Abstraction in Abstract Interpretation

Patrick COUSOT
École Normale Supérieure
45 rue d'Ulm
75230 Paris cedex 05, France

mailto:Patrick.Cousot@ens.fr
http://www.di.ens.fr/~cousot

Workshop on Refinement and Abstraction
ETL Osaka, Japan, November 15-17, 1999

Abstract

Abstract interpretation is a semantic approximation theory which has mainly been used for the design of static program analyzers. Our objective is to explain and illustrate the notion of abstraction/concretization and its numerous variants which are commonly used in abstract interpretation to formalize the loss of information. We also explain how the concrete model can be transformed into an abstract semantic model, and inversely for refinement.

Several examples are given for the design of programming language semantics as well as model-checking and program analysis algorithms. To illustrate the notions of relative completeness and of existence of a best abstraction, we show that transitional, demonic, natural and angelic denotational, predicate transformer and axiomatic semantics are all relatively complete, best abstractions of a maximal trace semantics (or equivalently that the maximal trace semantics is a refinement of all these semantics). To illustrate incompleteness, we consider model-checking of finite transition systems for a temporal logic, both with maximal trace semantics. The logic can be restricted to ensure relative completeness at the expense of expressiveness. To illustrate inexistence of best approximations, we consider several abstract domains for the abstraction of sets of vectors of numbers and sets of graphs (for so-called set-based analysis).

1. Introductive example

Abstract interpretation

- Abstract interpretation is a semantic approximation theory [2];
- Mainly used for the design of semantics [3] and static program analyzers [1].

References

[1] 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.

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

[3] 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.

Abstraction in abstract interpretation

- Abstraction is understood as an approximation:

> A program analyzer is an $\overbrace{\text{approximate implementation}}^{\text{abstract}}$ of the $\underbrace{\text{program (collecting) semantics.}}_{\text{interpretation}}$

Objects and their properties

- Programming is relative to objects:

$$\text{Refined object} \quad \underset{\text{abstraction}}{\overset{\text{refinement}}{\leftrightarrows}} \quad \text{Abstract object}$$

- Program proof/analysis to object properties:

$$\begin{array}{c}\text{Concrete}\\ \text{object property}\end{array} \quad \underset{\text{abstraction}}{\overset{\text{concretization}}{\leftrightarrows}} \quad \begin{array}{c}\text{Abstract}\\ \text{object property}\end{array}$$

Properties as sets

- A property is the set of objects which have this property;
- Example (properties of integers):
  - Positive: $\{1, 2, 3, 4, \ldots\}$
  - Odd: $\{1, 3, 5, 7, \ldots\}$
- There is often a confusion on the fact that abstract interpretation does not deal with abstract objects but with abstract properties of objects;
- This is because the two notions sometime coincide;
- The view of abstract interpretation as abstraction of properties is more powerful that pseudo-evaluation on abstract objects .

Example: rule of signs standard semantics

- Standard semantics:
  - Operational: what are the steps of evaluation of the expression when knowing an assignment of values to the free variables;
  - Example ($\rho = [x : 5, y : -3]$):

$$\begin{array}{l} \quad [\![x \times x + y \times y]\!]\rho \\ \longrightarrow ([\![x]\!]\rho \times [\![x]\!]\rho) + ([\![y]\!]\rho \times [\![y]\!]\rho) \\ \longrightarrow (5 \times 5) + (-3 \times -3)) \\ \longrightarrow 25 + 9 \\ \longrightarrow 34 \end{array}$$

- Denotational: what is the value of the expression when knowing an assignment of values to the free variables:

$$[\![e]\!] \in (\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z}$$

$$\begin{aligned} [\![n]\!]\rho &= n \\ [\![\mathtt{x}]\!]\rho &= \rho(\mathtt{x}) \\ [\![e_1 \times e_2]\!]\rho &= [\![e_1]\!].[\![e_1]\!] \\ [\![e_1 + e_2]\!]\rho &= [\![e_1]\!] + [\![e_1]\!] \end{aligned}$$

Example: rule of signs
1 — the abstract object point of view

- Objective: determine the sign of an expression;
- Pseudo-evaluation method:
  - replace values by their signs;
  - interpret arithmetic operators on signs:

    $+1 + +1 = +1$,

    $+1 \times -1 = -1$, etc.
- Abstraction:
  - $\overbrace{\text{integer}}^{\text{concrete object}} \longmapsto \overbrace{\text{sign}}^{\text{abstract object}}$
  - $\overbrace{\text{integer} \times \text{integer} \mapsto \text{integer}}^{\text{concrete operation}} \longmapsto \overbrace{\text{sign} \times \text{sign} \mapsto \text{sign}}^{\text{abstract operation}}$

.../...

- Example ($\rho = [x : +1, y : -1]$):

$$\begin{aligned} & [\![x \times x + y \times y]\!]\rho \\ \rightarrow\ & ([\![x]\!]\rho \times [\![x]\!]\rho) + ([\![y]\!]\rho \times [\![y]\!]\rho) \\ \rightarrow\ & (+1 \times +1) + (-1 \times -1)) \\ \rightarrow\ & +1 + +1 \\ \rightarrow\ & +1 \end{aligned}$$

- Correctness: the rule of signs is a step by step simulation of the standard semantics (inconclusive when no rule applies e.g. $+1 + -1 = ?$);
- Same idea in "subject reduction" of type theory.

Example: rule of signs
2 — the abstract property point of view

- Property of an expression: set of its possible semantics;
- Collecting semantics: the strongest program property:

$$\{\!|e|\!\} \in \wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z})$$
$$\{\!|e|\!\} \triangleq \{[\![e]\!]\} \quad (1)$$

- Abstract semantics: a computable approximation of the collecting semantics.

Approximation

- Two alternatives:
  - Universal/from above: consider a superset of the possible cases,
  - Existential/from below: consider a subset of the possible cases;
- By duality, only universal approximation need to be formely studied;
- The rule of signs is a universal approximation (i.e. $+1 + +1 = +1$ is valid whether $3 + 2 = 5$ or $3 + 2 = 1789!$) since more cases are considered than possible.

Approximation for the rule of signs

- $\alpha_0 : \mathbb{Z} \mapsto \mathbb{S}$ where $\mathbb{S} \triangleq \{+1, 0, -1\}$ (2)

$$\alpha_0(n) = -1 \text{ iff } n < 0$$
$$\alpha_0(n) = 0 \quad \text{iff } n = 0$$
$$\alpha_0(n) = +1 \text{ iff } n > 0$$

- $\alpha_1 \in \wp(\mathbb{Z}) \mapsto \wp(\mathbb{S})$ (3)

$$\alpha_1(N) \triangleq \{\alpha_0(n) \mid n \in N\}$$

$\gamma_1 \in \wp(\mathbb{S}) \mapsto \wp(\mathbb{Z})$

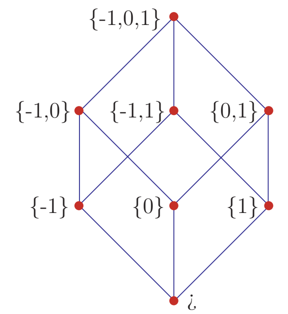
$$\gamma_1(S) \triangleq \{n \mid \alpha_0(n) \in S\}$$

Example:

$$\{0, 17\} \xmapsto{\alpha_1} \{0, +1\} \xmapsto{\gamma_1} \{0, 1, \ldots, 17, \ldots\}$$

.../...

- The lattice of signs [4]:



.../...

Reference

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

- $\alpha_2 \in \wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto (\mathbb{X} \mapsto \wp(\mathbb{Z}))$

$$\alpha_2(R) \triangleq \boldsymbol{\lambda} X \cdot \{\rho(X) \mid \rho \in R\}$$

$\gamma_2 \in (\mathbb{X} \mapsto \wp(\mathbb{Z})) \mapsto \wp(\mathbb{X} \mapsto \mathbb{Z})$ (4)

$$\gamma_2(r) \triangleq \{\rho \mid \forall X \in \mathbb{X} : \rho(X) \in r(X)\}$$

Example:

$$\{[X : 0, Y : 0], [X : 5, Y : 5]\}$$
$$\xmapsto{\alpha_1} [X : \{0, 5\}, Y : \{0, 5\}]$$
$$\xmapsto{\gamma_1} \{[X : 0, Y : 0], [X : 0, Y : 5], [X : 5, Y : 0], [X : 5, Y : 5]\}$$

.../...

.../...

- $\alpha_3 : (\mathbb{X} \mapsto \wp(\mathbb{Z})) \mapsto (\mathbb{X} \mapsto \wp(\mathbb{S}))$

  $\alpha_3(\rho) = \boldsymbol{\lambda} X \cdot \alpha_1(\rho(X))$

  $\gamma_3 : (\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto (\mathbb{X} \mapsto \wp(\mathbb{Z}))$ (5)

  $\gamma_3(\rho) = \boldsymbol{\lambda} X \cdot \gamma_1(\rho(X))$

Example:

$$
\begin{array}{ll}
 & [X : \{0, 5\}, Y : \{0, 5\}] \\
\xmapsto{\alpha_3} & [X : \{0, +1\}, Y : \{0, +1\}] \\
\xmapsto{\gamma_3} & [X : \mathbb{N}, Y : \mathbb{N}]
\end{array}
$$

.../...

- $\alpha_4 : \wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto (\mathbb{X} \mapsto \wp(\mathbb{S}))$

  $\alpha_4 = \alpha_3 \circ \alpha_2$

  $\gamma_4 : (\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{X} \mapsto \mathbb{Z})$ (6)

  $\gamma_4 = \gamma_2 \circ \gamma_3$

Example:

$$
\begin{array}{ll}
 & \{[X : 0, Y : 0], [X : 5, Y : 5]\} \\
\xmapsto{\alpha_3} & [X : \{0, +1\}, Y : \{0, +1\}] \\
\xmapsto{\gamma_3} & \{[X : n, Y : m] \mid n \in \mathbb{N} \wedge m \in \mathbb{N}\}
\end{array}
$$

.../...

- $\alpha_5 \in (\wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto \wp(\mathbb{Z})) \longmapsto ((\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{S}))$ (7)

  $\alpha_5(s) \triangleq \alpha_1 \circ s \circ \gamma_4$

  $\gamma_5 \in ((\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{S})) \longmapsto (\wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto \wp(\mathbb{Z}))$

  $\gamma_5(S) \triangleq \gamma_1 \circ S \circ \alpha_4$

Intuition:

$$
\begin{tikzcd}
\wp(\mathbb{X} \mapsto \mathbb{Z}) \arrow[r, "s"] \arrow[d, shift right, "\alpha_4"'] & \wp(\mathbb{Z}) \arrow[d, shift right, "\alpha_1"'] \\
\mathbb{X} \mapsto \wp(\mathbb{S}) \arrow[r, "S"] \arrow[u, shift right, "\gamma_4"'] & \wp(\mathbb{S}) \arrow[u, shift right, "\gamma_1"']
\end{tikzcd}
$$

.../...

- $\alpha_6 \in \wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z}) \longmapsto (\wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto \wp(\mathbb{Z}))$ (8)

  $\alpha_6(S) \triangleq \boldsymbol{\lambda} R \cdot \{s(\rho) \mid s \in S \wedge \rho \in R\}$

  $\gamma_6 \in (\wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto \wp(\mathbb{Z})) \longmapsto \wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z})$

  $\gamma_6(S) \triangleq \{s \mid \forall \rho \in \mathbb{X} \mapsto \mathbb{Z} : s(\rho) \in S(\{\rho\})\}$

Intuition:

$$
\begin{array}{l}
\gamma_6(\alpha_6(\{s\})) \\
= \{s' \mid \forall \rho' \in \mathbb{X} \mapsto \mathbb{Z} : s'(\rho') \in \{s''(\rho'') \mid s'' \in \{s\} \wedge \rho'' \in \{\rho'\}\}\} \\
= \{s' \mid \forall \rho' \in \mathbb{X} \mapsto \mathbb{Z} : s'(\rho') \in \{s(\rho')\}\} \\
= \{s' \mid \forall \rho' \in \mathbb{X} \mapsto \mathbb{Z} : s'(\rho') = s(\rho')\} \\
= \{s' \mid s' = s\} \\
= \{s\}
\end{array}
$$

.../...

- $\alpha_7 \in \wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z}) \longmapsto ((\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{S}))$ (9)
  $\alpha_7 \stackrel{\Delta}{=} \alpha_5 \circ \alpha_6$
  $\gamma_7 \in ((\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{S})) \longmapsto \wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z})$
  $\gamma_7 \stackrel{\Delta}{=} \gamma_6 \circ \gamma_5$

Intuition:

$$\begin{tikzcd}
\wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z}) \arrow[d, shift right, "\alpha_6"'] \\
\wp(\mathbb{X} \mapsto \mathbb{Z}) \mapsto \wp(\mathbb{Z}) \arrow[u, shift right, "\gamma_6"'] \arrow[d, shift right, "\alpha_5"'] \\
(\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{S}) \arrow[u, shift right, "\gamma_5"']
\end{tikzcd}$$

Specification of the rule of signs abstract semantics

- Standard semantics: $[\![e]\!]$
- Collecting semantics: $\{\!|e|\!\} \stackrel{\Delta}{=} \{[\![e]\!]\}$
- Abstract semantics: $(\!|e|\!) \stackrel{\Delta}{=} \alpha_7(\{\!|e|\!\})$ best approximation
  $(\!|e|\!) \stackrel{.}{\supseteq} \alpha_7(\{\!|e|\!\})$ suboptimal approximation
- Example of suboptimal abstract semantics:
  - $\alpha_7(\{\!|X - X|\!\})[X : \{+1\}]$
    $= \alpha_7(\{\!|0|\!\})[X : \{+1\}]$
    $= \{0\}$
  - $(\!|X - X|\!)[X : \{+1\}]$
    $= (\!|X|\!)[X : \{+1\}] - (\!|X|\!)[X : \{+1\}]$
    $= \{+1\} - \{+1\}$
    $= \{\text{-}1,0,+1\}$

Calculational design of the abstract semantics

$(\!|e|\!)$
$= \alpha_7(\{\!|e|\!\})$
$= \alpha_5 \circ \alpha_6(\{\!|e|\!\})$ by def. (9) of α_7
$= \alpha_1 \circ \alpha_6(\{\!|e|\!\}) \circ \gamma_4$ by def. (7) of α_5
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{s(\rho) \mid s \in \{\!|e|\!\} \wedge \rho \in \gamma_4(R)\})$ by (8)
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e]\!]\rho \mid \rho \in \gamma_4(R)\})$ by def. (1) of $[\![e]\!]$
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e]\!]\rho \mid \rho \in \gamma_2 \circ \gamma_3(R)\})$ by (6)
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e]\!]\rho \mid \rho \in \gamma_2(\boldsymbol{\lambda} Y \cdot \gamma_1(R(Y)))\})$ by def. (5) of γ_3
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e]\!]\rho \mid \rho \in \{\rho' \mid \forall Y \in \mathbb{X} : \rho'(Y) \in \gamma_1(R(Y))\}\})$
by def. (4) of γ_2
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e]\!]\rho \mid \forall Y \in \mathbb{X} : \rho(Y) \in \gamma_1(R(Y))\})$ by def. $\in$

We go on by structural induction on e.

.../...

- $e \equiv n$:

$\boldsymbol{\lambda} R \cdot \alpha_1(\{[\![n]\!]\rho \mid \forall Y \in \mathbb{X} : \rho(Y) \in \gamma_1(R(Y))\})$
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{n\})$ by def. $[\![n]\!]\rho \stackrel{\Delta}{=} n$
$= \boldsymbol{\lambda} R \cdot \{\alpha_0(n)\}$ by def. (3) of α_1
$= \boldsymbol{\lambda} R \cdot \{-1\}$ if $n < 0$ by def. (2) of α_0
$\boldsymbol{\lambda} R \cdot \{0\}$ if $n = 0$
$\boldsymbol{\lambda} R \cdot \{+1\}$ if $n > 0$

- $e \equiv X$:

$\boldsymbol{\lambda} R \cdot \alpha_1(\{[\![X]\!]\rho \mid \forall Y \in \mathbb{X} : \rho(Y) \in \gamma_1(R(Y))\})$
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{\rho(X) \mid \forall Y \in \mathbb{X} : \rho(Y) \in \gamma_1(R(Y))\})$
by def. $[\![X]\!]\rho \stackrel{\Delta}{=} \rho(X)$
$= \boldsymbol{\lambda} R \cdot \alpha_1(\{\rho(X) \mid \rho(X) \in \gamma_1(R(X))\})$
$= \boldsymbol{\lambda} R \cdot \alpha_1(\gamma_1(R(X)))$
$= \boldsymbol{\lambda} R \cdot R(X)$

.../...

- $e \equiv e_1 + e_2$:

$$
\begin{aligned}
& \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e_1 + e_2]\!]\rho \mid \rho \in \gamma_4(R)\}) \\
=\ & \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e_1]\!]\rho + [\![e_2]\!]\rho \mid \rho \in \gamma_4(R)\}) \\
& \qquad\qquad\qquad \text{by def. } [\![e_1 + e_2]\!]\rho \stackrel{\Delta}{=} [\![e_1]\!]\rho + [\![e_2]\!]\rho \\
\subseteq\ & \boldsymbol{\lambda} R \cdot \alpha_1(\{[\![e_1]\!]\rho + [\![e_2]\!]\rho' \mid \rho \in \gamma_4(R) \wedge \rho' \in \gamma_4(R)\}) \\
=\ & \boldsymbol{\lambda} R \cdot \alpha_1(\{x + y \mid x \in \{[\![e_1]\!]\rho \mid \rho \in \gamma_4(R)\} \wedge \\
& \qquad\qquad y \in \{[\![e_2]\!]\rho' \mid \rho' \in \gamma_4(R)\}\}) \\
\subseteq\ & \boldsymbol{\lambda} R \cdot \alpha_1(\{x + y \mid x \in \gamma_1 \circ \alpha_1(\{[\![e_1]\!]\rho \mid \rho \in \gamma_4(R)\}) \wedge \\
& \qquad\qquad y \in \gamma_1 \circ \alpha_1(\{[\![e_2]\!]\rho' \mid \rho' \in \gamma_4(R)\})\}) \\
\subseteq\ & \boldsymbol{\lambda} R \cdot \alpha_1(\{x + y \mid x \in \gamma_1((\!|e_1|\!)R) \wedge y \in \gamma_1((\!|e_2|\!)R)\}) \quad \text{by ind. hyp.} \\
=\ & \boldsymbol{\lambda} R \cdot ((\!|e_1|\!)R + (\!|e_2|\!)R) \text{ where } + \text{ is calculated by cases:}
\end{aligned}
$$

.../...

- $\alpha_1(\{x + y \mid x \in \gamma_1(+1) \wedge y \in \gamma_1(+1)\})$
  $= \alpha_1(\{x + y \mid x \in \mathbb{N}^+ \wedge y \in \mathbb{N}^+\})$
  $= \alpha_1(\mathbb{N}^+)$
  $= \{+1\}$ so that $\{+1\} + \{+1\} = \{+1\}$
- $\alpha_1(\{x + y \mid x \in \gamma_1(-1) \wedge y \in \gamma_1(+1)\})$
  $= \alpha_1(\{x + y \mid x \in \mathbb{N}^+ \wedge y \in \mathbb{N}^-\})$
  $= \alpha_1(\mathbb{Z})$
  $= \{-1, 0, +1\}$ so that $\{-1\} + \{+1\} = \{-1, 0, +1\}$
- etc.

Reference

[5] P. Cousot. The calculational design of a generic abstract interpreter. In M. Broy and R. Steinbrüggen, editors, *Calculational System Design*, volume 173, pages 421–505. NATO Science Series, Series F: Computer and Systems Sciences. IOS Press, 1999.

Summary of the abstract semantics

- $(\!|n|\!)R = \begin{cases} \{-1\} & \text{if } n < 0 \\ \{0\} & \text{if } n = 0 \\ \{+1\} & \text{if } n > 0 \end{cases}$
- $(\!|X|\!)R = R(x)$
- $(\!|e_1 + e_2|\!)R = (\!|e_1|\!)R + (\!|e_2|\!)R$

  where the “rule of signs”for addition $+$ is:

$+$	$\emptyset$	$\{-1\}$	$\{0\}$	$\{+1\}$	$\{-1,0\}$	$\{-1,+1\}$	$\{0,+1\}$	$\{-1,0,+1\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{-1,0,+1\}$
$\{-1\}$	$\emptyset$	$\{-1\}$	$\{-1\}$	$\{-1,0,+1\}$	$\{-1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$
$\{0\}$	$\emptyset$	$\{-1\}$	$\{0\}$	$\{+1\}$	$\{-1,0\}$	$\{-1,+1\}$	$\{0,+1\}$	$\{-1,0,+1\}$
$\{+1\}$	$\emptyset$	$\{-1,0,+1\}$	$\{+1\}$	$\{+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{+1\}$	$\{-1,0,+1\}$
$\{-1,0\}$	$\emptyset$	$\{-1\}$	$\{-1,0\}$	$\{-1,0,+1\}$	$\{-1,0\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$
$\{-1,+1\}$	$\emptyset$	$\{-1,0,+1\}$	$\{-1,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$
$\{0,+1\}$	$\emptyset$	$\{-1,0,+1\}$	$\{0,+1\}$	$\{+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$
$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$	$\{-1,0,+1\}$

2. Formalization of abstraction/concretization

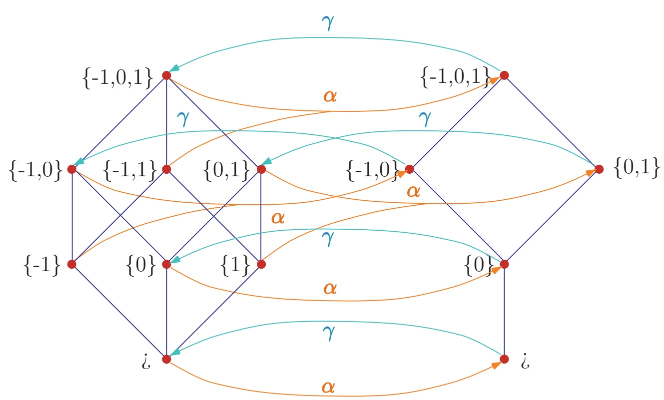
Galois connection — 1

The paire $\langle\alpha, \gamma\rangle$ is a *Galois connection*:

- α is $\subseteq$-monotone
  $\Rightarrow$ abstraction preserve implication;
- γ is $\subseteq$-monotone
  $\Rightarrow$ concretization preserves implication;
- $\gamma \circ \alpha$ is $\subseteq$-extensive
  $\Rightarrow$ an abstraction introduces a loss of information;
- $\alpha \circ \gamma$ is $\subseteq$-reductive
  $\Rightarrow$ a concretization can only be more precise.

Notation: $\langle \wp((\mathbb{X} \mapsto \mathbb{Z}) \mapsto \mathbb{Z}), \subseteq\rangle \underset{\alpha_7}{\overset{\gamma_7}{\leftrightarrows}} \langle (\mathbb{X} \mapsto \wp(\mathbb{S})) \mapsto \wp(\mathbb{S}), \dot{\subseteq}\rangle$

- Example of Galois connection based abstraction:



Galois connection — 2

An equivalent definition of

$$\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$$

is

- $\langle L, \leq\rangle$ and $\langle M, \sqsubseteq\rangle$ are posets;
- $\forall x \in L : \forall y \in M : \alpha(x) \sqsubseteq y \Leftrightarrow x \leq \gamma(y)$.

A few properties of Galois connections

- One function uniquely determine the other:
  $\alpha(x) = \sqcap\{y \mid x \leq \gamma(y)\}$
  $\gamma(y) = \sqcup\{x \mid \alpha(x) \sqsubseteq y\}$
- α has an adjoint iff it preserves existing lubs;
- γ has an adjoint iff it preserves existing glbs;

Surjections/injections

- α is surjective $\Leftrightarrow \gamma$ is injective $\Leftrightarrow \alpha \circ \gamma = \mathbf{1}$:

$$\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\substack{\longleftarrow \\ \twoheadrightarrow}}} \langle M, \sqsubseteq\rangle$$

- Assuming α surjective simplifies the formal presentation (not always possible in practice);
- Dually, α is injective $\Leftrightarrow \gamma$ is surjective $\Leftrightarrow \gamma \circ \alpha = \mathbf{1}$:

$$\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\substack{\twoheadleftarrow \\ \longrightarrow}}} \langle M, \sqsubseteq\rangle$$

Closure operators

If

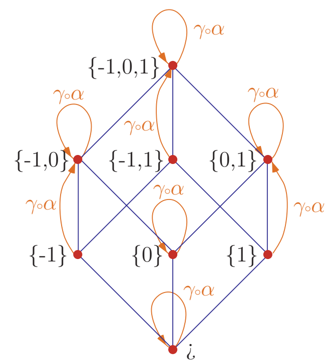
$$\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$$

- $\gamma \circ \alpha$ represents the approximation of concrete properties by a concrete representation of the abstract properties;
- $\gamma \circ \alpha$ is an upper closure operator:
  - monotone,
  - extensive,
  - idempotent.
- A formally equivalent formalization of abstraction [6, Sections 6.2].

Reference

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

- Example of closure operator based abstraction:



Moore family

If

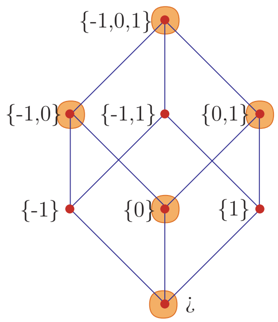
$$\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$$

- $\gamma \circ \alpha(M)$ represents the set of concrete representations of the abstract properties;
- $\gamma \circ \alpha(M)$ is a Moore family:
  - contains a top element (if M has a supremum),
  - closed by arbitrary intersections.
- A formally equivalent formalization of abstraction [7, Sections 6.1].

Reference

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

- Example of Moore family based abstraction:

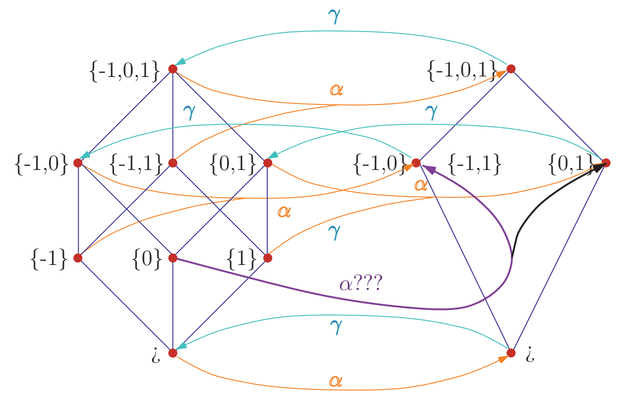


Best approximation

- $\gamma \circ \alpha(P)$ is the concrete representation of the best abstract approximation of P:
  - Its an upper-approximation: $P \leq \gamma \circ \alpha(P)$;
  - Its the best abstraction: if is another abstract approximation (i.e. $Q \in \gamma \circ \alpha(L)$ and $P \leq Q$) then $\gamma \circ \alpha(P)$ is more precise (in that $\gamma \circ \alpha(P) \leq Q$).
- The best approximation does exists and is unique (by antisymmetry).

In absence of best approximation?

The classical rule of signs has no $\{0\}$ [1]:



.../...

[1] because in practice one makes the algebraic simplification $x + 0 = 0 + x = 0$.

In absence of best approximation (continued)

- The best choice must be determined during this analysis:
  - $\alpha(\{0\}) = \{-1, 0\}$ is a better choice in $(\!|0 + -1|\!)R = \{-1, 0\}$
  - $\alpha(\{0\}) = \{+1, 0\}$ is a better choice in $(\!|0 + 1|\!)R = \{0, +1\}$
- In practice, one uses γ only and widening/narrowing operators [8] as e.g. in [9];
- Other alternatives (e.g. use a soundness relation) discussed in [10].

References

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

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

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

Compound abstraction [2]

- The abstraction is designed by composition:
  - of primitive abstractions;
  - of abstraction composition operators.

[2] In french we would use the term "compositional", which in the context of denotational semantics is already used to mean "by structural induction on the abstract syntax".

Example of primitive abstractions

- Elementwise abstraction: if $@ \in S \mapsto S^\sharp$ then:

$$\langle \wp(S), \subseteq \rangle \underset{\alpha_@}{\overset{\gamma_@}{\longleftrightarrow}} \langle \wp(S^\sharp), \subseteq \rangle$$

where:

$$\begin{aligned} \alpha_@(X) &\triangleq \{@(x) \mid x \in X\} \\ \gamma_@(Y) &\triangleq \{x \mid @(x) \in Y\} \end{aligned}$$

Examples of abstraction composition operators

- Abstraction composition: if $\langle L, \leq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\longleftrightarrow}} \langle M, \sqsubseteq \rangle$ and $\langle M, \sqsubseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\longleftrightarrow}} \langle N, \preceq \rangle$ then:

$$\langle L, \leq \rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\longleftrightarrow}} \langle N, \preceq \rangle$$

- Functional abstraction: if $\langle L, \leq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\longleftrightarrow}} \langle L^\sharp, \leq^\sharp \rangle$ and $\langle M, \sqsubseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\longleftrightarrow}} \langle M^\sharp, \sqsubseteq^\sharp \rangle$ then:

$$\langle L \xmapsto{\text{mon}} M, \dot{\sqsubseteq} \rangle \underset{\boldsymbol{\lambda} F \cdot \gamma_1 \circ F \circ \alpha_2}{\overset{\boldsymbol{\lambda} f \cdot \alpha_2 \circ f \circ \gamma_1}{\longleftrightarrow}} \langle L^\sharp \xmapsto{\text{mon}} M^\sharp, \dot{\sqsubseteq}^\sharp \rangle$$

Fixpoint transfer and approximation

[11, p. 309]: If

- $\langle L, \leq, 0, \vee \rangle$ is a cpo,
- $\langle L, \leq \rangle \underset{\alpha}{\overset{\gamma}{\longleftrightarrow}} \langle M, \sqsubseteq \rangle$,
- $F \in L \xmapsto{\text{mon}} L$,
- $G \in M \xmapsto{\text{mon}} M$
- $\alpha \circ F =/\sqsubseteq G \circ \alpha$ local completeness/approximation

then

$\langle M, \sqsubseteq, \perp, \sqcup \rangle$ is a cpo where $\perp = \alpha(0)$ and $\sqcup X = \alpha(\vee \gamma(X))$,

$\alpha \circ F \circ \gamma =/\sqsubseteq G$,

$\alpha(\mathrm{lfp}^{\leq} F) =/\sqsubseteq \mathrm{lfp}^{\sqsubseteq} G$.

Reference

[11] 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.

The lattice of abstract interpretations

- The abstract interpretations of a semantics are isomorphic to closure operators;
- So the complete lattice of abstract interpretations [12] is isomorphic to the complete lattice of closure operators on a complete lattice/cpo.

References

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

Soundness and completeness [3]

- $\{\!|P|\!\}$: collecting semantics of P
- $\langle L, \leq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle M, \sqsubseteq\rangle$: abstraction
- $(\!|P|\!)$: abstract semantics of P
- Soundness:

$$\forall P : \alpha(\{\!|P|\!\}) \sqsubseteq (\!|P|\!)$$

- (Global) completeness:

$$\forall P : \alpha(\{\!|P|\!\}) = (\!|P|\!)$$

[3] We should say *relative completeness* to stress the fact that we reason in set theoretical terms, so that, in logical terms, an oracle is assumed to exist for logical implication.

3. Abstraction/concretization in program analysis

Data flow analysis is an abstract interpretation

From [13, section 7.2.0.6.3]:

- $\tau = \langle \mathbb{S}, \mathbb{A}, t\rangle$: transition system ($\mathbb{S} \triangleq \mathbb{C} \times \mathbb{M}$ control $\times$ memory states, actions $\mathbb{A}$ are assignments and tests)
- $\mathbb{T}$: traces $\langle s_0, a_0, s_1\rangle\langle s_1, a_1, s_2\rangle \ldots \langle s_{n-1}, a_{n-1}, s_n\rangle$, $s_i \in \mathbb{S}$, $a_i \in \mathbb{A}$
- $\mathcal{M}_\tau \subseteq \mathbb{T}$: program semantics (set of prefix closed finite traces generated by τ);
- $\mathbb{E}$: set of expressions appearing in actions $\mathbb{A}$;

Reference

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

- Static partitioning abstraction [14]:

$$\alpha_p(M) \triangleq \prod_{\ell \in \mathbb{C}} \{\sigma'\langle s, a, \langle \ell, m\rangle\rangle \mid \sigma'\langle s, a, \langle \ell, m\rangle\rangle \in M\}$$

such that:

$$\langle \wp(\mathbb{T}), \subseteq\rangle \underset{\alpha_p}{\overset{\gamma_p}{\leftrightarrows}} \langle \mathbb{C} \mapsto \wp(\mathbb{T}), \dot{\subseteq}\rangle$$

- Pointwise abstraction: If $\langle \wp(\mathbb{T}), \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L, \sqsubseteq\rangle$ then:

$$\langle \mathbb{C} \mapsto \wp(\mathbb{T}), \dot{\subseteq}\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle \mathbb{C} \mapsto L, \dot{\sqsubseteq}\rangle$$

where: $\qquad \dot{\alpha}(M) \triangleq \boldsymbol{\lambda}\, \ell \cdot \alpha(M(\ell))$

Reference

[14] 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.

- *gen*(a): expressions $e \in \mathbb{E}$ generated by assignment/test $a \in \mathbb{A}$;
- *kill*(a): expressions $e \in \mathbb{E}$ killed by assignment/test $a \in \mathbb{A}$;
- Definition of availability at exit of a path σ:

$$\mathit{avail}(\sigma) = (\sigma = \sigma'\langle s, a, s'\rangle \;\textbf{?}\; (\mathit{avail}(\sigma') \cap \neg \mathit{kill}(a)) \cup \mathit{gen}(a) \;\textbf{:}\; \emptyset)$$

- Availability abstraction (availability at exit of all paths of M):

$$\alpha_a(M) \triangleq \cap\{\mathit{avail}(\sigma) \mid \sigma \in M\}$$

such that:

$$\langle \wp(\mathbb{T}), \subseteq\rangle \underset{\alpha_a}{\overset{\gamma_a}{\leftrightarrows}} \langle \wp(\mathbb{E}), \supseteq\rangle$$

- Availability at all points $\ell \in \mathbb{C}$:

$$\dot{\alpha}_a \circ \alpha_p(\mathcal{M}_\tau).$$

Program static analysis is an abstract interpretation

- $\mathbb{S} \triangleq \mathbb{C} \times \mathbb{Z}^n$ — States (finite control states $\mathbb{C}$)
- $\mathbb{P} \triangleq \wp(\mathbb{S})$ — Properties
- $\mathbb{I} \triangleq \{[a, b] \mid a \leq b\} \cup \{\bot\}$ — Interval abstract domains
- $\langle \wp(\mathbb{Z}), \subseteq\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle \mathbb{I}, \leq\rangle$ — Interval abstraction

  $\alpha_1(Z) \triangleq (Z = \emptyset \;\textbf{?}\; \bot \;\textbf{:}\; [\min Z, \max Z])$ $\qquad (\min \mathbb{Z} = -\infty,\ \max \mathbb{Z} = \infty)$
- $\langle \mathbb{P}, \subseteq\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle \mathbb{C} \mapsto ([1, n] \mapsto \mathbb{I}), \dot{\leq}\rangle$ — State abstraction

$$\alpha_2(S) \triangleq \prod_{\ell \in \mathbb{C}} \prod_{i=1}^{n} \alpha_1(\{z \mid \exists x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n : \langle \ell, \langle x_1, \ldots, x_{i-1}, z, x_{i+1}, \ldots, x_n\rangle\rangle \in S\})$$


```
{ n:_O_; i:_O_ }
  n := ?;
{ n:[-oo,+oo]; i:_O_ }
  i := 1;
{ n:[-oo,+oo]; i:[1,+oo] }
  while (i < n) do
    { n:[2,+oo]; i:[1,1073741822] }
      i := (i + 1)
    { n:[2,+oo]; i:[2,+oo] }
  od
{ n:[-oo,+oo]; i:[1,+oo] }
```

References

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

[16] P. Cousot. The Marktoberdorf'98 generic abstract interpreter. 1998. http://www.di.ens.fr/~cousot/Marktoberdorf98.shtml, Nov.

[17] P. Lacan, J.N. Monfort, Le Vinh Quy Ribal, A. Deutsch, and G. Gonthier. The software reliability verification process: The ARIANE 5 example. In *Proceedings DASIA 98 – DAta Systems IN Aerospace*, Athens, GR. ESA Publications, SP-422, 25–28 May 1998.

Grammar analysis is an abstract interpretation

- $G = \langle N, T, P, A\rangle$ context free grammar
- The semantics $[\![G]\!]$ of G is the terminal language generated by G;
- The FIRST algorithm is an abstract interpretation of the grammar fixpoint semantics [18] by:

$$\langle T^\star, \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(T \cup \{\epsilon\,^{4}\}), \subseteq\rangle$$

where:

$$\begin{aligned}\alpha(L) &\triangleq \{@(\sigma) \mid \sigma \in L\}\\ @(\epsilon) &\triangleq \epsilon\\ @(x\sigma) &\triangleq x\end{aligned}$$

Reference

[18] P. Cousot and R. Cousot. Abstract interpretation of algebraic polynomial systems. In M. Johnson, ed., *Proc. 6th Int. Conf. AMAST '97*, Sydney, AU, LNCS 1349, pages 138–154. Springer-Verlag, 13–18 Dec. 1997.

[4] ϵ is the empty string.

Set based analysis is an abstract interpretation

- Concrete domain: set $\mathbb{T}$ of tree on a finite signature $\mathbb{F} = \bigcup_{n\geq 0} \mathbb{F}^n$;
- Abstract domain: regular tree grammars G in Greibach normal form:

$$\begin{cases} \mathcal{X} ::= \boldsymbol{f}^n(\mathcal{Y}_1, \ldots, \mathcal{Y}_n) & \boldsymbol{f}^n \in \mathbb{F}^n\\ \mathcal{X} ::= \boldsymbol{f}^0 & \boldsymbol{f}^0 \in \mathbb{F}^0\end{cases}$$

where non-terminals $\mathcal{X}, \ldots$ correspond to program elements (variables, etc...)

- Concretization: $\gamma(G) \triangleq$ the set of finite trees generated by the grammar G;

Reference

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

- No best approximation ($\boldsymbol{f} \in \mathbb{F}^3$, $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c} \in \mathbb{F}^1$, $\boldsymbol{n} \in \mathbb{F}^0$):

$$\begin{aligned}\mathcal{X}_0 &\triangleq \emptyset\\ \mathcal{X}_1 &\triangleq \{\boldsymbol{f}(\boldsymbol{a}(\boldsymbol{n}), \boldsymbol{b}(\boldsymbol{n}), \boldsymbol{c}(\boldsymbol{n}))\}\\ \mathcal{X}_2 &\triangleq \{\boldsymbol{f}(\boldsymbol{a}(\boldsymbol{n}), \boldsymbol{b}(\boldsymbol{n}), \boldsymbol{c}(\boldsymbol{n})), \boldsymbol{f}(\boldsymbol{a}^2(\boldsymbol{n}), \boldsymbol{b}^2(\boldsymbol{n}), \boldsymbol{c}^2(\boldsymbol{n}))\}\\ &\ldots\ \ldots\ \ldots\\ \mathcal{X}_k &\triangleq \{\boldsymbol{f}(\boldsymbol{a}(\boldsymbol{n}), \boldsymbol{b}(\boldsymbol{n}), \boldsymbol{c}(\boldsymbol{n})), \ldots, \boldsymbol{f}(\boldsymbol{a}^k(\boldsymbol{n}), \boldsymbol{b}^k(\boldsymbol{n}), \boldsymbol{c}^k(\boldsymbol{n}))\}\\ &\ldots\ \ldots\ \ldots\\ \mathcal{X}_\omega &\triangleq \bigcup_{k\geq 0} \mathcal{X}_k = \{\boldsymbol{f}(\boldsymbol{a}^k(\boldsymbol{n}), \boldsymbol{b}^k(\boldsymbol{n}), \boldsymbol{c}^k(\boldsymbol{n})) \mid k \geq 0\}\end{aligned}$$

is not context free

- The $\mathcal{X}_k$, $k \geq 0$ can all be described by a regular tree grammar but not $\mathcal{X}_\omega$. So $\mathcal{X}_\omega$ is approximated by the regular language:

$$\{\boldsymbol{f}(\boldsymbol{a}^k(\boldsymbol{n}), \boldsymbol{b}^\ell(\boldsymbol{n}), \boldsymbol{c}^m(\boldsymbol{n})) \mid k, \ell, m \geq 0\}.$$

Type inference is an abstract interpretation

- Concrete standard domain for λ-expressions:

$\mathbb{W} \triangleq \{\omega\}$	wrong
$z \in \mathbb{Z}$	integers
$v, \varphi \in \mathbb{U} \cong \mathbb{W}_\perp \oplus \mathbb{Z}_\perp \oplus (\mathbb{U} \overset{\text{con}}{\longmapsto} \mathbb{U})_\perp$	domain of values
$\mathtt{x} \in \mathbb{X}$	program variables
$R \in \mathbb{R} \triangleq \mathbb{X} \mapsto \mathbb{U}$	environments
$\phi \in \mathbb{S} \triangleq \mathbb{R} \mapsto \mathbb{U}$	λ-expression type

Reference

[20] P. Cousot. Types as abstract interpretations, invited paper. In *24th POPL*, pages 316–331, Paris, FR, Jan. 1997. ACM Press.

- Concrete collecting domain:

$$\mathbb{P} \triangleq \wp(\mathbb{S})$$

- Abstract domain:

$m \in \mathbb{M}$	$m ::= \texttt{int} \mid m_1 -> m_2$	Church/Curry/Hindley monotype
$H \in \mathbb{H} \triangleq \mathbb{X} \mapsto \mathbb{M}$		type environments
$\theta \in \mathbb{I} \triangleq \mathbb{H} \times \mathbb{M}$		typings
$T \in \mathbb{T} \triangleq \wp(\mathbb{I})$		program types

- In general a λ-expression has (infinitely) many types:

$$\lambda\texttt{x.x} \quad \text{has types} \quad \{\langle H, m -> m\rangle \mid H \in \mathbb{H} \wedge m \in \mathbb{M}\}$$

- Concretization:

$$\begin{aligned}
\gamma &\in \mathbb{M} \mapsto \wp(\mathbb{U})\\
\gamma(\texttt{int}) &\triangleq \mathbb{Z} \cup \{\bot\}\\
\gamma(m_1 -> m_2) &\triangleq \{\phi \in \mathbb{U} \overset{\text{con}}{\longmapsto} \mathbb{U} \mid \forall v \in \gamma(m_1) : \phi(v) \in \gamma(m_2)\} \cup \{\bot\}\\
\gamma &\in \mathbb{H} \mapsto \wp(\mathbb{R})\\
\gamma(H) &\triangleq \{R \in \mathbb{R} \mid \forall \texttt{x} \in \mathbb{X} : R(\texttt{x}) \in \gamma(H(\texttt{x}))\}\\
\gamma &\in \mathbb{I} \mapsto \wp(\mathbb{S})\\
\gamma\langle H, m\rangle &\triangleq \{\phi \in \mathbb{S} \mid \forall R \in \gamma(H) : \phi(R) \in \gamma(m)\}\\
\gamma &\in \mathbb{T} \mapsto \wp(\mathbb{S})\\
\gamma(T) &\triangleq \bigcap_{\theta \in T} \gamma(\theta)\\
\gamma(\emptyset) &\triangleq \mathbb{S}
\end{aligned}$$

- Galois connection:

$$\langle \mathbb{P}, \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \mathbb{T}, \supseteq\rangle$$

- Typable programs cannot go wrong: if a λ-expression e has type T and $T \neq \emptyset$ then $\forall R \in \mathbb{R} : [\![e]\!]R \neq \omega$ (since $[\![e]\!] \in \gamma(T)$);

Model checking is an abstract interpretation

- $\langle \mathbb{S}, t\rangle$: transition system ($t \subseteq \mathbb{S}^2$);
- $\Sigma_{\mathbb{S}}$: set of traces on states $\mathbb{S}$;
- $\mathcal{M}_t \subseteq \Sigma_{\mathbb{S}}$: model (set of maximal traces) generated by t;
- $X_{\downarrow s} \triangleq \{s\sigma \mid s\sigma \in X\}$ subset of traces of $X \subseteq \Sigma_{\mathbb{S}}$ starting with state s
- $\varphi \in \mathbb{L}$: formulae of temporal logic $\mathbb{L}$;
- $[\![\varphi]\!] \subseteq \Sigma_{\mathbb{S}}$: semantics of the closed temporal formula (set of maximal traces);

Reference

[21] P. Cousot and R. Cousot. Temporal abstract interpretation. In *27th POPL*, Boston, Mass., Jan. 2000. ACM Press. To appear.

- Boolean universal model checking is $\alpha_t^{\forall}(\llbracket\varphi\rrbracket)$

$$\langle\wp(\Sigma), \supseteq\rangle \underset{\alpha_t^{\forall}}{\overset{\gamma_t^{\forall}}{\leftrightarrows}} \langle\wp(\mathbb{S}), \supseteq\rangle$$

$$\alpha_t^{\forall}(\Phi) \triangleq \{s \in \mathbb{S} \mid \mathcal{M}_{t\downarrow s} \subseteq \Phi\}$$

- Boolean existential model checking is dual ($\alpha_t^{\exists}(\Phi) = \neg\alpha_t^{\forall}(\neg\Phi)$) so:

$$\langle\wp(\Sigma), \subseteq\rangle \underset{\alpha_t^{\exists}}{\overset{\gamma_t^{\exists}}{\leftrightarrows}} \langle\wp(\mathbb{S}), \subseteq\rangle$$

$$\alpha_t^{\exists}(\Phi) \triangleq \{s \in \mathbb{S} \mid (\mathcal{M}_{t\downarrow s} \cap \Phi) \neq \emptyset\}$$

Abstract model checking is the composition of abstract interpretations

- State abstraction:

$$\langle\wp(\mathbb{S}), \subseteq\rangle \underset{\alpha_s}{\overset{\gamma_s}{\leftrightarrows}} \langle\wp(\mathbb{S}^{\sharp}), \subseteq\rangle$$

- Trace-based model abstraction:

$$\langle\wp(\Sigma), \subseteq\rangle \underset{\alpha_m}{\overset{\gamma_m}{\leftrightarrows}} \langle\wp(\Sigma^{\sharp}), \subseteq\rangle$$

- Abstract model checking:

$$\alpha_t^{\forall\sharp} \triangleq \alpha_s \circ \alpha_t^{\forall} \circ \gamma_m$$

4. Abstraction/concretization in semantics

Semantics are abstract interpretations

- The various semantics of programming languages can be understood as abstract interpretations of a maximal trace semantics;
- (and the fixpoint characterizations of these semantics can all be constructively derived from the maximal trace semantics generated by a transition system (i.e. small-step operational semantics), see [22]).

Reference

[22] 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. (Full version to appear in TCS.)

Maximal trace semantics

- The basic maximal trace semantics is:

$$\mathcal{T}^\natural = \{\bullet \longrightarrow \bullet \longrightarrow \bullet \ldots \bullet \longrightarrow \bullet \longrightarrow \overset{\text{blocking state}}{\bullet}\} \quad \longleftarrow \text{finite traces}$$

$$\cup \; \{\underset{\text{any state}}{\bullet} \longrightarrow \bullet \longrightarrow \bullet \ldots \bullet \underset{\text{transition}}{\longrightarrow} \bullet \longrightarrow \bullet \ldots\} \quad \longleftarrow \text{infinite traces}$$

- $\mathcal{T}^\natural \in \wp(\mathbb{T})$, $\mathbb{T}$ is the set of traces over states $\mathbb{S}$.

Transition semantics

- Transition semantics:

$$\alpha(\mathcal{T}^\natural) = \{\langle \overset{a}{\bullet}, \overset{b}{\bullet} \rangle \mid \bullet \longrightarrow \bullet \longrightarrow \bullet \ldots \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{b}{\bullet} \longrightarrow \ldots \bullet \longrightarrow \bullet \in \mathcal{T}^\natural\}$$

$$\cup \; \{\langle \overset{a'}{\bullet}, \overset{b'}{\bullet} \rangle \mid \bullet \longrightarrow \bullet \ldots \bullet \longrightarrow \overset{a'}{\bullet} \longrightarrow \overset{b'}{\bullet} \longrightarrow \bullet \ldots \bullet \longrightarrow \ldots \in \mathcal{T}^\natural\}$$

- α is a Galois connection: $\langle \wp(\mathbb{T}), \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle \wp(\mathbb{S} \times \mathbb{S}), \subseteq \rangle$
- This is an approximation. For example, fairness information is lost:
  if $\mathcal{T}^\natural = a^\star b$ then $\gamma \circ \alpha(\mathcal{T}^\natural) = a^\star b \mid a^\omega$.

Natural, demoniac & angelic semantics

- Natural trace semantics: $\mathcal{T}^\natural$;
- Angelic abstraction[5]:

$$\alpha(\mathcal{T}^\natural) = \{\bullet \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \bullet \mid \bullet \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \bullet \in \mathcal{T}^\natural\};$$

- Demoniac abstraction[6]:

$$\alpha(\mathcal{T}^\natural) = \mathcal{T}^\natural \cup \{\bullet \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \bullet \mid \bullet \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \bullet \rightarrow \ldots \in \mathcal{T}^\natural\}.$$

The α's are Galois connections.

[5] Eliminate all infinite traces.
[6] Introduce all arbitrary finite traces for states possibly starting an infinite trace.

Relational semantics

$$\alpha \in \wp(\mathbb{T}) \longmapsto \wp(\mathbb{S} \times \mathbb{S}_\perp), \qquad \mathbb{S}_\perp = \mathbb{S} \cup \{\perp\}$$

$$\mathcal{R} = \alpha(\mathcal{T})$$

$$= \{\langle \overset{a}{\bullet}, \overset{b}{\bullet} \rangle \mid \overset{a}{\bullet} \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \overset{b}{\bullet} \in \mathcal{T}\}$$

$$\cup \; \{\langle \overset{a}{\bullet}, \perp \rangle \mid \overset{a}{\bullet} \rightarrow \bullet \rightarrow \ldots \rightarrow \bullet \rightarrow \bullet \rightarrow \ldots \in \mathcal{T}\}$$

α is a Galois connection.

Non-deterministic denotational semantics

$$\alpha \in \wp(\mathbb{S} \times \mathbb{S}_\perp) \longmapsto (\mathbb{S} \mapsto \wp(\mathbb{S}_\perp))$$

$$\begin{aligned}\mathcal{D} &= \alpha(\mathcal{R}) \\ &= \boldsymbol{\lambda} s \cdot \{s' \in \mathbb{S}_\perp \mid \langle s, s' \rangle \in \mathcal{R}\} \quad \text{right image}\end{aligned}$$

α is a Galois isomorphism.

Predicate transformer semantics

$$\alpha \in (\mathbb{S} \xmapsto{\text{mon}} \wp(\mathbb{S}_\perp)) \longmapsto (\wp(\mathbb{S}_\perp) \mapsto \wp(\mathbb{S}))$$

$$\begin{aligned}\mathcal{W} &= \alpha(\mathcal{D}) \\ &= \boldsymbol{\lambda} Q \cdot \{s \in \mathbb{S} \mid \forall s' \in \mathbb{S}_\perp : s' \in \mathcal{D}(s) \Rightarrow s' \in Q\}\end{aligned}$$

α is a Galois injection.

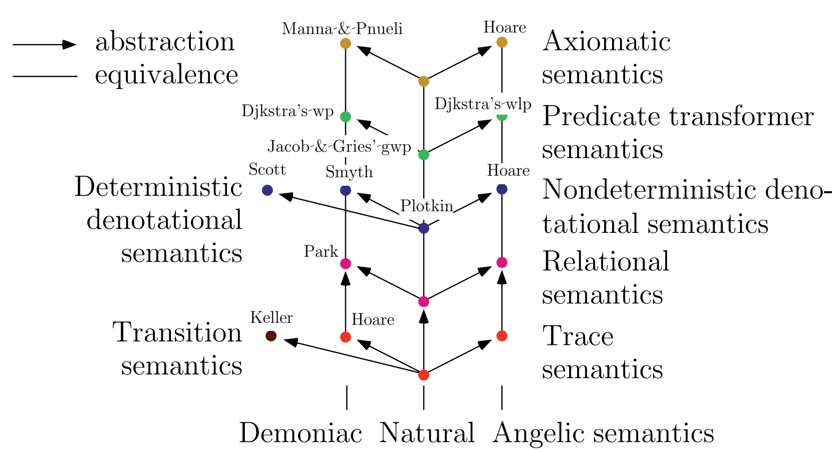
Axiomatic semantics

$$\alpha \in (\wp(\mathbb{S}) \xmapsto{\text{mon}} \wp(\mathbb{S}_\perp)) \longmapsto \wp(\wp(\mathbb{S}) \times \wp(\mathbb{S}_\perp))$$

$$\begin{aligned}\mathcal{H} &= \alpha(\mathcal{W}) \\ &= \{\langle P, Q \rangle \mid P \subseteq \mathcal{W}(Q)\}\end{aligned}$$

α is a Galois injection.

The hierarchy of semantics



5. Conclusion

What is abstract interpretation?

- Semantics as well as program analysis algorithms approximate the incomputable collection of all possible behaviors of programs on computers [24];
- Abstract interpretation is a theory of abstraction understood as a discrete approximation of computer system behavior specifications [23];

Reference

[23] P. Cousot. Abstract interpretation. *Symposium on Models of Programming Languages and Computation, ACM Comput. Surv.*, 28(2):324–328, 1996.

[24] P. Cousot. Program analysis: The abstract interpretation perspective. *ACM Comput. Surv.*, 28A(4es):165–es, Dec. 1996.

On the coincidence of abstraction/refinement in program verification and abstraction/concretization in abstract interpretation (tentative)

- Abstraction/refinement in program verification:

  α: concrete object $\longmapsto$ abstract object
- Abstraction/concretization in abstract interpretation:

  α: $\wp$(concrete object) $\longmapsto$ $\wp$(abstract object)
- Coincidence:
  - Lift the reasoning on objects to reasoning on object properties (e.g. using predicate transformers) ???
  - Use category theory (various attempts that we made did not bring any new practical idea) ???

- What I know about refinement [7]:
  - $c : C \mapsto C$ concrete operation $a : A \mapsto A$ abstract operation
  - $\alpha : C \mapsto A$ abstraction
  - $\alpha \circ c = a \circ \alpha$ refinement condition (10)
- Lifting to sets:
  - $\alpha^\sharp(X) \triangleq \{\alpha(x) \mid x \in X\}$ so $\langle \wp(C), \subseteq \rangle \xrightleftharpoons[\alpha^\sharp]{\gamma^\sharp} \langle \wp(A), \subseteq \rangle$
  - $c^\sharp(X) \triangleq \{c(x) \mid x \in X\}$
  - $a^\sharp(Y) \triangleq \{a(y) \mid x \in Y\}$

  so that (10) implies: $\alpha^\sharp \circ c^\sharp = a^\sharp \circ \alpha^\sharp$. If α is surjective then $a^\sharp = \alpha^\sharp \circ c^\sharp \circ \gamma^\sharp$ (i.e. $a^\sharp$ is the abstract interpretation of $c^\sharp$).
- So (?) reasoning on objects in program development by refinement is "equivalent to" reasoning on their local properties (i.e. topos theory is relevant)?

[7] nothing, so its what I guess about refinement!

THE END

Contents

Abstraction in abstract interpretation . . . 1
Abstract . . . 2

1. Introductive example 3
Abstract interpretation . . . 4
Abstraction in abstract interpretation . . . 5
Objects and their properties . . . 6
Properties as sets . . . 7
Example: rule of signs standard semantics . . . 8
Example: rule of signs — 1 — the abstract object point of view . . . 10
Example: rule of signs — 2 — the abstract property point of view . . . 12
Approximation . . . 13
Approximation for the rule of signs . . . 14
Specification of the rule of signs abstract semantics . . . 22
Calculational design of the abstract semantics . . . 23

Summary of the abstract semantics . . . 27

2. Formalization of abstraction/concretization 28
Galois connection — 1 . . . 29
Galois connection — 2 . . . 31
A few properties of Galois connections . . . 32
Surjections/injections . . . 33
Closure operators . . . 34
Moore family . . . 36
Best approximation . . . 38
In absence of best approximation? . . . 39
In absence of best approximation (continued) . . . 40
Compound abstraction . . . 41
Example of primitive abstractions . . . 42
Examples of abstraction composition operators . . . 43
Fixpoint transfer and approximation . . . 44
The lattice of abstract interpretations . . . 45

Soundness and completeness . . . 46

3. Abstraction/concretization in program analysis 47
Data flow analysis is an abstract interpretation . . . 48
Program static analysis is an abstract interpretation . . . 51
Grammar analysis is an abstract interpretation . . . 53
Set based analysis is an abstract interpretation . . . 54
Type inference is an abstract interpretation . . . 56
Model checking is an abstract interpretation . . . 60
Abstract model checking is the composition of abstract interpretations 62

4. Abstraction/concretization in semantics 63
Semantics are abstract interpretations . . . 64
Maximal trace semantics . . . 65
Transition semantics . . . 66
Natural, demoniac & angelic semantics . . . 67
Relational semantics . . . 68
Non-deterministic denotational semantics . . . 69

Predicate transformer semantics 70
Axiomatic semantics 71
The hierarchy of semantics 72

5. Conclusion **73**
What is abstract interpretation? 74

Electronic bibliography

