# Calculational Design of Semantics and Static Analyzers by Abstract Interpretation


Patrick Cousot

École Normale Supérieure
DMI, 45, rue d'Ulm
75230 Paris cedex 05
France
cousot@dmi.ens.fr
http://www.dmi.ens.fr/~cousot



NATO International Summer School 1998 on
*Calculational System Design*
Marktoberdorf, Germany
Organized by F.L. Bauer, M. Broy, E.W. Dijkstra,
D. Gries and C.A.R. Hoare.
28 July – 9 August, 1998



**Abstract**

The five 45mn lectures are conceived as an elementary introduction to *abstract interpretation* [9, 11]. Taking a pragmatic point of view, we formally design and implement a static analyzer for a simple imperative programming language by abstract interpretation of an operational semantics.


## 1 Design of a hierarchy of semantics

The lexer and parser transform the input program in concrete syntax into abstract syntax. The small step rewriting operational semantics is shown to be equivalent to the use of labels to designate program points. This is the basis for defining an execution trace semantics from which we formally design, by abstract interpretation, a hierarchy of semantics, including the weakest precondition and strongest postcondition semantics [5].

## 2 Design of a basic generic abstract interpreter

A basic generic abstract interpreter is formally designed by approximating the strongest postcondition semantics. The development is by stepwise refinements and approximation of the fixpoint semantics, as follows:

1. approximation of the strongest postcondition semantics by the strongest forward invariant semantics (by disregarding the nonterminating behaviors so as to consider safety properties only);

2. isomorphic decomposition into local invariants by partitioning according to program points to get a system of fixpoint equations [4];

3. non-relational approximation of the local invariants parameterized by a domain of abstract values approximating sets of concrete values of variables to get an attribute-independent abstract interpretation [4];

4. chaotic iteration strategies [10] for solving the fixpoint equations for abstract domains satisfying the ascending chain condition;

The basic generic abstract interpreter is designed in CAML. A very simple application to initialization and sign analysis is considered.

## 3 Design of abstract domains

The *reduced product* [11] of abstract interpretations is the basic way of composing independent abstraction. A generic implementation of the reduced product can be obtained by considering the *direct product* enhanced with *local decreasing iterations* [24], an idea which is also applicable to improve the precision of the analysis of conditionals.

The *disjunctive completion* of an abstract domain consists in including the concrete disjunctions missing in the abstract domain [11]. A generic implementation is considered for atomistic finite abstract domains.

It is shown how sophisticated abstract domains can be designed by composition (using reduced products) and enhancement (using disjunctive completion) of simple abstract domains. Several such abstract domains are implemented (see [23] for other examples).

## 4 Combined forward and backward analyzes

The weakest precondition and strongest postcondition semantics are equivalent [5], but this is no longer true after approximation. So the same abstraction may lead to different results when considering forward and backward analyzes. Consequently the basic generic abstract interpreter of Sec. 2 is extended to include backward analyzes.

Then forward and backward analyzes are combined iteratively [3, 12, 18].

An application to abstract debugging [1] is considered.

## 5 Infinitary abstract interpretations

The most precise static program analyzes are obtained by abstract interpretations involving infinite abstract domains not satisfying the ascending chain condition [14]. Then the convergence of the upwards iteration for the system of fixpoint equations must be enforced using *widenings*. The post-fixpoint approximation of the least fixpoint can then be improved using a downwards iteration with *narrowing* [9]. The classical example of intervals is implemented [8] with an efficient chaotic iteration strategy [2]. Other classical examples involve e.g. linear inequalities [19], linear congruences [25, 26].

## Lecture notes

The lecture notes [7] are extracted from [12]. [14] is also recommended as introductory reading. Those interested in logic programming may read [14, 20] while several analyzes related to functional programming are considered in [15, 16, 6]. See e.g. for

Non-numerical abstract domains are considered e.g. in [17]. Such abstract domains are particularly important for pointer analysis [21, 27], analysis of communication topologies [28], etc.

From a more theoretical point of view, several models of abstraction are discussed in [13]. A practical application of program static analysis by abstract interpretation is briefly discussed in [22].

## Bibliography


[1] F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *Proc. PLDI*, pages 46–55. ACM Press, 1993.

[2] F. Bourdoncle. Efficient chaotic iteration strategies with widenings. In D. Bjørner, M. Broy, and I.V. Pottosin, editors, *Proc. FMPA, Academgorodok, Novosibirsk,* RUS, LNCS 735, pages 128–141. Springer-Verlag, jun 28–jul 2, 1993.

[3] 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, FRA, 21 mar 1978.

[4] P. Cousot. Semantic foundations of program analysis. In S.S. Muchnick and N.D. Jones, editors, *Program Flow Analysis: Theory and Applications*, chapter 10, pages 303–342. Prentice-Hall, 1981.

[5] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *ENTCS*, 6, 1997. URL: `http://www.elsevier.nl/locate/entcs/volume6.html`, 25 pages.

[6] P. Cousot. Types as abstract interpretations, invited paper. In *24$^{th}$ POPL*, pages 316–331, Paris, FRA, jan 1997. ACM Press.

[7] P. Cousot and R. Cousot. Introduction to abstract interpretation. course notes for the ≪ NATO International Summer School 1998 on Calculational System Design ≫, Marktoberdorff, July 28 – August 9, 1998.

[8] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proc. 2$^{nd}$ Int. Symp. on Programming*, pages 106–130. Dunod, 1976.

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

[10] P. Cousot and R. Cousot. Automatic synthesis of optimal invariant assertions: mathematical foundations. In *ACM Symposium on Artificial Intelligence & Programming Languages*, Rochester, N.Y., SIGPLAN Notices 12(8):1–12, 1977.

[11] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *6$^{th}$ POPL*, pages 269–282, San Antonio, Texas, 1979. ACM Press.

[12] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *J. Logic Prog.*, 13(2–3):103–179, 1992. (The editor of JLP has mistakenly published the unreadable galley proof. For a correct version of this paper, see `http://www.dmi.ens.fr/~cousot`.).

[13] P. Cousot and R. Cousot. Abstract interpretation frameworks. *J. Logic and Comp.*, 2(4):511–547, aug 1992.

[14] P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe and M. Wirsing, editors, *Proc. Int. Work. PLILP '92, Leuven,* BEL*, 13–17 aug 1992*, LNCS 631, pages 269–295. Springer-Verlag, 1992.

[15] P. Cousot and R. Cousot. Galois connection based abstract interpretations for strictness analysis, invited paper. In D. Bjørner, M. Broy, and I.V. Pottosin, editors, *Proc. FMPA, Academgorodok, Novosibirsk,* RUS, LNCS 735, pages 98–127. Springer-Verlag, jun 28–jul 2, 1993.

[16] P. Cousot and R. Cousot. Higher-order abstract interpretation (and application to comportment analysis generalizing strictness, termination, projection and PER analysis of functional languages), invited paper. In *Proc. 1994 ICCL, Toulouse,* FRA, pages 95–112. IEEE Comp. Soc. Press, 16–19 may 1994.

[17] P. Cousot and R. Cousot. Formal language, grammar and set-constraint-based program analysis by abstract interpretation. In *Proc. 7^th^ FPCA*, pages 170–181, La Jolla, Calif., 25–28 jun 1995. ACM Press.

[18] P. Cousot and R. Cousot. Refining model checking by abstract interpretation. *Automated Software Engineering Journal, special issue on Automated Software Analysis*, 6(1), 1999. To appear.

[19] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *5^th^ POPL*, pages 84–97, Tucson, Ariz., 1978. ACM Press.

[20] S.K. Debray. Formal bases for dataflow analysis of logic programs. In G. Levi, editor, *Advances in Logic Programming Theory*, International Schools for Computer Scientists, section 3, pages 115–182. Clarendon Press, 1994.

[21] A. Deutsch. Semantic models and abstract interpretation techniques for inductive data structures and pointers, invited paper. In *Proc. PEPM '95, La Jolla, Calif.*, pages 226–229. ACM Press, 21–23 jun 1995.

[22] A. Deutsch, G. Gonthier, and M. Turin. La vérification des programmes d'ariane. *Pour la Science*, 243:21–22, jan 1998. (in French).

[23] R. Giacobazzi and F. Ranzato. Completeness in abstract interpretation: A domain perspective. In M. Johnson, editor, *Proc. 6^th^ Int. Conf. AMAST '97, Sydney,* AUS, LNCS 1349, pages 231–245. Springer-Verlag, 13–18 dec 1997.

[24] P. Granger. Improving the results of static analyses of programs by local decreasing iterations. Res . rep. LIX/RR/91/08, Laboratoire d'Informatique, École Polytechnique, Palaiseau, FRA, dec 1991.

[25] P. Granger. Static analysis of linear congruence equalities among variables of a program. In S. Abramsky and T.S.E. Maibaum, editors, *Proc. Int. J. Conf. TAPSOFT '91, Brighton,* GBR*, Volume 1 (CAAP '91),*, LNCS 493, pages 169–192. Springer-Verlag, 1991.

[26] P. Granger. Static analyses of congruence properties on rational numbers. In P. van Hentenryck, editor, *Proc. SAS '97, Paris,* FRA*, 8–10 sep 1997*, LNCS 1302, pages 278–292. Springer-Verlag, 1997.

[27] A. Venet. Abstract cofibred domains: Application to the alias analysis of untyped programs. In R. Cousot and D.A. Schmidt, editors, *Proc. SAS '96, Aachen,* GER*, 24–26 sep 1996,*, LNCS 1145, pages 368–382. Springer-Verlag, 1996.

[28] A. Venet. Abstract interpretation of the $\pi$-calculus. In M. Dam, editor, *Analysis and Verification of Multiple-Agent Languages, 5th LOMAPS Workshop, Stockhlom* SWE*, 24–26 jun 1996*, LNCS 1192, pages 51–75. Springer-Verlag, 1996.