Formalizations of Abstraction in the Abstract Interpretation Theory

Patrick Cousot

École Normale Supérieure
DI, 45 rue d'Ulm, 75230 Paris Cedex 05
France

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

Dagstuhl seminar 06281 « The challenge of Software Verification » 09-13/07/2006

Property Semantics

- Σ : computations (formalize program execution)
- $\wp(\Sigma)$: properties (the computations that have the property)
- F : property transformer (usually effect of a command on computations)
- S : property semantics

$$S^0 = \bot$$
$$S^{\delta+1} = F(S^\delta)$$
$$S^\lambda = \bigsqcup_{\beta<\lambda} S^\beta$$

assumed ultimately stationary, with limit $S = S^\varepsilon = S^{\varepsilon+1}$

- $\sqsubseteq$: implication, $\sqcup$ lub

The Classical Abstraction formalized by Galois Connections.

$$\langle \wp(\Sigma), \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L, \leq \rangle$$

concrete properties — abstraction — abstract properties

$$\alpha(P) \leq Q \iff P \sqsubseteq \gamma(Q)$$

($\Rightarrow$) Approximation from above (sound since concrete implies abstract)

($\Leftarrow$) Always exists a best approximation of concrete properties P : $\alpha(P)$

Many equivalent formalizations: closure operators, Moore families, etc... see CC [POPL 77].

Example 1 of abstraction: Schneider's notion of program properties

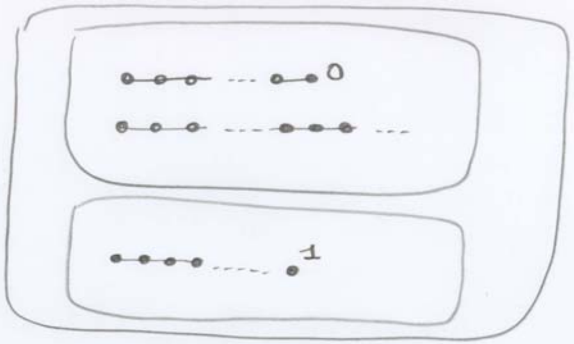
- S : states
- S^∞ : traces (finite or infinite sequence of states)
- $\wp(S^\infty)$: semantics (set of traces)
- $\wp(\wp(S^\infty))$: properties (set of semantics)

$$\langle \wp(\wp(S^\infty)), \subseteq \rangle \underset{\alpha_\cup}{\overset{\gamma_\cup}{\leftrightarrows}} \langle \wp(S^\infty), \subseteq \rangle$$

$$\alpha_\cup(P) \triangleq \bigcup P$$

- All properties in $\wp(S^\infty)$ are safety $\cap$ liveness (Schneider)
- Some properties in $\wp(\wp(S^\infty))$ are not in $\wp(S^\infty)$ whence neither safety nor liveness

Counter-example.



Examples

[print 0]
[print 0 [] while true do skip]
[print 1]

Counter-examples

[print 0 [] print 1]

Example 2 : the safety abstraction

- Prefix closure of a set of traces :

$$\alpha_P(T) = \{\sigma \in S^+ \mid \exists \sigma' : \sigma\sigma' \in T\}$$

- Limit closure of a set of traces :

$$\alpha_L(T) := T \cup \{\sigma \in S^\omega \mid \forall i : \exists j \geq i : \sigma_0 \ldots \sigma_j \in T\}$$

- Safety abstraction :

$$\langle \wp(\wp(S^\infty)), \subseteq \rangle \underset{\alpha_L \circ \alpha_P \circ \alpha_U}{\overset{\gamma_U \circ \gamma_P \circ \gamma_L}{\leftrightarrows}} \langle \wp(S^\infty), \subseteq \rangle$$

- There is a best safety abstraction of any property

Advantage of the Galois connection based formalization of the abstraction

- There is a best (ie. most precise) way to approximate any concrete operation in the abstract
- Example :

$$F : \wp(\Sigma) \xrightarrow{m} \wp(\Sigma)$$

$$\bar{F} = \alpha \circ F \circ \gamma \qquad \text{the best}$$

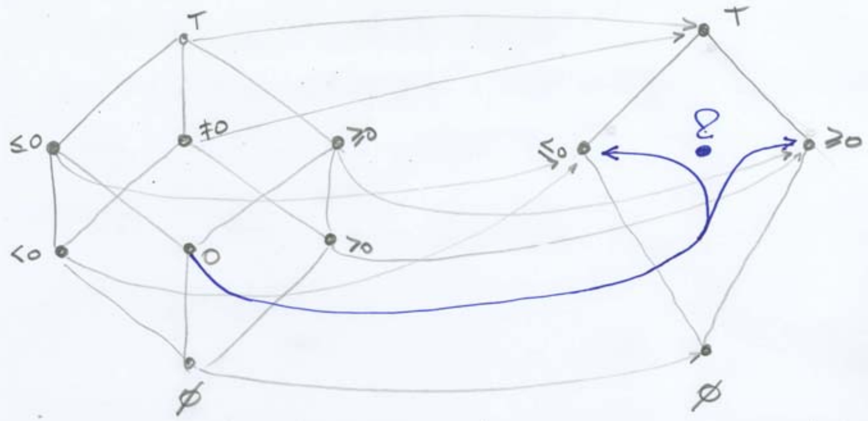
can be weakened into :

$$\alpha \circ F \leq \bar{F} \circ \alpha$$

or

$$F \circ \gamma \sqsubseteq \gamma \circ \bar{F}$$

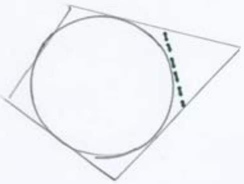
In absence of best abstraction



There are different minimal (or no minimal) abstract properties over-approximating a given concrete property.

Many examples of absence of best approximation ⟹ No Galois Connection

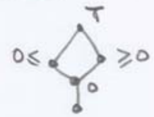
- Convex polyhedra CH [POPL'78]



- Regular expressions or (contex free) grammars approximating a language on a finite alphabet CC [FPCA'95]

Enriching the abstract domain

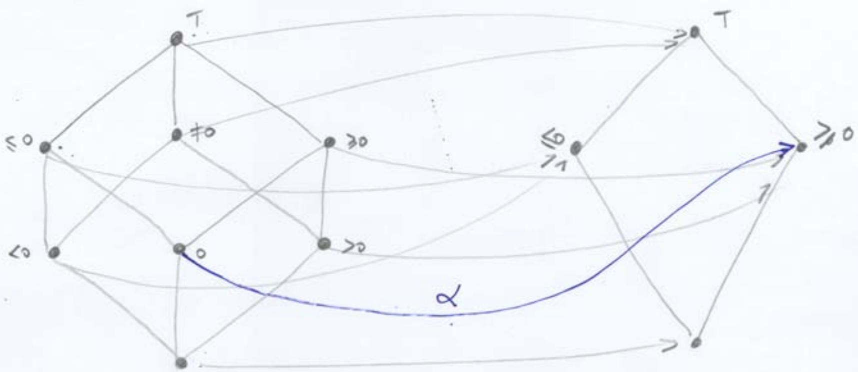
- It is always possible to refine the abstract domain (by adding missing best approximations) to get a Galois connection
- Example :



- Too complex in general (must add infinitely many abstract properties, usually too complex)

  Example : polyhedra ⟶ convex sets.

Abstraction-based approximation



- Make an arbitrary choice among the (minimal?) upper approximation by defining the abstraction α

Inconvenience of an abstraction-based approximation

- The choice of the "useful" abstraction is made once for all
- Cannot be adapted to the context of use

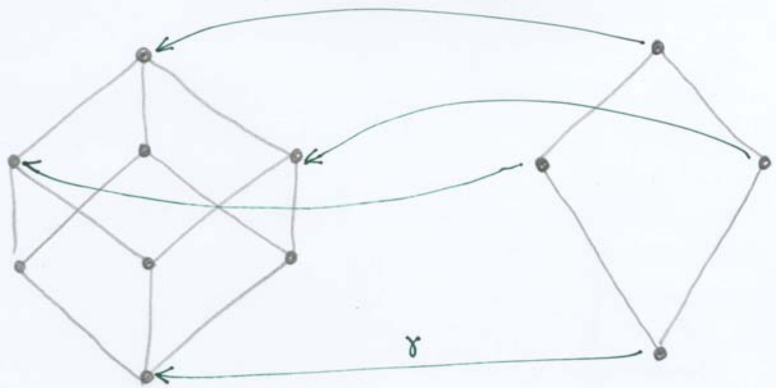
Example

(≥0) + 0 —α→ (≥0)

(≤0) + 0 —α→ (≤0)

} incompatible locally best choices

Concretization-based approximation



- Define the meaning of abstract properties
- Postpone the decision on how to abstract concrete properties

Advantage of a concretization-based abstraction

- The choice of the abstraction $\overline{P}$ of a concrete property P can be made in context
- Nevertheless the soundness condition remains always the same

$$P \subseteq \gamma(\overline{P})$$

- Example:

$$(\geq 0) + 0 \overset{\gamma}{\subseteq} \mapsto (\geq 0) + (\geq 0)$$

$$(\leq 0) + 0 \overset{\gamma}{\subseteq} \mapsto (\leq 0) + (\leq 0)$$

- Note: soundness is non trivial (e.g. Sintzoff rule of signs is erroneous)

Abstract semantics

- $\overline{S}^0 = \overline{I}$
  $\overline{S}^{\delta+1} = \overline{F}(\overline{S}^\delta)$
  $\overline{S}^\lambda = \overline{\bigsqcup}_{\beta<\delta} \overline{S}^\beta$
  assumed to be ultimately stationary at rank $\overline{\epsilon}$
- Local soundness conditions:
  $\bot \subseteq \gamma(\overline{I})$
  $F \circ \gamma \mathrel{\dot{\subseteq}} \gamma \circ \overline{F}$
  $\bigsqcup_i \gamma(x_i) \subseteq \gamma(\overline{\bigsqcup}_i x_i)$
- Soundness theorem:

$$S = S^\epsilon \subseteq \gamma(\overline{S}) = \gamma(\overline{S}^{\overline{\epsilon}})$$

Ensuring convergence

(1) The abstract iterates are (usually) increasing
→ the lattice satisfies the ascending chain condition

Example: finite lattice in abstract model checking

(2) widening
- $\gamma(x) \subseteq \gamma(x \nabla y)$, $\gamma(y) \subseteq \gamma(x \nabla y)$
- $\overline{S}_0 = \overline{I}$, $\overline{S}_{n+1} = \overline{S}_n \nabla \overline{F}(\overline{S}_n)$ if $\overline{S}_n \subseteq \overline{F}(\overline{S}_n)$, $\overline{S}_{n+1} = \overline{S}_n$ if $\overline{F}(\overline{S}_n) \subseteq S_n$ is ultimately stationary at $\overline{S}$

$\Rightarrow$ $S \subseteq \gamma(\overline{S})$ — soundness

Why is widening better than finitary choices of the abstract domain

- Termination in both cases
- The widening can always be chosen to be more precise.

Proof: (1)

```
x = 0
while x ≤ n do
   x := x+1
od
```

→ $x \in [0, n]$ by natural analysis with widening

$n \in \mathbb{N}$ is any given constant

(2) no abstract domain satisfying the ascending chain condition can contain all desired answers $\bigcup_{n \in \mathbb{N}} [0, n]$

(3) any finitary analysis will be strictly less precise on infinitely many programs.

Reduced Product

- Concrete domain : $\langle L, \sqsubseteq, \bot, \sqcup, \sqcap, F \rangle$
- Abstract domains : $\langle \overline{L}_i, \overline{\sqsubseteq}_i, \overline{\bot}_i, \overline{\sqcup}_i, \overline{\sqcap}_i, \overline{F}_i \rangle$, $i \in [1, n]$
- Reductions :

  $\rho_{ij}(\overline{P}_i, \overline{P}_j) \sqsupseteq \gamma_i(\overline{P}_i) \sqcap \gamma_j(\overline{P}_j)$

  $\rho(\overline{P}_1, \ldots, \overline{P}_n) =$ iterate $\rho_{i,j}(\overline{P}_i, \overline{P}_j)$ $i, j \in [1, n], i \neq j$ until stabilization (or stopped by narrowing CC[POPL 77])
- Apply ρ during iteration (if not everywhere)

⚠ : A widening converging on each $\overline{L}_i$ may not converge on $\prod_{i=1}^{n} \overline{L}_i$.

Application : ASTRÉE

- see www.astree.ens.fr
- **Which Program Run-Time Properties are Proved by ASTRÉE?**

  ASTRÉE aims at proving that the C programming language is correctly used and that there can be no *Run-Time Errors* (RTE) during any execution in any environment. This covers:

  - Any use of C defined by the international norm governing the C programming language (ISO/IEC 9899:1999) as having an undefined behavior (such as division by zero or out of bounds array indexing),
  - Any use of C violating the implementation-specific behavior of the aspects defined by ISO/IEC 9899:1999 as being specific to an implementation of the program on a given machine (such as the size of integers and arithmetic overflow),
  - Any potentially harmful or incorrect use of C violating optional user-defined programming guidelines (such as no modular arithmetic for integers, even though this might be the hardware choice), and also
  - Any violation of optional, user-provided assertions (similar to `assert` diagnostics for example), to prove user-defined run-time properties.
- demonstration of ASTRÉE ...

References

- Abstract interpretation frameworks :
  - Patrick Cousot & Radhia Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511—547, August 1992.
- Widenings :
  - Patrick Cousot & Radhia Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 238—252, Los Angeles, California, 1977. ACM
  - Patrick Cousot & Radhia Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation, invited paper. In M. Bruynooghe & M. Wirsing, editors, *Programming Language Implementation and Logic Programming, Proceedings of the Fourth International Symposium, PLILP'92*, Leuven, Belgium, 13—17 August 1992, Lecture Notes in Computer Science 631, pages 269—295. © Springer, Berlin, Germany, 1992.
- Reduced product :
  - Patrick Cousot & Radhia Cousot. Systematic design of program analysis frameworks. In *Conference Record of the Sixth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* pages 269—282, San Antonio, Texas, 1979. ACM Press, New York, U.S.A.
- Polyhedral analysis (γ, ∇ based)
  - Patrick Cousot & Nicolas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference R ecord of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Program ming Languages*, pages 84—97, Tucson, Arizona, 1978. ACM Press, New York, NY, USA.

- Grammar-based analysis (σ, ∇-based)

  - Patrick Cousot & Radhia Cousot. Formal Language, Grammar and Set-Constraint-Based Program Analysis by Abstract Interpretation. In *Conference Record of FPCA '95 SIGPLAN/SIGARCH/WG2.8 Conference on Functional Programming and Computer Architecture*, pages 170—181, La Jolla, California, U.S.A., 25-28 June 1995. ACM Press, New York, U.S.A.

- ASTRÉE
  - www.astree.ens.fr

  - Bruno Blanchet, Patrick Cousot, Radhia Cousot, Jérôme Feret, Laurent Mauborgne, Antoine Miné, David Monniaux, & Xavier Rival.
    A Static Analyzer for Large Safety-Critical Software.
    In *PLDI 2003 — ACM SIGPLAN SIGSOFT Conference on Programming Language Design and Implementation* , 2003 Federated Computing Research Conference, June 7—14, 2003, San Diego, California, USA, pp. 196—207, © ACM.