« Parametric Abstraction »

Patrick Cousot
École normale supérieure
45 rue d'Ulm, 75230 Paris cedex 05, France

Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

NSAD'05 — Paris, France — Friday 21 Jan. 2005

Content

Abstract Domains

Static Analysis

– Static analysis computes an overapproximation A of an abstract semantics $\mathsf{lfp}^{\sqsubseteq}_{\perp} \mathcal{F} \sqsubseteq A$ where $F \in \mathcal{D} \mapsto \mathcal{D}$

– A compositional approach is preferable:

- The abstract domain $\mathcal{D}$ is defined by combination of elementary abstract domains L
- The abstract transformer $\mathcal{F}$ is defined inductively (e.g. by induction on the program syntax) by composition of elementary abstract tranformers f ...

This structure $\langle L, \sqsubseteq, \perp, \ldots, f\rangle\ldots$ 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: $\sqsubseteq \ \perp \ \top \ \sqcup \ \sqcap$
- (forward/backward) transformers $f \in L^n \mapsto L$
- convergence accelerators $\Delta \ \nabla$

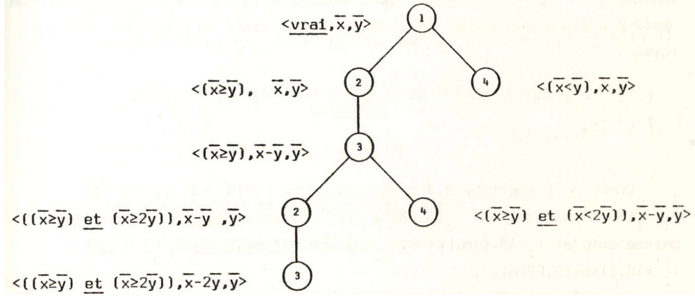
– a meaning $\gamma \in L \mapsto \wp(D)$

Example : Symbolic Execution

From [1, Sec. 3.4.5]:

```
{1}  ·
     tantque x≥y faire
{2}
         x:=x-y;
{3}
     refaire;
{4}
```

Program



Symbolic execution tree

References

[1] P. Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes.* Thèse d'État ès sciences mathématiques, Université scientifique et médicale de Grenoble, Grenoble, 21 mars 1978.

[2] J.C. King. Symbolic Execution and Program Testing, CACM 19:7(385–394), 1976.

Symbolic Execution

Example : Symbolic Execution (Cont'd)

– An abstract interpretation

– The abstract properties of L have the form:

$$\prod_{c \in \text{Control}} \{\langle Q_i,\ E_i\rangle \mid i \in \Delta_c\}$$

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

$$\{\langle c,\ x\rangle \mid \exists \bar{x} : \bigvee_{i \in \Delta_c} Q_i(\bar{x}) \wedge x = E_i(\bar{x})\}$$

Example : Symbolic Execution (Cont'd)

– Test transformer:

$$\texttt{test}[\![b]\!](\{\langle Q_i,\ E_i\rangle \mid i \in \Delta_c\}) = \{\langle Q_i \wedge b[x\backslash E_i(\bar{x})],\ E_i\rangle \mid i \in \Delta_c\}$$

– Assignment transformer:

$$\texttt{assign}[\![x := e(x)]\!](\{\langle Q_i,\ E_i\rangle \mid i \in \Delta_c\}) = \{\langle Q_i,\ e[x\backslash E_i(\bar{x})]\rangle \mid i \in \Delta_c\}$$

Example : Symbolic Execution (Cont'd)

– Program:

```
{1}
     tantque x≥y faire
{2}
          x:=x-y;
{3}
     refaire;
{4}
```

– Program transformer $\mathcal{F}$:

$$\left\{\begin{array}{l} P_1 = \{\langle \underline{\text{vrai}}, \bar{x}, \bar{y}\rangle\} \\ P_2 = \underline{\text{test}}(\lambda(x,y).[x \geq y])(P_1\ \underline{\text{ou}}\ P_3) \\ P_3 = \underline{\text{affectation}}(\lambda(x,y).[x-y,y])(P_2) \\ P_4 = \underline{\text{test}}(\lambda(x,y).[x<y])(P_1\ \underline{\text{ou}}\ P_3) \end{array}\right.$$

Example : Symbolic Execution (Cont'd)

– Fixpoint iteration:

$$\left[\ P_i^0 = \emptyset \quad (i=1,\ldots,4)\right.$$

$$\left[\begin{array}{l} P_1^1 = \{\langle \underline{\text{vrai}}, \bar{x}, \bar{y}\rangle\} \\ P_2^1 = \underline{\text{test}}(\lambda(x,y).[x \geq y])(P_1^1\ \underline{\text{ou}}\ P_3^0) = \{\langle (\bar{x} \geq \bar{y}), \bar{x}, \bar{y}\rangle\} \\ P_3^1 = \underline{\text{affectation}}(\lambda(x,y).[x-y,y])(P_2^1) = \{\langle (\bar{x} \geq \bar{y}), \bar{x}-\bar{y}, \bar{y}\rangle\} \\ P_4^1 = \underline{\text{test}}(\lambda(x,y).[x<y])(P_1^1\ \underline{\text{ou}}\ P_3^0) = \{\langle (\bar{x} < \bar{y}), \bar{x}, \bar{y}\rangle\} \end{array}\right.$$

$$\left[\begin{array}{l} P_1^2 = \{\langle \underline{\text{vrai}}, \bar{x}, \bar{y}\rangle\} \\ P_2^2 = \{\langle (\bar{x} \geq \bar{y}), \bar{x}, \bar{y}\rangle\ ,\ \langle ((\bar{x} \geq \bar{y})\ \underline{\text{et}}\ (\bar{x} \geq 2\bar{y})), \bar{x}-\bar{y}, \bar{y}\rangle\} \\ P_3^2 = \{\langle (\bar{x} \geq \bar{y}), \bar{x}-\bar{y}, \bar{y}\rangle\ ,\ \langle ((\bar{x} \geq \bar{y})\ \underline{\text{et}}\ (\bar{x} \geq 2\bar{y})), \bar{x}-2\bar{y}, \bar{y}\rangle\} \\ P_4^2 = \{\langle (\bar{x} < \bar{y}), \bar{x}, \bar{y}\rangle\ ,\ \langle (\bar{x} < 2\bar{y}), \bar{x}-\bar{y}, \bar{y}\rangle\} \end{array}\right.$$

...

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 $\exists A \in L : \mathsf{lfp}\, F \sqsubseteq A$ that is the lattice constraint $\exists p \in P : A = e(p) \wedge F(A) \sqsubseteq A$ 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])

References

[3] M.D. Ernst, J. Cockrell, W.G. Griswold and D. Notkin. Dynamically Discovering Likely Program Invariants to Support Program Evolution. IEEE Transactions on Software Engineering, v.27 n.2, p.99–123, February 2001

[4] S. Gulwani and G.C. Necula. Discovering affine equalities using random interpretation. 30th ACM POPL, p.74–84, January 2003

[5] A. Aiken. Introduction to Set Constraint-Based Program Analysis. SCP 35(1999):79-111, 1999.

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

$\{a_1x + b_1y + c_1 = 0\}$	$a_1 = b_1 = c_1 = 0$
`x:=0; y:=0;`	
$\{a_2x + b_2y + c_2 = 0\}$	$c_2 = 0$
`while ?? do`	
$\{a_3x + b_3y + c_3 = 0\}$	$a_3 = a_2 = a_5,\ b_3 = b_2 = b_5,$
`x := x+1`	$c_3 = c_2 = c_5$
$\{a_4x + b_4y + c_4 = 0\}$	$a_4 = a_3,\ b_4 = b_3,\ c_4 = c_3 - a_3$
`y := y-1`	
$\{a_5x + b_5y + c_5 = 0\}$	$a_5 = a_4,\ b_5 = b_4,\ c_5 = c_4 + b_4$
`od`	$a_6 = a_2 = a_5,\ b_6 = b_2 = b_5,$
$\{a_6x + b_6y + c_6 = 0\}$	$c_6 = c_2 = c_5$

Solutions of the Example Parametric Constraints

for all $a \in \mathbb{R}$:

$\{0x + 0y + 0 = 0\}$
```
x:=0; y:=0;
```
$\{ax + ay + 0 = 0\}$
```
while ?? do
```
$\{ax + ay + 0 = 0\}$
```
  x := x+1
```
$\{ax + ay - a = 0\}$
```
  y := y-1
```
$\{ax + ay + 0 = 0\}$
```
od
```
$\{ax + ay + 0 = 0\}$

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:
  - H. Seidl, M. Petter. *Infering polynomial invariants with Polyinvar.*

Example of Application to the Generation of Execution Examples

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

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

Parametric trace	Path condition	Parameter constraints
$1{:}\langle B_1, X_1, Y_1\rangle$	$X_1 = 0 \wedge Y_1 \leq 0$	$X_1 = X_2, Y_1 = Y_2$
$2{:}\langle B_2, X_2, Y_2\rangle$	$B_2 = true \wedge X_2 = 0$	$B_2 = B_3, X_2 = X_3, Y_2 = Y_3$
$3{:}\langle B_3, X_3, Y_3\rangle$	$B_3 = true \wedge X_3 = 0$	$B_3 = B_4, X_3 = X_4$
$4{:}\langle B_4, X_4, Y_4\rangle$	$B_4 = true \wedge X_4 = 0$	

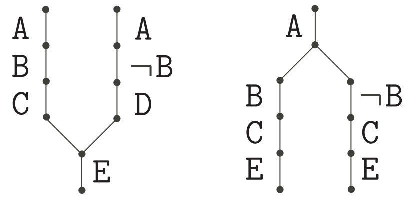
Solution (a.o.): $B_1 = \text{true}$, $X_1 = 0$, $Y_1 = -1000$

Handling Tests

– Tests can be handled by case analysis

– Nondeterminism yield parametric symbolic execution trees:

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



⟶ backtracking (e.g.)

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: $X_1 = 2$, $Y_1 = 1$

Param. trace	Path cond.	Parameter constraints
$1{:}\langle X_1, Y_1\rangle$		$X_1 > 0,\ X_1 = X_2,\ Y_1 = Y_2$
$2{:}\langle X_2, Y_2\rangle$	$X_2 \geq Y_2$	$X_2 = X_3 + Y_3,\ Y_2 = Y_3$
$3{:}\langle X_3, Y_3\rangle$	$X_3 \geq 0$	$X_3 = X_4,\ Y_3 = Y_4$
$1{:}\langle X_4, Y_4\rangle$		$X_4 > 0,\ X_4 = X_5,\ Y_4 = Y_5$
$2{:}\langle X_5, Y_5\rangle$	$X_5 \geq Y_5$	$X_5 = X_6 + Y_6,\ Y_5 = Y_6$
$3{:}\langle X_6, Y_6\rangle$	$X_6 \geq 0$	$X_6 = X_7,\ Y_6 = Y_7$
$1{:}\langle X_7, Y_7\rangle$		$X_7 < Y_7,\ X_7 = X_8,\ Y_7 = Y_8$
$4{:}\langle X_8, Y_8\rangle$	$X_8 + 1 \leq Y_8$	$X_8 = 0$

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){
2:    X = X-Y;
3: }
4: assert(X==0);
```

Param. trace	Path cond.	Parameter constraints
$1{:}\langle X_1^i,\ Y_1^i\rangle$		$X_1^i > 0,\ X_1^i = X_2^i,\ Y_1^i = Y_2^i$
$2{:}\langle X_2^i,\ Y_2^i\rangle$	$X_2^i \geq Y_2^i$	$X_2^i = X_3^i + Y_3^i,\ Y_2^i = Y_3^i$
$3{:}\langle X_3^i,\ Y_3^i\rangle$	$X_3^i \geq 0$	$X_3^i = X_1^{i+1},\ Y_3^i = Y_1^{i+1}$
$\ldots$	$i = 0 \ldots n$	
$1{:}\langle X_1^n,\ Y_1^n\rangle$		$X_1^n < Y_1^n,\ X_1^n = X_4,\ Y_1^n = Y_4$
$4{:}\langle X_4,\ Y_4\rangle$	$X_4 + 1 \leq Y_4$	$X_4 = 0$

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 [1] (symbolically, à la Miné [6]?) and very long loop unrollings

References

[6] A. Miné. Relational abstract domains for the detection of floating-point run-time errors. ESOP'04, LNCS 2986, p. 3–17, Springer.

THE END, THANK YOU

[1] Rounding must be handled in the same way in the program and the solver