoretic prerequisites, possible over any domain

\[
tabu_C(\pi) = \begin{cases} 
\bot & \text{if } \pi \sqsubseteq C \\
\pi & \text{otherwise}
\end{cases}
\]
Propositional Clause Learning

When assignment is “nearly conflicting”, drive the search away from the conflict.
Propositional Clause Learning

When assignment is “nearly conflicting”, drive the search away from the conflict

\[ \pi \]

\[ C \]

\[ \varphi \]

\[ C \]

\[ \varphi \]
Propositional Clause Learning

When assignment is “nearly conflicting”, drive the search away from the conflict.

Propositional clause: $C = (p:t, q:t, r:f) = (p:t) \sqcap (q:t) \sqcap (r:f)$

Complements drive the search away from conflict:

$unit_{(p:t, q:t, r:f)}(\pi) = \begin{cases} 
\pi \sqcap \neg(p:t) & \pi \subseteq (q:t) \land \pi \subseteq (r:f) \\
\pi \sqcap \neg(q:t) & \pi \subseteq (p:t) \land \pi \subseteq (r:f) \\
\pi \sqcap \neg(r:f) & \pi \subseteq (p:t) \land \pi \subseteq (q:t) 
\end{cases}$

Decomposition allows us to express “nearly conflicting”.
Complementable Meet Irreducibles

Clause learning requires a weak complementation property of the abstraction

\[(p:t, q:f) = (p:t) \sqcap (q:f)\]

No precise complement \hspace{1cm} Precise complement

Every element needs to have a decomposition into precisely complementable elements.
Complementable Meet Irreducibles

Examples of lattices with complementable meet irreducibles
Complementable Meet Irreducibles

Examples of lattices with complementable meet irreducibles

Intervals and Octagons are intersections of complementable half-spaces
Complementable Meet Irreducibles

Examples of lattices with complementable meet irreducibles

Intervals and Octagons are intersections of complementable half-spaces

Trace abstraction based on control history

$Branches \rightarrow \{left, right, \top\}$

$CFG$

$=$

Monday, 23 July 12
Generalised Unit Rule
Generalised Unit Rule

Intervals

Conflict $c$
Generalised Unit Rule

Intervals

Conflict $c$

Element $o$

$unit_c(o)$
Generalised Unit Rule

Intervals

Conflict $c$

Element $o$

$unit_c(o)$

Trace abstractions:

Conflict $c$
Generalised Unit Rule

Intervals

Conflict \( c \)

Element \( o \)

\( \text{unit}_c(o) \)

Trace abstractions:

Conflict \( c \)

Element \( o \)
Generalised Unit Rule

Trace abstractions:

Conflict $c$

Element $o$

unit$_c(o)$
CDCL-Style Static Analysis

Abs. Interpretation based SMT Solver

Instances/Applications
An SMT Solver based on ACDCL

(Joint work with Alberto Griggio, implemented using MathSAT infrastructure)
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)

\[ x \in [-\infty, 0] \]
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)

\[ x \in [-\infty, 0] \]

\[ y \in [-\infty, 0] \]
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)

\[ x \in [-\infty, 0] \]
\[ y \in [-\infty, 0] \]
\[ \triangleq \]
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)

\[ x \in [-\infty, 2] \]
\[ x \in [-\infty, 0] \]
\[ y \in [-\infty, 0] \]
\[ y \in [-\infty, 7] \]
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)
Generalised Conflict Graphs

FirstUIP conflict graph analysis can be lifted to work over abstractions

\[ x = y \land x + y \geq 10 \]

Graph nodes are meet irreducibles (e.g., half spaces)

1. Generalise each node of the conflict graph using heuristic choice
2. Cut the graph
We check whether the weakened bound altogether. We attempt to weaken the current element to the deduction. Then we step backwards through the trail and remove all bounds over variables from which is sufficient to deduce an element. In the trail, we start with trivial abduction backjump potential. In order to generalise the deduction decisions, and thus will hold for a larger portion of the search lead to propagations that are affected by a smaller number of an additional, important criterion. Since learnt lemmas are heuristic need to be taken into account. However, there is considerations similar to those mentioned for the decision.

We suggest that the precise way in which bounds are relaxed there may be many incomparable relaxations. Our experiments abduction that relaxes bounds iteratively. As mentioned earlier, improvements. This is part of ongoing and future work.

Our choice heuristic, called (iii) feasibility of systems of inequalities over bounded variables. We make our benchmarks and the tool available for experimentation by other re-

We have performed two different sets of experiments. In

We have experimented with stronger but computationally expensive generalisation techniques such as finding systems of inequalities over bounded variables. We make our benchmarks and the tool available for experimentation by other re-

Fig. 2. Comparison of the main bit-vector division in the SMT-COMP 2011 results have been obtained on an Intel Xeon machine with floating-point variables. We make our benchmarks and the tool available for experimentation by other re-

We have fixed our experiments with stronger but computationally expensive generalisation techniques such as finding.
ACDCL for Programs

Treat program analysis as a logical problem:

\[ \pi \models P \text{ iff trace } \pi \text{ is an erroneous trace generated by program } P \]
ACDCL for Programs

Treat program analysis as a logical problem:
\[ \pi \models P \iff \text{trace } \pi \text{ is an erroneous trace generated by program } P \]

Fwd/bwd lfp analysis
with strongest postcondition and preimage

Fwd/bwd gfp with
weakest precondition and universal post.

\[ \text{gfp}(amod_\varphi) \]

\[ \text{lfp}(aconf_\varphi) \]

Dual widen

Dual narrow

Search

refined transformer

Prove

\[ ? \]

\[ \text{SAT} \]

\[ \text{UNSAT} \]

\[ \sim \text{partial safety proof} \]
Example 1: Interval Conflict Graphs

Diagram:

- Node $n_1$ with edges:
  - $[a \leq -2]$ to $c_1$
  - $[a \geq 1]$ to $c_2$
  - $[a = -1]$ to $c_3$
  - $[a = 0]$ to $c_4$

- Node $n_2$ with edges:
  - $b := -1$ to $c_1$
  - $b := 1$ to $c_3$
  - $b := 2$ to $c_4$
  - $b := -2$ to $c_2$

- Node $[b = 0]$ with edges:
  - $\leftarrow$ to $n_2$

Constraints:

- $b := 2$
- $b := -2$
- $b := 1$
- $b := -1$
- $b := 0$
**Example 1: Interval Conflict Graphs**

DL0

- $c_1$: \[ a \leq -2 \]
- $c_2$: \[ a = -1 \]
- $c_3$: \[ a = 0 \]
- $c_4$: \[ a \geq 1 \]

- $n_1$: \[ a = -1 \]
- $n_2$: \[ a = 1 \]
- $n_3$: \[ b = -2 \]
- $n_4$: \[ b = 2 \]

- $c_1$: \[ b = -1 \]
- $c_2$: \[ b = 1 \]
- $c_3$: \[ b = 0 \]

$\text{SAFE}!$

Generalise!

Find cut

DPLL is Abstract Interpretation
Example 1: Interval Conflict Graphs

DL0

$c_1: a \leq -2$
$c_2: a \leq -1$
$c_3: a \geq 0$
$c_2: a \geq -1$
$c_4: a \geq 1$

$n_2: b \leq 2$
$n_2: b \geq -2$

$\vdash b \leq 0$
$\vdash b \geq 0$

$[a \leq -2]$
$[a \geq 1]$
$[a = -1]$
$[a = 0]$

$b := -1$
$b := 1$
$b := 2$
$b := -2$

$[b = 0]$
Example 1: Interval Conflict Graphs

\[
\begin{align*}
\text{DL0} & \quad c_1 \colon a \leq -2 \\
& \quad c_2 \colon a \leq -1 \\
& \quad c_3 \colon a > 0 \\
& \quad c_4 \colon a > -1 \\
& \quad n_1 \colon b \leq 2 \\
& \quad n_2 \colon b \geq -2 \\
& \quad \frac{1}{2} \colon b \leq 0 \\
& \quad \frac{1}{2} \colon b \geq 0
\end{align*}
\]

\[
\begin{align*}
\text{DL1} & \quad n_1 \colon a \leq -42
\end{align*}
\]
Example 1: Interval Conflict Graphs

DL0

\[ c_2 : a \leq -1 \]
\[ c_3 : a \leq 0 \]
\[ c_3 : a \geq 0 \]
\[ c_2 : a \geq -1 \]
\[ c_1 : a \leq -2 \]
\[ n_2 : b \leq 2 \]
\[ n_2 : b \geq -2 \]
\[ b : b \leq 0 \]
\[ b : b \geq 0 \]

\[ n_1 \]
[\[ a \leq -2 \]\[ a \geq 1 \]\[ a = -1 \]\[ a = 0 \]\[ b = -2 \]\[ b = 0 \]\[ b = 2 \]]

DL1

\[ n_1 : a \leq -42 \]
\[ c_1 : a \leq -42 \]
\[ c_2 : \bot \]
\[ c_3 : \bot \]
\[ c_4 : \bot \]
\[ n_2 : b \geq 2 \]
\[ b : b \leq 0 \]
\[ b : b \geq 0 \]

SAFE
Example 1: Interval Conflict Graphs

DL0

\[ n_2 : b \leq 2 \quad n_2 : b \geq -2 \]

\[ \frac{c_2}{c_1} : a \leq -2 \quad c_3 : a \leq 0 \quad c_3 : a \geq 0 \quad c_2 : a \geq -1 \quad c_4 : a \geq 1 \]

DL1

\[ \frac{c_1}{n_1} : a \leq -2 \quad n_2 : b \geq 1 \quad \frac{c_2 \perp}{c_3 \perp} : \perp \]

\[ \frac{c_4}{n_1} : \perp \]

SAFE  →  Generalise!

Under-approximate
weakest pre-condition

\( [a \leq -2] \)

\( [a \geq 1] \)

\( [a = 1] \)

\( [a = 0] \)

\( b := -1 \)

\( b := 1 \)

\( b := 2 \)

\( b := -2 \)

\( [b = 0] \)

\( \frac{1}{1} \)
Example 1: Interval Conflict Graphs

ACDCL “intelligently” decomposes the problem
Example 2: Problem Dependent Decomposition

Input Range

\[
\begin{align*}
-\frac{\pi}{2} & \quad \frac{\pi}{2}
\end{align*}
\]

Sine function

Program output
Example 2: Problem Dependent Decomposition

result $\leq 2.0$

Partitions

Safety bound

result $\geq -2.0$
Example 2: Problem Dependent Decomposition

\[
\begin{align*}
\text{result} \leq 1.5 & \quad -\frac{\pi}{2} \\
\text{result} \geq -1.5 & \quad \frac{\pi}{2}
\end{align*}
\]
Example 2: Problem Dependent Decomposition

\[ \text{result} \leq 1.2 \]

\[ \text{result} \geq -1.2 \]
Example 2: Problem Dependent Decomposition

\[ \frac{\pi}{2} \leq 1.1 \]

result \( \leq 1.1 \)

\[ \frac{\pi}{2} \geq -1.1 \]

result \( \geq -1.1 \)
Example 2: Problem Dependent Decomposition

\[ \frac{-\pi}{2} \leq \text{result} \leq \frac{\pi}{2} \]

result \( \leq 1.01 \)

result \( \geq -1.01 \)
Example 2: Problem Dependent Decomposition

Intelligent decomposition of the analysis
Oh, East is East, and West is West, and never the twain shall meet, Till Earth and Sky stand presently at God’s great Judgment Seat; But there is neither East nor West, Border, nor Breed, nor Birth, When two strong men stand face to face, tho’ they come from the ends of the earth!

Thanks for your attention!