Abstract Domains

Patrick Cousot
École normale supérieure
45 rue d’Ulm, 75230 Paris cedex 05, France
Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

Content

- Abstract domains ........................................ 4
- Example abstract domain: Symbolic execution .... 8
- Parametric abstract domains ......................... 12
- Generation of execution examples by parametric symbolic execution ................................. 20

Static Analysis

- Static analysis computes an overapproximation $A$ of an abstract semantics $\text{ifp}^= F \sqsubseteq A$ where $F \in D \mapsto D$
- A compositional approach is preferable:
  - The abstract domain $D$ is defined by combination of elementary abstract domains $L$
  - The abstract transformer $F$ is defined inductively (e.g. by induction on the program syntax) by composition of elementary abstract transformers $f$ ...

This structure $(L, \sqsubseteq, \bot, \ldots, f)$ leads to the idea of Abstract Domain/Abstract Algebra.
Abstract Domain

A mathematical structure/programming language module defining:

- A concrete semantic domain \( D \) (representing program computations)
- A set \( L = \) of encodings of computation properties
- A set of abstract operations, including:
  - a lattice structure: \( \leq \perp \top \sqcup \sqcap \)
  - (forward/backward) transformers \( f \in L^n \mapsto L \)
  - convergence accelerators \( \Delta \nabla \)
- A meaning \( \gamma \in L \mapsto \wp(D) \)

Example: Symbolic Execution

From [1, Sec. 3.4.5]:

\[
\begin{array}{c}
\{\langle Q_i, E_i \rangle | i \in \Delta_c \} \\
\prod_{c \in \text{Control}} \end{array}
\]

(\text{where } Q_i \text{ is a path condition and } E_i \text{ is a valuation in terms of initial values } \bar{x}) \text{ with concretization}

\[
\{\langle c, x \rangle | \exists \bar{x} : \bigvee_{i \in \Delta_c} Q_i(\bar{x}) \land x = E_i(\bar{x})\}
\]
Example : Symbolic Execution (Cont’d)

- Test transformer:

\[
\text{test}[b](\{\langle Q_i, E_i \rangle \mid i \in \Delta_c \}) = \\
\{\langle Q_i \land b[x| E_i(\bar{x})], E_i \rangle \mid i \in \Delta_c \}
\]

- Assignment transformer:

\[
\text{assign}[x := e(x)](\{\langle Q_i, E_i \rangle \mid i \in \Delta_c \}) = \\
\{\langle Q_i, e[x| E_i(\bar{x})] \rangle \mid i \in \Delta_c \}
\]

Example : Symbolic Execution (Cont’d)

- Program:

\[
\begin{array}{l}
(1) \text{tontue x} y \text{ faire} \\
(2) \text{x} = x \cdot y \\
(3) \text{refaire}; \\
(4)
\end{array}
\]

- Program transformer \(\mathcal{F}\):

\[
\begin{align*}
\mathcal{F}_1 &= \{<\text{vrai,x,y}>\} \\
\mathcal{F}_2 &= \text{test}(\lambda(x,y).x\cdot y)(\mathcal{F}_1 \cup \mathcal{F}_1) \\
\mathcal{F}_3 &= \text{affectation}(\lambda(x,y).x\cdot y,y)(\mathcal{F}_2) \\
\mathcal{F}_4 &= \text{test}(\lambda(x,y).x\cdot y)(\mathcal{F}_1 \cup \mathcal{F}_1)
\end{align*}
\]

Example : Symbolic Execution (Cont’d)

- Fixpoint iteration:

\[
\begin{align*}
\mathcal{F}_1 &= \{<\text{vrai,x,y}>\} \\
\mathcal{F}_2 &= \text{test}(\lambda(x,y).x\cdot y)(\mathcal{F}_1 \cup \mathcal{F}_1) = \{<\text{vrai,x,y}>\} \\
\mathcal{F}_3 &= \text{affectation}(\lambda(x,y).x\cdot y,y)(\mathcal{F}_2) = \{<\text{vrai,x,y}>\} \\
\mathcal{F}_4 &= \text{test}(\lambda(x,y).x\cdot y)(\mathcal{F}_1 \cup \mathcal{F}_1) = \{<\text{vrai,x,y}>\}
\end{align*}
\]

\[
\ldots
\]

---

Principle of Parametric Abstraction
Parametric Abstraction

- All abstract elements can be expressed in similar symbolic parametric form:

\[ L = \{ e(p) \mid p \in P \} \]

where the set \( P \) of parameters is either numerical or symbolic

- The fixpoint approximation \( \exists A \in L : \uparrow_p F \subseteq A \) that is the lattice constraint \( \exists p \in P : A = e(p) \land F(A) \subseteq A \) can be expressed as sufficient parametric constraints on the parameters \( p \in P \)

Solving the Parametric Constraints

- by sample executions (e.g. runtime generation of invariants [3])
- by random interpretation [4]
- by using constraint solvers (e.g. [5])

Example of Numerical Parametric Abstraction

Affine equalities Karr[76]

- Abstract domain:

\[ L = \{ \langle a_0, \ldots, a_n \rangle \mid \forall i = 0, \ldots, n : a_i \in \mathbb{R} \} \]

- Concretization:

\[ \gamma(\langle a_0, \ldots, a_n \rangle) = \{ \langle x_1, \ldots, x_n \rangle \mid a_0 + \sum_{i=0}^n a_i x_i = 0 \} \]

Example of Numerical Parametric Constraints

\[
\begin{align*}
\{ a_1 x + b_1 y + c_1 = 0 \} & \quad a_1 = b_1 = c_1 = 0 \\
\{ a_2 x + b_2 y + c_2 = 0 \} & \quad c_2 = 0 \\
\text{while} \ ?? \ \text{do} & \\
\{ a_3 x + b_3 y + c_3 = 0 \} & \quad a_3 = a_2 = a_5, b_3 = b_2 = b_5, \\
x := x+1 & \quad c_3 = c_2 = c_5 \\
\{ a_4 x + b_4 y + c_4 = 0 \} & \quad a_4 = a_3, b_4 = b_3, c_4 = c_3 - a_3 \\
y := y-1 & \\
\{ a_5 x + b_5 y + c_5 = 0 \} & \quad a_5 = a_4, b_5 = b_4, c_5 = c_4 + b_4 \\
\text{od} & \quad a_6 = a_2 = a_5, b_6 = b_2 = b_5, \\
\{ a_6 x + b_6 y + c_6 = 0 \} & \quad c_6 = c_2 = c_5
\end{align*}
\]

References


Solutions of the Example Parametric Constraints
for all \( a \in \mathbb{R} \):
\[
\begin{align*}
\{0x + 0y + 0 = 0\} \\
x:=0; \quad y:=0; \\
\{ax + ay + 0 = 0\} \\
\text{while } ?? \text{ do} \\
\{ax + ay + 0 = 0\} \\
x := x+1 \\
\{ax + ay - a = 0\} \\
y := y-1 \\
\{ax + ay + 0 = 0\} \\
o\d \\
\{ax + ay + 0 = 0\}
\end{align*}
\]

Other Examples of Numerical Parametric
Constraints Taken From VMCAI’05 and NSAD’05

– VMCAI’05:
- Jérôme Feret. *The arithmetic-geometric progression abstract domain*
- Sriram Sankaranarayanan, H.B. Spipma, Z. Manna. *Scalable Analysis of Linear Systems Using Mathematical Programming*

– NSAD’05:

---

**The Problem...**

- Find an example of execution satisfying given specifications
- **Examples:**
  - Automatic test data generation
  - Automatic generation of an alarm example
  - Automatic generation of a false alarm example (abstraction refinement)
Abstraction from Above and from Below

- Examples:
  - Over-approximation: invariance
  - Under-approximation: execution example

- Formally: dual

- What about under-approximation?
  - Finite state: trivial
  - Infinite state: nothing done in static analysis
    difficulty with dual widening/narrowing

---

Handling Tests

- Tests can be handled by case analysis
- Nondeterminism yield parametric symbolic execution trees:

  A; if (B) { C; } else { D; }; E

  \[
  \begin{array}{c}
  A \\
  \downarrow \\
  B \\
  \downarrow \\
  C \\
  \downarrow \\
  D \\
  \downarrow \\
  E
  \end{array}
  \]

  \[\neg B \]

  \[\neg B\]

  \[\neg B\]

  \[\neg B\]

  -> backtracking (e.g.)

---

Parametric Symbolic Execution

1: \( B := (X>=Y); \)
2: if (B) {
3: \( Y := 1 / X; \)
4: }
5:

- ASTREE signals a potential error at point 3: when \( X = 0 \)
- An iterated forward/backward polyhedral analysis yields a necessary path condition to reach point 3: with \( X = 0 \)

<table>
<thead>
<tr>
<th>Parametric trace</th>
<th>Path condition</th>
<th>Parameter constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( B_1, X_1, Y_1 )</td>
<td>( X_1 = 0 \land Y_1 \leq 0 )</td>
<td>( X_1 = X_2, Y_1 = Y_2 )</td>
</tr>
<tr>
<td>2: ( B_2, X_2, Y_2 )</td>
<td>( B_2 = true \land X_2 = 0 )</td>
<td>( B_2 = B_3, X_2 = X_3, Y_2 = Y_3 )</td>
</tr>
<tr>
<td>3: ( B_3, X_3, Y_3 )</td>
<td>( B_3 = true \land X_3 = 0 )</td>
<td>( B_3 = B_4, X_3 = X_4 )</td>
</tr>
<tr>
<td>4: ( B_4, X_4, Y_4 )</td>
<td>( B_4 = true \land X_4 = 0 )</td>
<td></td>
</tr>
</tbody>
</table>

Solution (a.o.): \( B_1 = true, X_1 = 0, Y_1 = -1000 \)

---

Handling loops: (1) by Syntactic Unrolling

1: while (X>0){
2: \( X = X-Y; \)
3: }
4: assert(X==0);

Solution (a.o.) with 2 loop unrollings:
- \( X_1 = 2, Y_1 = 1 \)
- \( X_3 = X_4, Y_3 = Y_4 \)
- \( X_5 = X_6 + Y_6, Y_5 = Y_6 \)
- \( X_7 = X_8, Y_7 = Y_8 \)
- \( X_9 = X_{10}, Y_9 = Y_{10} \)

<table>
<thead>
<tr>
<th>Param. trace</th>
<th>Path cond.</th>
<th>Parameter constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: ( X_1, Y_1 )</td>
<td>( X_1 &gt; 0, X_1 = X_2, Y_1 = Y_2 )</td>
<td></td>
</tr>
<tr>
<td>2: ( X_2, Y_2 )</td>
<td>( X_2 \geq Y_2 )</td>
<td>( X_2 = X_3 + Y_3, Y_2 = Y_3 )</td>
</tr>
<tr>
<td>3: ( X_3, Y_3 )</td>
<td>( X_3 \geq 0 )</td>
<td>( X_3 = X_4, Y_3 = Y_4 )</td>
</tr>
<tr>
<td>4: ( X_4, Y_4 )</td>
<td>( X_4 \geq 0 )</td>
<td>( X_4 = X_5, Y_4 = Y_5 )</td>
</tr>
<tr>
<td>5: ( X_5, Y_5 )</td>
<td>( X_5 \geq Y_5 )</td>
<td>( X_5 = X_6 + Y_6, Y_5 = Y_6 )</td>
</tr>
<tr>
<td>6: ( X_6, Y_6 )</td>
<td>( X_6 \geq 0 )</td>
<td>( X_6 = X_7, Y_6 = Y_7 )</td>
</tr>
<tr>
<td>7: ( X_7, Y_7 )</td>
<td>( X_7 &lt; Y_7, X_7 = X_8, Y_7 = Y_8 )</td>
<td></td>
</tr>
<tr>
<td>8: ( X_8, Y_8 )</td>
<td>( X_8 + 1 \leq Y_8 )</td>
<td>( X_8 = 0 )</td>
</tr>
</tbody>
</table>
Handling Loops: (2) by Bounded Syntactic Unrolling

- Add a distance (from origin/to end) extra parameter to path elements:
  \( \langle Q_0, E_0, 0 \rangle \langle Q_1, E_1, 1 \rangle \ldots \langle Q_{n-1}, E_{n-1}, n-1 \rangle \langle Q_n, E_n, n \rangle \)
- Consider the \( k \)-limiting parametric symbolic execution tree made up of all paths of length up to \( k \) and corresponding concrete constraints
- Strengthen by global reachability constraints and iterated forward/backward analysis of the symbolic execution tree
- Solve minimizing the path length

Handling Loops: (3) by Semantic Unrolling

1: while (X>0){
  Param_trace | Path cond. | Parameter constraints
  X = X-Y;
  1: (X_i, Y_i) | \( X_i > 0, X_i = X_{i+1}, Y_i = Y_{i+1} \)
  2: (X_2, Y_2) | \( X_2 \geq Y_2 \) | \( X_2 = X_1 + Y_3, Y_2 = Y_3 \)
  3: (X_3, Y_3) | \( X_3 \geq 0 \) | \( X_3 = X_2 + 1, Y_3 = Y_2 + 1 \)
  \cdots
  1: (X_n, Y_n) | \( X_n < Y_n \) | \( X_n = X_1, Y_n = Y_4 \)
  4: (X_4, Y_4) | \( X_4 + 1 \leq Y_4 \) | \( X_4 = 0 \)

Trial and error solvers choose \( n = 1, 2, 3, \ldots \) which amounts to loop unrolling. Forward/backward abstract interpretation? Random interpretation? Symbolic computation (à la Maple)?

Conclusion

- Very/extremely preliminary ongoing work
- More to do:
  - Think more about the formalization of parametric symbolic execution as an abstraction from below
  - Produce an implementation to allow for experimentation
  - Worry about floats\footnote{Rounding must be handled in the same way in the program and the solver} (symbolically, à la Miné [6]?)
  and very long loop unrollings

References


THE END, THANK YOU