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

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

\[ \begin{align*}
S^0 &= \top \\
S^{n+1} &= F(S^n) \\
S^1 &= \bigcup S^0
\end{align*} \]

assumed ultimately stationary, with
limit \( S = \lim S^\infty = S^{n+1} \)

- \( E \) : implication, \( \leq \) lub

---

The Classical Abstraction formalized
by Galois Connections.

\[ \langle \Phi(\Sigma), \leq \rangle \xleftrightarrow{\sigma} \langle h, \leq \rangle \]

concrete properties abstract abstract properties

\[ \alpha(p) \leq q \iff p \leq \alpha(q) \]

\((\Rightarrow)\) 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
- \( \Phi(S^0) \) : traces (finite or infinite sequence of states)
- \( \Phi(\Phi(S^0)) \) : semantics (set of traces)
- \( \Phi(\Phi(S^0)) \) : property (set of semantics)

\[ \langle \Phi(\Phi(S^0)), \leq \rangle \xleftrightarrow{\sigma} \langle \Phi(S^0), \leq \rangle \]

\[ \sigma(p) \leq \subseteq U \]

- All properties in \( \Phi(S^0) \) are safety or liveness (Schneider)
- Some properties in \( \Phi(\Phi(S^0)) \) are not in \( \Phi(S^0) \)
whence neither safety nor liveness
Counter-example:

\[
\begin{array}{c}
\cdots \cdots \cdots \\
\cdots \cdots \cdots \\
\end{array}
\]

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:
  \[ \alpha^L(T) = T \cup \{ \sigma \in S^w \mid \forall i : \exists j \geq i : \sigma_0 \cdots \sigma_j \in T \} \]
- Safety abstraction:
  \[ \langle \mathcal{G}(S^w), \subseteq \rangle \leftrightarrow \exists \mathcal{G}(S^w), \subseteq \]
- There is a best safety abstraction of any property

Advantage of the Galois connection based formalization of the abstraction

- There is a best (i.e. most precise) way to approximate any concrete operation in the abstract

- Example:
  \[ F : \mathcal{G}(E) \xrightarrow{m} \mathcal{G}(E) \]
  \[ F = \alpha \circ F \circ \delta \]
  the best

  can be weakened into:
  \[ \alpha \circ F \leq F \circ \delta \]
  or
  \[ F \circ \delta \leq \delta \circ F \]

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 $\Rightarrow$ No Galois Connection

- Convex polyhedra $\text{CH}[\text{PoPL'78}]

- Regular expressions or (context free) grammars approximating a language on a finite alphabet $\text{CC}[\text{PPCA'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: $\text{OS} \xrightarrow{T} \text{TO}$

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

- Example: polyhedra $\rightarrow$ convex sets.

---

Abstraction-based approximation

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

---

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

\[
\begin{align*}
\geq 0 & \xrightarrow{0} \geq 0 \\
\leq 0 & \xrightarrow{0} \leq 0
\end{align*}
\]

$\{\text{incompatible locally best choices}\}$
Conceptualization-based approximation

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

Advantage of a conceptualization-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:

\[
\begin{align*}
\overline{O} + 0 & \rightarrow 20 + \overline{O} \\
\overline{E} + 0 & \rightarrow 30 + \overline{E}
\end{align*}
\]

- Note: soundness is non-trivial (e.g., S/\overline{E} rule of signs is erroneous).

Abstract semantics

\[
\begin{align*}
\delta^0 &= \delta \\
\delta_{i+1} &= \delta^i \\
\delta_i = \delta & \text{ assumed to be ultimately stationary at rank } \delta
\end{align*}
\]

- Local soundness conditions:
  \[ P \subseteq \gamma(\overline{P}) \]
  \[ F \circ \gamma \subseteq \gamma \circ F \]
  \[ \bigvee \gamma(x) \subseteq \gamma(\bigvee x) \]
- Soundness theorem:
  \[ S = S^\delta \subseteq \gamma(S) = \gamma(S^\delta) \]

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
   - \( \gamma(x) \subseteq \gamma(x \lor y) \rightarrow \gamma(y) \subseteq \gamma(x \lor y) \)
   - \( \delta_0 = \delta, \delta_{i+1} = \delta_i \lor F(\delta_i) \text{ if } S_i \subseteq F(\delta_i) \)
   - \( \delta_n = \delta_n \lor F(\delta_n) \subseteq S_n \text{ is ultimately stationary at } \delta \)

\[ S \subseteq \gamma(S) \text{ - soundness} \]
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 \( x \leq n \) do \( x \leftarrow x + 1, n \) by mutual analysis with widening
  
  (a) \( n + 1 \) is any given constant
  
  (b) no abstract domain satisfying the ascending chain condition can contain all desired answers \( \nu \in [0, n] \)
  
  (3) any finitary analysis will be strictly less precise in infinitely many programs.

---

**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 are no run-time errors (RTEs) during any execution in any environment. This covers:

- Any use of C-language generic and standard libraries (ISO/IEC 9899:1990), such as division by zero or out of bounds array indexing.
- Any use of C-language specific behavior of the objects defined by the ISO/IEC 9899:1990 standard, such as the use of integers and arithmetic overflows.
- Any potentially harmful or incorrect use of C-language specific 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 diagnoses for example), to prove user-defined run-time properties.

**Demonstration of ASTRÉE...**

---

**Reduced Product**

- Concrete domain: \( \langle E, F, +, \cdot, 0, 1 \rangle \)
- Abstract domain: \( \langle \vec{E}, \vec{F}, +, \cdot, 0, 1 \rangle \)

- Reductions:
  \[ p_i (\vec{E}, \vec{F}) \equiv \chi_i (\vec{E}) \cap \chi_i (\vec{F}) \]
  \[ p (\vec{E}, \ldots, \vec{F}) \equiv \mbox{iterate } p_i (\vec{E}, \vec{F}) \mbox{ until stabilization (or stopped by narrowing CCFPSL 99)} \]

- Apply \( p \) during iteration (if not everywhere)

\[ \Delta : \mbox{a widening converging on each } \vec{E} \mbox{ may not converge on } \vec{F}_{\vec{E}} \]

---

**References**

- Abstract interpretation framework:
  - Us. terms:
  - Reduced product:
    - Polyhedral analysis (\( \Delta \), \( \Sigma \) based):
Grammar-based analysis (\(\delta, \nu\)-based)


- ASTREE
  - www.astree.ens.fr