Abstract Interpretation – 40 years back + some years ahead

Patrick Cousot
p.cousot@cs.nyu.edu cs.nyu.edu/~pcousot

Before starting (1972-73): formal syntax

- **Radhia Rezig**: works on precedence parsing (R.W. Floyd, N. Wirth and H. Weber, etc.) for Algol 68
  - Pre-processing (by **static analysis and transformation**) of the grammar before building the **bottom-up** parser
- **Patrick Cousot**: works on context-free grammar parsing (J. Earley and F. De Remer)
  - Pre-processing (by **static analysis and transformation**) of the grammar before building the **top-down** parser

Before starting (1972-73): formal semantics

- **Patrick Cousot**: works on the operational semantics of programming languages and the derivation of implementations from the formal definition
  - Static analysis of the formal definition and transformation to get the implementation by “pre-evaluation” (similar to the more recent “partial evaluation”)


Vision (1973)

Intervals ➞

Assertions ➞

Static analysis ➞

1973: Dijkstra’s handmade proofs

- Radhia Rezig: attends Marktoberdorf summer school, July 25–Aug. 4, 1973
  ➞ Dijkstra shows program proofs (inventing elegant backward invariants)

- Radhia has the idea of automatically inferring the invariants by a backward calculus to determine intervals

1974: origin

- Radhia Rezig shows her interval analysis ideas to Patrick Cousot
  ➞ Patrick very critical on going backwards from $[-\infty, +\infty]$ and claims that going forward would be much better
  ➞ Patrick also very skeptical on forward termination for loops

- Radhia comes back with the idea of extrapolating bounds to $\pm\infty$ for the forward analysis

- We discover widening = induction in the abstract and that the idea is very general

An important encounter

- I do my military service as a scientist with Jean Ichbiah
- Work on the revision of LIS (ancestor of Green → ADA)
- Will always be a very strong support on our work
Notes of Radhia Rezig on forward iteration from $\Box = \bot$ (1) versus backward iteration from $[-\infty, +\infty]$ (2)

First seminar in Grenoble: a warm welcome
- “Not all functions are increasing, for example, $\sin$”
- “This is woolly” ($fumeux$)
- “This will have applications in hundred years”

The IRIA-SESORI contract (1975–76)
- The project evaluator (Bernard Lohro) points us to the literature on constant propagation in data flow analysis (Kildall thesis).
- It appears that it is completely related to some of ours ideas, but a.o.
  - We are not syntactic (as in boolean DFA)
  - We have no need for some hypotheses (e.g. distributivity not even satisfied by constant propagation!)
  - We have no restriction to finite lattices (or ACC)
  - We have no need of an a-posteriori proof of correctness (e.g. with respect to the MOP as in DFA)
  - ...

The IRIA-SESORI contract (1975-76)
- New general ideas
  - The formal notions of abstraction/approximation
  - The formal notion of abstract induction (widening) to handle infiniteness and/or complexity
  - The systematic correct design with respect to a formal semantics
  - ...
The IRIA-SESORI contract (1975-76)

• The first contract report:

The first reports (1975)

The first abstract interpreter with widening (as of 23 Sep. 1975)

The first research report (Nov. 1975)

The first publication (1976)

• The first publication (ISOP II, Apr. 76)

Maturation (1976 – 77): from an algorithmic to an algebraic point of view

• Narrowing, duality
• Transition systems, traces
• Fixpoints, chaotic/asynchronous iterations, approximation
• Abstraction, formalized by Galois connections, closure operators, Moore families, ...
• Numeric and symbolic abstract domains, combinations of abstract domains
• Recursive procedures, relational analyses, heap analysis
• etc.
A Visitor

- Hi, I am Steve Warshall
- The theorem?
- Yes
- Steve Schuman told me you are doing interesting work
- ...
- You should publish in Principles of Programming Languages.

And a bit of mathematics...

On submitting to POPL

- For POPL’77, we submit (on Aug. 12, 1976) copies of a two-hands written manuscript of 100 pages. The paper is accepted!
On abstracting: transition system

Reachability semantics is an abstraction of the relational semantics
(in PC’s thesis, 21 march 1978 also § 3 of POPL’79)

On convincing ...

- During PC’s thesis defense, it was suggested that abstraction/approximation is useless since computers are finite and executions are timed-out (so, the second part of the thesis on fixpoint approximation/widening/narrowing/... is superfluous!)
- Fortunately we do not listen (otherwise we would have invented enumeration methods that fail to scale)
- On the contrary, in 1978, during a seminar at Harvard (1), G. Birkhoff appears interested, according to his questions & feedback, in the effective computational aspects of lattice fixpoint theory

The principles (1977–79) are lasting

- Define the semantics (operational, denotational, axiomatic, ...) of the programming language (as a ... / trace semantics / transition system / transformers / ...)
- Define the strongest property of interest (also called the collecting semantics)
- Express this collecting semantics in fixpoint (constraint, rule-based,...) form
- Define the abstraction/concretization compositionally (by composition of elementary abstractions and abstraction constructors/functors)
- Design the abstract proof / analysis semantics by calculus using [structural] abstraction i.e. abstract domain + abstract fixpoint
- Combine abstractions (e.g. reduced product)

Abstract interpretation: Research takes time
• Type checking and inference is an abstract interpretation:

Typing

POPL 1997:

Typing

ESOP 2012:

Probabilistic static analysis

Probabilistic static analysis

Probabilistic Abstract Interpretation

Patrick Cousot and Michael Monerau
Courant Institute, NYU and École Normale Supérieure, France

Abstract. Abstract interpretation has been widely used for verifying properties of computer systems. Here, we present a way to extend this framework to the case of probabilistic systems. The probabilistic abstraction framework that we propose allows us to systematically lift any classical analysis or verification method to the probabilistic setting by separating in the program semantics the probabilistic behavior from the (non-)deterministic behavior. This separation provides new insights for designing novel probabilistic static analyses and verification methods.

We define the concrete probabilistic semantics and propose different ways to abstract them. We provide examples illustrating the expressiveness and effectiveness of our approach.


Termination

- POPL 2012:

An Abstract Interpretation Framework for Termination

Patrick Cousot
CNRS, École Normale Supérieure, and INRIA, France
Cousot Institute, NYU, USA
cousot@ens.fr, picousot@nyu.edu

Abstract

Proof, verification and analysis methods for termination all rely on two induction principles: (1) a variant function or induction on data ensuring progress towards the end and (2) some form of induction on the program structure.

The abstract interpretation design principle is first illustrated for the design of new forward and backward proof, verification and analysis methods for safety. The safety-collecting semantics defining the strongest safety property of programs is first expressed in a constructive fixpoint form. Safety proof and checking/verification methods then immediately follow by fixpoint induction. Static analysis of abstract safety properties such as invariance are constructively designed by fixpoint abstraction (or approximation) to automatically infer safety properties. So far, no such clear design principle did exist for termination so that the existing approaches are scattered and largely not compatible with each other.

Denotational Semantics

- POPL 1992:

Inductive Definitions, Semantics and Abstract Interpretation

Patrick Cousot
LIX, École Polytechnique
91200 Palaiseau (France)
cousot@lri.fr

Abstract

We introduce and illustrate a specification method combining role-based inductive definitions, well-founded induction principles, transfinite theory and abstract interpretation for general use in computer science. Fins as well as infinite objects can be specified, at various levels of detail related by abstractions. General proof principles are applicable to prove properties of the specified objects.

The specification method is illustrated by introducing C/SOS, a structured operational semantics generalization of Plotkin’s [28] structured operational semantics (SOS) so as to describe the finite, as well as the infinite behavior of programs in a uniform way and by constructively deriving inductive presentations of the other (reductions, denotational, predicate transformers, . . . ) semantics from C/SOS by abstract interpretation.

Termination

Hierarchies of semantics

- POPL 2012:

Hierarchies of semantics

Radhia Cousot
CNRS, École Normale Supérieure, and INRIA, France
rcousot@ens.fr

Abstract

For (1), we show that this design principle applies equally well to potential and definite computations. The trace-based termination collecting semantics is given a fixpoint definition. In abstraction yields a fixpoint definition of the best variant function. By further abstraction of this best variant function, we derive the Floyd/Johnson termination proof method as well as new static analysis methods to effectively compute approximations of this best variant function.

For (2), we introduce a generalization of the syntactic notion of structural induction (as found in Hoare logic) into a semantic structural induction based on the new semantic concept of so-called trace cover covering execution traces by sequences, a new basis for formulating program properties. Its abstractions allow for generalized secure proof, verification and static analysis by induction on both program structure, control and data. Examples of particular instances include Floyd’s handling of loop-carry points as well as nested loops, Barendregt’s incoherence and termination correctness proof method, and Podlasek-Rybalchenko transition invariants.
Hierarchy of semantics

• TCS 2002:

Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation

Patrick Cousot

Département d’Informatique, École Normale Supérieure, 45 rue d’Ulm, 75230 Paris cedex 05, France, Patrick.Cousot@ens.fr, http://www.di.ens.fr/~cousot

We construct a hierarchy of semantics by successive abstract interpretations. Starting from the maximal trace semantics of a transition system, we derive the big-step semantics, termination and nontermination semantics, Plotkin’s natural, Smyth’s demonic and Hoare’s angelic relational semantics and equivalent nondeterministic denotational semantics (with alternative powerdomains to the Egli-Milner and Smyth constructions), D. Scott’s deterministic denotational semantics, the generalized and Dijkstra’s conservative/liberal predicate transformer semantics, the generalized/total and Hoare’s partial correctness axiomatic semantics and the corresponding proof methods. All the semantics are presented in a uniform fixpoint form and the correspondences between these semantics are established through composable Galois connections, each semantics being formally calculated by abstract interpretation of a more concrete one using Kleene and/or Tarski fixpoint approximation transfer theorems.

Parallelism

Parallelism

• POPL 2017:

Ogre and Pythia: An Invariance Proof Method for Weak Consistency Models

Jade Alglave

University College London
Microsoft Research Cambridge, UK
jaalglave@microsoft.com, j.alglave@ucl.ac.uk

Patrick Cousot

New York University, USA
der. École Normale Supérieure, PSL, France
pcousot@cims.nyu.edu, cousot@ens.fr

Bi-inductive Structural Semantics*

Patrick Cousot

Département d’informatique, École normale supérieure, 45 rue d’Ulm, 75230 Paris cedex 05, France

Radhia Cousot

CNRS & École polytechnique, 91128 Palaiseau cedex, France

Abstract

We propose a simple order-theoretic generalisation, possibly non monotone, of set-theoretic inductive definitions. This generalisation covers inductive, co-inductive and bi-inductive definitions and is preserved by abstraction. This allows structural operational semantics to describe simultaneously the finite/terminating and infinite/diverging behaviors of programs. It is illustrated on grammars and the structural bifinitary small/big-step trace/relational/operational semantics of the call-by-value λ-calculus (for which co-induction is shown to be inadequate).

Key words: fixpoint definition, inductive definition, co-inductive definition, bi-inductive definition, non-monotone definition, grammar, structural operational semantics, SOS, trace semantics, relational semantics, small-step semantics, big-step semantics, divergence semantics.
Abstract interpretation: Industrialization

- Very first industrial implementation:
  - The interval analysis was implemented in the AdaWorld compiler for IBM PC 80286 by J.D. Ichbiah and his Alsys SA corporation team in 1980–87.

Warm welcome

- Real-time software development companies: we have to pay for this new option of the ADA compiler, but:
  - The machine code size is significantly reduced → we cannot sell as much memory as we did before;
  - Many bugs are found at compile time → we make less money with our debugging services.

AbsInt Angewandte Informatik GmbH

- Astrée sold by AbsInt:
Abstract interpretation based static analyzers

- **Polyspace static analysis** [www.mathworks.com/products/polyspace.html](http://www.mathworks.com/products/polyspace.html)
- **Julia** (Java) [www.juliasoft.com](http://www.juliasoft.com)
- **Ikos**, NASA [ti.arc.nasa.gov/opensource/ikos/](http://ti.arc.nasa.gov/opensource/ikos/)
- **Clousot** for code contract, Microsoft, [github.com/Microsoft/CodeContracts](http://github.com/Microsoft/CodeContracts)
- **Infer** (Facebook) [http://fbinfer.com](http://fbinfer.com)
- **Zoncolan** (Facebook)
- **Google**
- ...

The future is hard to predict

- From my thesis in 1978:
  - From “30 years of Abstract Interpretation”:
  - **Programming**
    - The evolution of programming languages and programming assistance systems has greatly helped to considerably speed up the development and scale up the size of conceivable programs
    - Software quality remains much far beyond, essentially because it is anti-economical
    - Until the next catastrophe, evolution of the standards, revolution of the customers, or new laws holding computer scientists accountable for bugs
  - **Formal methods**
    - Formal methods might then become profitable at every stage of program design
    - The answers, if any, will definitely have to scale up, at a reasonable cost
    - Up to now, research has mainly concentrated on easy avenues with short-term rewards
    - Small groups cannot make it, large groups fail to share common interests
    - There is still a long long way to go

Abstract interpretation

- Beyond programming, abstraction is the only way to apprehend complex systems
- Therefore, the scope of application of abstract interpretation always is large
- Over 30 years, abstract interpretation theory, practice and achievements have grown despite trends and evanescent applications
- Hopefully, abstract interpretation will continue to be useful in the future

The future is hard to predict

- From my thesis in 1978:
  - Computer, economical and biological systems
  - Sequential and parallel programs

THE END

Many thanks to all of you who contributed to abstract interpretation!
The future is hard to predict

- From the Dagstuhl Seminar “Formal Methods — Just a Euro-Science?” in December 2010:
  - More properties:
    - Security (not dynamically checkable)
    - ...
  - More systems and tools:
    - Parallel and distributed systems,
    - Cyber-physical (continuous+discrete)
    - Biological, financial, ...
  - Better practices:
    - Verification from design to implementation

Hopes (10 years)

- Complex data structures (libraries like for numerical domains)
- Program security
- Parallel & distributed systems, weak consistency models

Dreams (40 years)

1. The semantics is specified structurally and compositionally
2. The abstraction is specified by composition of Galois connections
   POPL 2014:
   **A Galois Connection Calculus for Abstract Interpretation**
   Patrick Cousot
   CNRS-LR, NYU, USA - pcousot@lri.fr
   Radhia Cousot
   CNRS-Emeritio, ENS, France - r cousot@ens.fr
3. The calculational design of the abstract interpreter is supported by libraries and tools
4. All modular and compositional

Dreams (40 years)

4. The design of static analyzers is computer-assisted by automatic composition of certified public-domain modules for:
   - Abstract domains
   - Syntax and semantics to fixpoint equations
   - Parallel/distributed fixpoint solvers (direct or with convergence acceleration)
   - User-interface automatic design
   - Automatic fixing of errors
The End