« Combination of Abstractions in the ASTRÉE Static Analyzer »

Patrick Cousot

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

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

Radhia Cousot

CNRS & École polytechnique
Route de Saclay
91440 Palaiseau Cedex, France

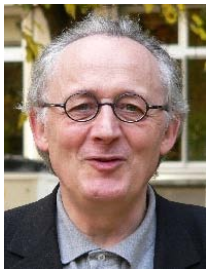
Radhia.Cousot@polytechnique.fr
www.enseignement.polytechnique.fr/profs/informatique/Radhia.Cousot/

Visiting IBM T.J. Watson Research Center —— Hawthorne N.Y.

2007 Programming Language Day at Watson — Hawthorne
Monday May 7th, 2007

Project Members

Bruno Blanchet [1]



Patrick Cousot

Radhia Cousot

Jérôme Feret

Laurent Mauborgne

Antoine Miné

David Monniaux

Xavier Rival

[1] Nov. 2001 — Nov. 2003.

ASTRÉE

ASTRÉE objectives

– ASTRÉE is a static program analyzer [2, 3, 7, 10] based on abstract interpretation [5, 6]

– ASTRÉE used to prove automatically the absence of run time errors in C programs [2]

– ASTRÉE is applied to large industrial embedded control/command safety-critical synchronous real-time software

– ASTRÉE is efficient and very precise for its application domain

– ASTRÉE was recently extended [13] to handle other kinds (e.g. communication) of embedded software, some of which are hand-written (using union, pointer arithmetics, etc).

[2] without recursion, dynamic memory allocation and library calls.

ASTRÉE design

ASTRÉE was designed using:

– a syntax-directed representation of the program control flow (functions, block structures);

– functional representation of abstract environments with sharing [2], for memory and time efficiency, and limited support for analysis parallelization [15];

– basic abstract domains, tracking variables independently (integer and floating-point intervals [4] using staged widenings);

– relational abstract domains tracking dependencies between variables

  – symbolic computation and linearization of expressions [12],

  – packed octagons [14],

  - application-aware domains (such as the ellipsoid abstract domain for digital filters [8] or the arithmetic-geometric progression abstract domain [9], e.g. to bound potentially diverging computations);
- abstract domains tracking dependencies between boolean variables and other variables (boolean partitioning domain [3]), or the history of control flow branches and values along the execution trace (trace partitioning domain [11]);
- a memory abstract domain [3] recently extended to cope with unions and pointer arithmetics [3] [13].
- abstract domains are parametrized [4] and applied locally [5] [3].

[3] Contrary to many program analysis systems, ASTRÉE does not have separate phases for pointer/aliasing analysis and arithmetic analysis.

[4] e.g. maximal height of decision trees.

[5] e.g. to variables packs.

Motivation for the combination of abstract domains

Different Classes of Run-time Errors

1. Errors terminating the execution [6]. ASTRÉE warns and continues by taking into account only the executions that did not trigger the error.
2. Errors not terminating the execution with predictable outcome [7]. ASTRÉE warns and continues with worst-case assumptions.
3. Errors not terminating the execution with unpredictable outcome [8]. ASTRÉE warns and continues by taking into account only the executions that did not trigger the error.

3. is sound with respect to C standard, unsound with respect to C implementation, unless no false alarm.

[6] floating-point exceptions e.g. (invalid operations, overflows, etc.) when traps are activated

[7] e.g. overflows over signed integers resulting in some signed integer.

[8] e.g. memory corruptionss.

Origins of False Alarms

1. Imprecise abstract transformer $\Rightarrow$ fix algorithm
2. Imprecise parametrization $\Rightarrow$ fix parameters
3. Imprecise widening $\Rightarrow$ fix algorithm
4. Inepressive combination of abstract domains $\Rightarrow$
   - must introduce new abstract domain
   - without redesigning existing abstract domains
   - but with interactions and reductions with existing abstract domains to enhance global precision

Abstract domains

Abstract domains

– Abstract domain role: encapsulate a class of program properties

– Abstract domain structure:

  - Abstract sets of execution trace fragments, as defined by a concretization function
  - Data structures for abstract properties encoding concrete properties of trace fragments
  - Sound algorithms for logical structure (approximation preorder, etc.)
  - Sound algorithms for transformers (post-image, etc.)
  - Sound extrapolators (widening/narrowing)
  - Communication channels to implement an approximate reduced product of abstractions

Abstract domain constructors

– Non-relational lifting: non-relational abstract domain D on values $\mapsto\!\!\circ\!\!\rightarrow$ abstract domain $\prod_{X \in \mathsf{Var}} D$ on all variables (using balanced binary trees)

– Relational lifting: relational abstract domain $D(P)$ on packs $P \subseteq \mathsf{Var}$ of variables $\mapsto\!\!\circ\!\!\rightarrow$ abstract domain $D(\mathsf{Var})$ on all variables

– Trace partitionning: past history abstraction domain [9] $\times$ current memory state abstraction domain $\mapsto\!\!\circ\!\!\rightarrow$ prefix traces abstraction domain (using maps implemented as trees) [11]

– Boolean partitionning: prefix traces abstraction domain $\times$ boolean variables $\mapsto\!\!\circ\!\!\rightarrow$ prefix traces abstraction domain (using decision trees)

[9] (e.g. a sub-sequence of branches taken

Reductions

Precondition refinement

– Use information provided by other abstract domains to improve the precondition of a transformer

– Example: ellipsoid domain for filters:

$$\underset{\text{precondition}}{Y^2 - a.Y.Z - b.Z^2 \leqslant k^2} \xrightarrow[X = a'.Y. + b'.Z + c']{\text{assignment}} \underset{\text{postcondition}}{X^2 - a''.X.Y - b''.Y^2 \leqslant (f(k))^2}$$

– If no precondition of the required form is available, so synthesize one from
  - the interval domain (for Y and Z)
  - the symbolic domain (for $Y = Z$?)

Postcondition refinement

– Use information provided by other abstract domains to improve the postcondition of a transformer

– Two cases:
  - Information computed by a domain propagated to its underlying domains
  - Information resquested by a domain missing the information

Example of postcondition refinement of underlying domains

Code fragment computing an absolute value:

Program	Intervals	Octagons
	$\mathtt{Y} \in [-\infty, +\infty]$	
`X=Y;`	$\mathtt{X}, \mathtt{Y} \in [-\infty, +\infty]$	$\mathtt{X} = \mathtt{Y}$
`if (X<0) {`	$\mathtt{X} \in [-\infty, -1], \mathtt{Y} \in [-\infty, +\infty]$	$\mathtt{X} = \mathtt{Y} < 0$
`  X=-Y;`	$\mathtt{X}, \mathtt{Y} \in [-\infty, +\infty]$	$\mathtt{X} = -\mathtt{Y} > 0$
`} else {`	$\mathtt{X} \in [0, +\infty], \mathtt{Y} \in [-\infty, +\infty]$	$\mathtt{X} = \mathtt{Y} \geqslant 0$
`}`	$\mathtt{X} \in [-\infty, +\infty], \mathtt{Y} \in [-\infty, +\infty]$	$-\mathtt{X} \leqslant \mathtt{Y} \leqslant \mathtt{X}$
`if (X<100){`	$\mathtt{X} \in [-\infty, 99], \mathtt{Y} \in [-\infty, +\infty]$	$-\mathtt{X} \leqslant \mathtt{Y} \wedge \mathtt{Y} \leqslant \mathtt{X} \leqslant 99$
`  ...Y...`		
`}`		

Intervals with postcondition refinement by octagons: the octagons domains receives ranges of variables from interval domain, reduce them and send back the result to the interval domain.

Program	Reduced ntervals	Octagons
	$\mathtt{Y} \in [-\infty, +\infty]$	
`X=Y;`	$\mathtt{X}, \mathtt{Y} \in [-\infty, +\infty]$	$\mathtt{X} = \mathtt{Y}$
`if (X<0) {`	$\mathtt{X} \in [-\infty, -1], \mathtt{Y} \in [-\infty, -1]$	$\mathtt{X} = \mathtt{Y} < 0$
`  X=-Y;`	$\mathtt{X} \in [1, +\infty], \mathtt{Y} \in [-\infty, -1]$	$\mathtt{X} = -\mathtt{Y} > 0$
`} else {`	$\mathtt{X} \in [0, +\infty], \mathtt{Y} \in [0, +\infty]$	$\mathtt{X} = \mathtt{Y} \geqslant 0$
`}`	$\mathtt{X} \in [0, +\infty], \mathtt{Y} \in [-\infty, +\infty]$	$-\mathtt{X} \leqslant \mathtt{Y} \leqslant \mathtt{X}$
`if (X<100){`	$\mathtt{X} \in [0, 99], \mathtt{Y} \in [-99, 99]$	$-\mathtt{X} \leqslant \mathtt{Y} \wedge \mathtt{Y} \leqslant \mathtt{X} \leqslant 99$
`  ...Y...`		
`}`		

Example of information request

Analysis with the octagon domain

```
{ Y <= X <= Z }
X = A
{ X = A, Y <= Z }
```

– Nothing is known on `A` whence on `X`

– Ask for an interval of `A` or `X` to the interval domain [10]

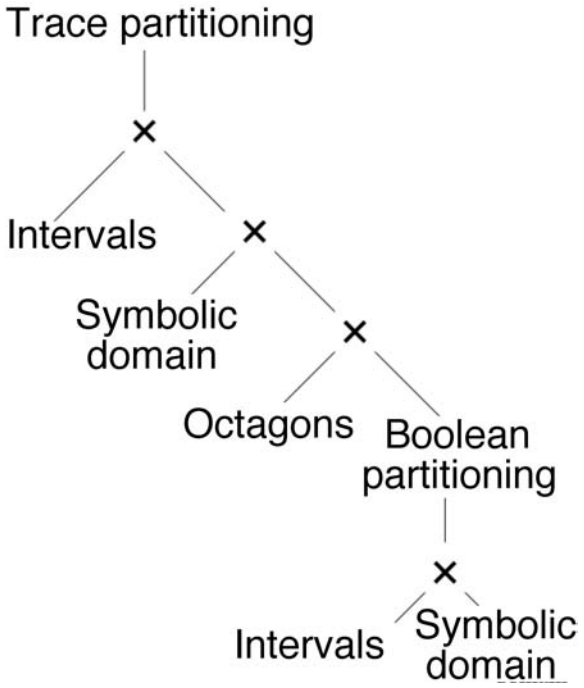
– Because the abstract domains are organized into a hierarchy, circularities are avoided

[10] The octagon domain is more precise than the interval domain but the interval domain might have benefitted from reductions from other domains.

Implementation as a networks of abstract domains

Hierarchies of abstract domains

– Abstract domains are organized in several hierarchies, e.g.



– Bottom-up evaluation

– Communication of abstract properties via common ancestor by communication primitives

Abstract domain of messages

– set $IO^\sharp$ of abstract properties (messages sent/received by communication primitives)

– concretization $\gamma_{IO^\sharp} : IO^\sharp \to D$ (D is the set of concrete execution trace fragments)

– communication channels

  - $\top$: initially, no information on channel
  - content read and modified by communication primitives

Communication primitives

Each abstract domain $(D^\sharp, \gamma_{D^\sharp})$ has two communication primitives:

– Emit abstract information on communication channels:
  – $\textsc{extract}_{D^\sharp} : D^\sharp \times IO^\sharp \to IO^\sharp$
  – $io' = \textsc{extract}_{D^\sharp}(c, io)$:
    - io: contents of the output channel before the constraint is emitted,
    - c: abstract element in the abstract domain $D^\sharp$,
    - io': content of the channel reduced by information extracted from c (hence the name EXTRACT).
  – $\gamma_{D^\sharp}(c) \cap \gamma_{IO^\sharp}(io) \subseteq \gamma_{IO^\sharp}(\textsc{extract}_{D^\sharp}(c, io))$ (reduction of communication channel)

– Receive abstract information on communication channel:
  – $\textsc{refine}_{D^\sharp} : D^\sharp \times IO^\sharp \to D^\sharp$
  – $c' = \textsc{refine}_{D^\sharp}(c, io)$:
    - c: element of the abstract domain
    - io: abstract information on the input channel
    - c': refinement for c by the received abstract information (hence the name REFINE)
  – $\gamma_{D^\sharp}(c) \cap \gamma_{IO^\sharp}(io) \subseteq \gamma_{D^\sharp}(\textsc{refine}_{D^\sharp}(c, io))$ (reduction of abstract property)

Improving the precision of widenings

Widening precision improvement strategies

– Delaying widenings: Replace ∇ by $\cup$ finitely often

– Reducing widenings: Reduce result of ∇ using information provided by other abstract domains

Delaying widenings

Problem with the delaying of widenings

– Let ∇ be a widening on an abstract domain D

– Replacing ∇ by $\cup$ improves precision

– However, fair replacement [11] of ∇ by $\cup$ may no longer ensure termination

[11] any infinite iteration would have an unbounded number of widenings

Counter-example

- $D^\sharp \triangleq \{S \subseteq [0;1] \mid 0, 1 \in S\}$
- $f(S) = S \cup \{\frac{1}{2^{n+1}} \mid \frac{1}{2^n} \in S\}$
- $a \nabla_{D^\sharp} b = \rho(a, a \cup b)$
- $\rho(a, b)$ is obtained by making the convex union of several connected components of b until there are fewer connected components than in a, and fewer than five connected components
- $\nabla_{D^\sharp}$ is a widening [12]

[12] along the abstract iterates the number of connected components decreases until it reaches 1, and the interval $[0; 1]$ is the only element with one connected component.

– The sequence:

$$\begin{cases} u_0 = \{0; 1\}, \\ u_{2n} = u_{2n-1} \nabla_{D\sharp} f(u_{2n-1}), \\ u_{2n+1} = u_{2n} \cup f(u_{2n}) \end{cases}$$

is $u_{2n} = \{0\} \cup [\frac{1}{2^{2n}}; 1]$ and $u_{2n+1} = \{0; \frac{1}{2^{2n+1}}\} \cup [\frac{1}{2^{2n}}; 1]$.

– Whence it is not ultimately stationary.

Reducing widenings

Problem with the reduction of widenings

– Let ∇_1 be a widening on an abstract domain D_1

– Let ∇_2 be a widening on an abstract domain D_2

– $\nabla_1 \times \nabla_2$ is a widening on the product $D_1 \times D_2$ [13]

– However, $\nabla_1 \times \nabla_2$ may no longer ensure termination of the analysis in case of interdomain reductions

[13] interpreted as a conjunction in the concrete

Counter-example

```
{ A:[100, 100], X:[0,0], Y:[0, +oo] }
while (X < Y) { X = X + 1 }
```

$[0,0] \cup ((X \cap [-\infty, \max(Y)]) + [1,1]) \subseteq X$ inequation $F(X) \subseteq X$

$X_0 = \bot, \quad X_n = X_{n-1} \triangledown F(X_{n-1})$ until $F(X_n) \subseteq X_n$

Iterates with widening and no reduction:

$$
\begin{aligned}
X &= \bot \\
X &= \bot \triangledown [0,0] = [0,0] \\
X &= [0,0] \triangledown [0,1] = [0,+\infty] \\
X &= [0,+\infty] \triangledown [0,+\infty] = [0,+\infty] \text{ stable}
\end{aligned}
$$

Widening always enforces termination

```
{ A:[100, 100], X:[0,0], Y:[0, +oo] }
while (X < Y) { X = X + 1 }
```

$[0,0] \cup ((X \cap [-\infty, \max(Y)]) + [1,1]) \subseteq X$ inequation $F(X) \subseteq X$

Iterates with widening and reduction by $X \leqslant A$:

$$
\begin{array}{lll}
X = \bot & & \\
X = \bot \triangledown [0,0] = [0,0] & & \\
X = [0,0] \triangledown [0,1] = [0,+\infty] & & \\
X \to [0,99] & & \text{reduction by octagons} \\
X = [0,99] \triangledown [0,100] = [0,+\infty] & & \text{unstable } [0,+\infty] \not\subseteq [0,99] \\
X \to [0,99] & & \text{reduction by octagons} \\
X = [0,99] \triangledown [0,100] = [0,+\infty] & & \text{unstable } [0,+\infty] \not\subseteq [0,99]
\end{array}
$$

$\ldots$

Widening no longer enforces termination because of reductions

Convergence enforcement with delayed and reduced widenings

Abstract domains hierarchy

– The analysis is performed using a finite product

$$D^\sharp = \prod_{i \in I} D_i^\sharp$$

of totally ordered abstract domains $(D_i^\sharp, \sqsubseteq_i^\sharp)_{i \in I}$ [14]

– Each $D_i^\sharp$ has a widening operator ∇_i

– The infimum is $\bot = (\bot_i)_{i \in I}$

– Let $\mathrm{F}_{D^\sharp} \in D^\sharp \to D^\sharp$ be the fixpoint transformer

[14] A domain is larger than its underlying domains.

Strengthened definition of widenings

Each ∇_i on $D_i^\sharp$ is such that:

- for any $a, b \in D_i^\sharp$, we have both $a \sqsubseteq_i^\sharp a \nabla_i b$ and $b \sqsubseteq_i^\sharp a \nabla_i b$
- Define the relation $\rightarrow_i'$ [15] as

  "for any $a, d \in D_i^\sharp$, $a \rightarrow_i d$ if and only if there exist $b, c \in D_i^\sharp$ such that $a \sqsubseteq_i^\sharp b$ and $d = b \nabla_i c$"
- $\rightarrow_i'$ is well-founded.

[15] read $a \rightarrow_i d$ as "a can be widened to d"

Iterates with widenings

– Let $((b_i)_n) \in (\{\mathit{false}; \mathit{true}\}^I)^{\mathbb{N}}$ be a family of booleans such that $\forall i \in I$, the sequence $((b_i)_n)_{n\in\mathbb{N}}$ takes the value true an unbounded number of times.

– Let $(\rho_i)_{i\in I} \in D^\sharp$ be a finite family of abstract elements.

– Then, the sequence:

$$\begin{cases} (w_i)_0 = \bot_i, \\ u_{n+1} = \mathrm{F}_{D^\sharp}(w_n), \\ (v_i)_{n+1} = \begin{cases} \max_{\sqsubseteq_i^\sharp}((w_i)_n, (u_i)_{n+1}) & \text{whenever } (b_i)_n = \mathit{false}, \\ (w_i)_n \nabla_i (u_i)_{n+1} & \text{otherwise ,} \end{cases} \\ (w_i)_{n+1} = \mathrm{m\ n}_{\sqsubseteq_i^\sharp}((v_i)_{n+1}, \rho_i), \end{cases}$$

is ultimately stationary (and sound).

Convergence of the analysis

– To avoid cyclic reductions between components of the product domain: after a widening step, a domain can only refine its underlying domains.

– The abstract iterates in the abstract domains that are at the top of the hierarchy are ultimately stationary.

– Once the abstract properties in the domains that are above one domain are stable, the reduction of abstract properties in this domain can be seen as an intersection with a constant abstract property.

– Thus, its abstract iterates are ultimately stationary.

Conclusion

The power of widenings

– Infinite abstractions with widenings are provably superior to finite abstractions
– The precision of widenings can be improved by fairly delaying its applications and reducing its extrapolations (while still ensuring convergence)
– The abstract domains can be organized in hierarchical networks to minimize programming efforts when introducing new abstractions and reductions in the static analyzer

THE END, THANK YOU

Reference

Presentation based on:

[1] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, & X. Rival. Combination of Abstractions in the ASTRÉE Static Analyzer. In 11th Annual Asian Computing Science Conference (ASIAN'06), National Center of Sciences, Tokyo, Japan, December 6—8, 2006. LNCS, Springer (to appear).

Bibliography

[2] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software, invited chapter. In T. Mogensen, D.A. Schmidt, and I.H. Sudborough, editors, *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, LNCS 2566, pages 85–108. Springer, 2002.

[3] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proc. ACM SIGPLAN '2003 Conf. PLDI*, pages 196–207, San Diego, 2003. ACM Press.

[4] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106–130, Paris, France, 1976. Dunod, Paris, France.

[5] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In 4^{th} *ACM POPL*, pages 238–252, January 1977.

[6] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, August 1992.

[7] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE analyzer. In M. Sagiv, editor, *Proc. 14th ESOP '2005, Edinburgh*, LNCS 3444, pages 21–30. Springer, 4–8 Apr. 2005.

[8] J. Feret. Static analysis of digital filters. In D. Schmidt, editor, *Proc. 30th ESOP '2004, Barcelona*, LNCS 2986, pages 33–48. Springer, Mar. 27 – Apr. 4, 2004.

[9] J. Feret. The arithmetic-geometric progression abstract domain. In R. Cousot, editor, *Proc. 6th VMCAI '2005, Paris*, LNCS 3385, pages 2–58. Springer, Jan. 17–19, 2005.

[10] L. Mauborgne. ASTRÉE: Verification of absence of run-time error. In P. Jacquart, editor, *Building the Information Society*, chapter 4, pages 385–392. Kluwer Academic Publishers, 2004.

[11] L. Mauborgne and X. Rival. Trace partitioning in abstract interpretation based static analyzers. In M. Sagiv, editor, *Proc. 14th ESOP '2005, Edinburgh*, LNCS 3444, pages 21–30. Springer, 4–8 Apr. 2005.

[12] A. Miné. Symbolic methods to enhance the precision of numerical abstract domains. In *VMCAI'06*, volume 3855 of *LNCS*, pages 348–363. Springer, 2002.

[13] A. Miné. Field-sensitive value analysis of embedded C programs with union types and pointer arithmetics. In *Proc. LCTES 2006*, Ottawa, Ontario, Canada, 14–16 June 2006, pages 54–63. ACM Press, 2006.

[14] A. Miné. The octagon abstract domain. *Higher-Order and Symbolic Computation*, 19:31–100, 2006.

[15] David Monniaux. The parallel implementation of the ASTRÉE static analyzer. In Kwangkeun Yi, editor, *APLAS*, volume 3780 of *LNCS*. Springer, 2005.

