Analyse par intervalles et interprétation abstraite: une question de vocabulaire ?

Alexandre Chapoutot

ENSTA ParisTech

GT MEA – March 19, 2015

Interval Analysis and Abstract Interpretation

Starting point of this talk

Interval Analysis: **Ramon Moore** "Interval Analysis", 1960.

Abstract interpretation: **Patrick Cousot and Radhia Cousot** "Abstract Interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints", 1977.

Disclaimer

Goal of this talk: clarifying some vocabulary and concepts used in these two research fields.

It is not a comparison of them.

Interval analysis

. . . The title refers to a set of concepts and techniques based on treating an interval of real numbers as a new kind of numbers, represented by a pair of real numbers, namely, its left and right endpoints.
The techniques can be programmed for computers in order to obtain simultaneously upper and lower bounds to exact solutions of equations of various types. . .

R. Moore, in preface of Interval Analysis, 1966.

Interval numbers and arithmetic

Interval numbers

$$[a, b] = \{x \in \mathcal{R} \mid a \leq x \leq b\}$$

Simplified version of the interval arithmetic (Moore, p. 8-9)

Operation	Interval arithmetic
$[a, b] + [c, d]$	$[a + c, b + d]$
$[a, b] - [c, d]$	$[a - d, b - c]$
$[a, b] \times [c, d]$	$[\min(E), \max(E)]$ with $E = (ac, ad, bc, bd)$
$[a, b] \div [c, d]$	$\left[\frac{a}{d}, \frac{b}{c}\right]$ if $0 \neq [c, d]$

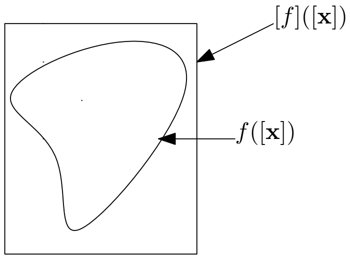
Theses operations are a computable version of sets operations

$$[a, b] \diamond [c, d] = \{x \diamond y \mid \forall x \in [a, b] \wedge y \in [c, d]\} \quad \text{with} \quad \diamond \in \{+, -, \times, \div\}$$

Inclusion function

An **inclusion function** $[f] : \mathcal{IR}^n \to \mathcal{IR}^m$ for a function $f : \mathcal{R}^n \to \mathcal{R}^m$ satisfies for all $[\mathbf{x}] \in \mathcal{IR}^n$,

$$\begin{aligned} f([\mathbf{x}]) &= \{f(\mathbf{x}) \mid \mathbf{x} \in [\mathbf{x}]\} \\ &\subseteq [f]([\mathbf{x}]) \end{aligned}$$



Properties of inclusion function

Interval arithmetic is **inclusion isotonic**, i.e.,

Theorem (Moore, p. 11)

If $[f]([x_1], \dots, [x_n])$ is a rational expression in the interval variables $[x_1]$, …, $[x_n]$, i.e., a finite combination of x_1, …, x_n and a finite set of constant intervals with interval arithmetic operations, then

$$[x_1'] \subset [x_1], \dots, [x_n'] \subset [x_n] \Rightarrow [f]([x_1'], \dots, [x_n']) \subset [f]([x_1], \dots, [x_n])$$

for every set of interval number $[x_1]$, …, $[x_n]$ for which the interval arithmetic operations in $[f]$ are defined.

Consequence: "…we can **bound the range** of a real rational function over intervals…"

If $f(x_1, \dots, x_n)$ is real rational expression in which each variable x_1, …, x_n occurs only once then

$$[f]([x_1], \dots, [x_n]) = \{f(x_1, \dots, x_n) \mid x_i \in [x_i], i = 1, \dots, n\} \ .$$

Problem solved

- $\mathcal{IR}^n$ can be endowed with a metric and an (inclusion) order
- inclusion function can be extended do deal with sin, cos, $\int$, etc.

Examples of considered problem in interval analysis

$$f(\mathbf{x}) \diamond \mathbf{y} \quad \text{with} \quad \diamond \in \{\leq, =, \geq, \text{etc.}\}$$

$$\mathbf{x} \diamond f^{-1}(\mathbf{y})$$

$$\text{minimize} \quad f(\mathbf{x}) \quad \text{subject to} \quad p(\mathbf{x}) < 0 \quad \text{and} \quad q(\mathbf{x}) = 0$$

$$\dot{\mathbf{x}} = f(\mathbf{x})$$

etc.

A generic algorithm

- A big box $[x]$ containing the solution
- Walk through $[x]$ to isolate solution

Usually combined with methods such as bisection, contraction or paving.

Abstract interpretation

A program denotes computations in some universe of objects. Abstract interpretation of programs consists in using that denotation to describe computations in another universe of abstract objects, so that the results of abstract execution give some information on the actual computations. . .

P. Cousot and R. Cousot, in abstract of POPL article, 1977.

Abstract interpretation

Abstract interpretation is **semantics-based program analysis**.

The main ingredients

- the standard meanings of a program.
- a simplified meaning (used to answer specific questions) of a program.
- a mathematical relations between these two meaning to guarantees correctness (or soundness).

The standard meaning is usually given by an **operational semantics**.

A simplified (or approximated) meaning is usually based on a set representation of the operational semantics, i.e. **collecting semantics**.

A mathematical approach (**ordered sets**) guarantee:

- the correction of the approximation w.r.t. the standard meaning (**Galois connections**).
- a computable approximation.

A small introductory example – 1

Example settings

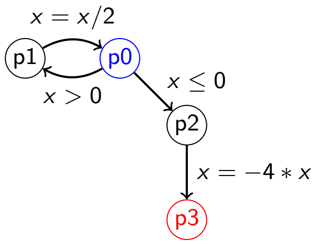
A small program for which we want to prove that at control-point p3 the values of variable x are always positive.

Source code with program points

```
p0:  while x > 0 do
p1:      x = x / 2
     done
p2:  x = -4 * x
p3:  exit
```

Note: p0 is the entry point and p3 is the exit point

Control flow graph



Control flow graph (CFG) defines a relation between the program points. It is used to describe an execution.

A small introductory example – 1

Example settings

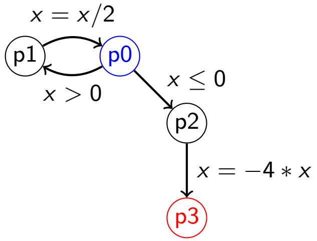
A small program for which we want to prove that at control-point p3 the values of variable x are always positive.

Source code with program points

```
p0:  while x > 0 do
p1:      x = x / 2
     done
p2:  x = -4 * x
p3:  exit
```

Note: p0 is the entry point and p3 is the exit point

Control flow graph


$$\begin{tikzcd}
p1 \arrow[r, bend left, "x = x/2"] & p0 \arrow[l, bend left, "x > 0"] \arrow[dr, "x \leq 0"] & \\
& & p2 \arrow[d, "x = -4 * x"] \\
& & p3
\end{tikzcd}$$

Trace of execution: At each program point is associated the values of the variables. For example,

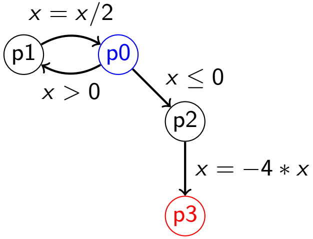
- At the entry point $x = 5$:
  $p0, 5 \rightarrow p1, 5 \rightarrow p0, 2 \rightarrow p1, 2 \rightarrow p0, 0 \rightarrow p2, 0 \rightarrow p3, 0$

A small introductory example – 2

Example settings

A small program for which we want to prove that at control-point p3 the values of variable x are always positive.

Control flow graph


$$\begin{tikzcd} p1 \arrow[r, bend left, "x = x/2"] & p0 \arrow[l, bend left, "x > 0"] \arrow[d, "x \leq 0"] \\ & p2 \arrow[d, "x = -4 * x"] \\ & p3 \end{tikzcd}$$

Note: $x \in \mathcal{Z}$ so possible infinite number of traces of execution.

As consequence,

- Software testing will never produces a 100% affirmative answer.

One solution

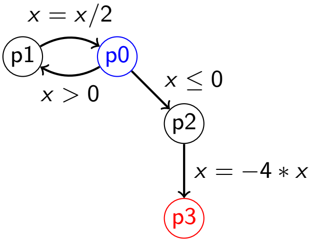
Manipulates **set of values** to reduce the number of executions we have to check to answer the question.

A small introductory example – 3 (Set of executions)

Example settings

A small program for which we want to prove that at control-point p3 the values of variable x are always positive.

Control flow graph



Note: each operation is applied on each value of the set.

For example, if $x \in \{3, 4\}$ we have:

$$\begin{tikzcd}
 & p0, \{3,4\} \arrow[r] & p1, \{3,4\} \arrow[r] & p0, \{1,2\} \arrow[d] & \\
 & & & p1, \{1,2\} \arrow[d] & \\
 & & & p0, \{0,1\} \arrow[dl] \arrow[dr] & \\
 & & p1, \{1\} \arrow[d] & & p2, \{0\} \arrow[d] \\
p3, \{0\} & p2, \{0\} \arrow[l] & p0, \{0\} \arrow[l] & & p3, \{0\}
\end{tikzcd}$$

Problem

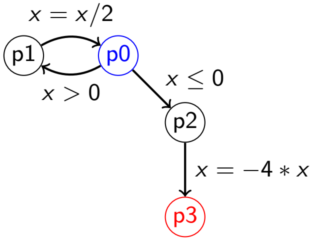
Set of values are usually not representable in computers.

A small introductory example – 4 (Abstract values)

Example settings

A small program for which we want to prove that at control-point `p3` the values of variable x are always positive.

Control flow graph



Remark: we do not need all the values to answer the question. Assume, we have "abstract" values to represent positive (`Pos`) and negative (`Neg`) values.

We need only two "abstract" executions to answer the question that is **abstract interpretation**.

Easy case: all values are negative

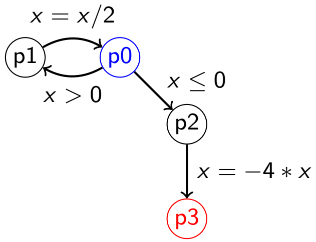
$p0, \text{Neg} \longrightarrow p2, \text{Neg} \longrightarrow p3, \text{Pos}$

A small introductory example – 4 (Abstract values)

Example settings

A small program for which we want to prove that at control-point `p3` the values of variable x are always positive.

Control flow graph



Remark: we do not need all the values to answer the question. Assume, we have "abstract" values to represent positive (`Pos`) and negative (`Neg`) values.

We need only two "abstract" executions to answer the question that is **abstract interpretation**.

Hard case: All values are positive (**non terminating execution**)

$$\begin{tikzcd}
& p1, \mathrm{Pos} \arrow[r] & p0, \mathrm{Pos} \arrow[r] & \cdots \\
p0, \mathrm{Pos} \arrow[ur] \arrow[r] & p2, \mathrm{Zero} \arrow[r] & p3, \mathrm{Zero} &
\end{tikzcd}$$

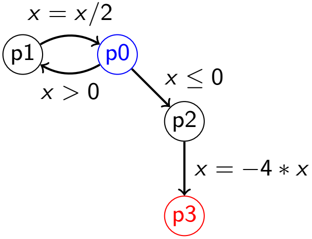
Solution State-based collecting semantics: gather all values encounter at each program points.

A small introductory example – 5 (Collecting semantics)

Example settings

A small program for which we want to prove that at control-point p3 the values of variable x are always positive.

Control flow graph



Idea: To avoid infinite abstract traces, we gather at each control point the set of all possible values met for all possible execution trace.

$$\begin{tikzcd} & p1, \text{Pos} \arrow[dl, bend right] & & \\ p0, \text{Pos} \arrow[ur, bend right] \arrow[r] & p2, \text{Zero} \arrow[r] & p3, \text{Zero} \end{tikzcd}$$

Collecting semantics

It is the solution of a **fixed-point of semantic equations** defined over an ordered set.

$$\mathcal{X}_0 = [\![x = x/2]\!]\,(\mathcal{X}_1) \cup \mathcal{X}_{\text{init}}$$
$$\mathcal{X}_1 = [\![x > 0]\!]\,(\mathcal{X}_0)$$
$$\mathcal{X}_2 = [\![x \leq 0]\!]\,(\mathcal{X}_0)$$
$$\mathcal{X}_3 = [\![x = -4 * x]\!]\,(\mathcal{X}_2)$$

Collecting semantics as fixpoint

We can associate a system of semantic equations to the collecting semantics of the form

$$\mathcal{X} = F(\mathcal{X}) \quad \text{with} \quad \mathcal{X} \in (\wp(\mathtt{Var} \to \mathcal{D}))^n$$

with n the number of program points.

Question

Does this system of equation have a solution?

Solution: YES if $\mathcal{X}$ is **a lattice** and F is **continuous**.
Algorithm based on Kleene method (combined with iteration strategies):

```
1: $\vec{\mathcal{X}}_0 := \vec{\emptyset}$
2: repeat
3:     $\vec{\mathcal{X}}_i := \vec{\mathcal{X}}_{i-1} \dot{\cup} F(\vec{\mathcal{X}}_{i-1})$
4: until $\vec{\mathcal{X}}_i \dot{\subseteq} \vec{\mathcal{X}}_{i-1}$
```

Lattice

Definition

A lattice is an ordered set P such that for all couple of elements of P:

- a least upper bound exists: $\forall x, y \in P, \quad x \sqcup y \in P$.
- a greatest lower bound exists: $\forall x, y \in P, \quad x \sqcap y \in P$.

Powerset

$$\begin{tikzcd}[arrows=-]
 & \{a,b,c\} \arrow[dl] \arrow[d] \arrow[dr] & \\
\{a,b\} \arrow[d] \arrow[dr] & \{a,c\} \arrow[dl] \arrow[dr] & \{b,c\} \arrow[dl] \arrow[d] \\
\{a\} \arrow[dr] & \{b\} \arrow[d] & \{c\} \arrow[dl] \\
 & \emptyset &
\end{tikzcd}$$

$$\{a, b\} \cup \{c\} = \{a, b, c\}$$

$$\{a, b\} \cap \{b, c\} = \{b\}$$

Functions between ordered sets

Monotony

Let $\langle P, \sqsubseteq_P \rangle$ and $\langle Q, \sqsubseteq_Q \rangle$ two Poset and $f : P \to Q$. The function f is monotone (or non-decreasing) iff

$$\forall x, y \in P, \quad x \sqsubseteq_P y \Rightarrow f(x) \sqsubseteq_Q f(y) \ .$$

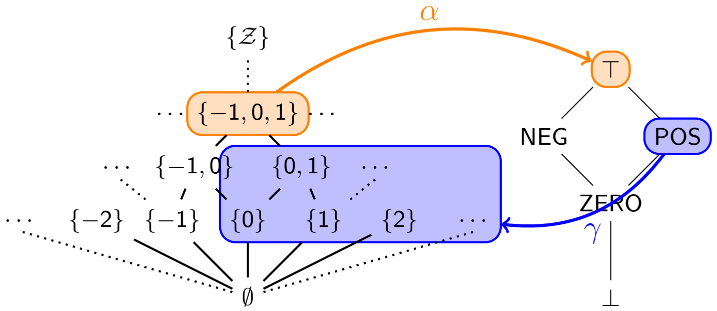
Continuity

Let $\langle P, \sqsubseteq_P \rangle$ and $\langle Q, \sqsubseteq_Q \rangle$ two Poset and $f : P \to Q$. The function f is continuous iff

$$\forall \text{ increasing chain } X \subset P, \quad \sqcup_Q f(X) = f(\sqcup_P X) \ .$$

Relation between concrete and abstract semantics

Abstract values are a symbolic representation of set of values.



$\{\mathcal{Z}\}$
$\cdots \{-1,0,1\} \cdots$
$\cdots \{-1,0\} \quad \{0,1\} \cdots$
$\cdots \{-2\} \quad \{-1\} \quad \{0\} \quad \{1\} \quad \{2\} \cdots$
$\emptyset$
α
$\top$
NEG
POS
ZERO
γ
$\bot$

These two words are connected each other with:

- Function γ: gives a meaning to abstract values.
- Function α: maps each set of values to an abstract value.

which form a **Galois connection**.

Galois connection

The abstraction and the concretization functions form a **Galois connection** between the concrete domain P and the abstract domain Q.

Definition

Let $\langle P, \sqsubseteq_P \rangle$ and $\langle Q, \sqsubseteq_Q \rangle$ two lattices. A Galois connection exists between P and Q iff there are two functions α and γ such that:

1. α is monotone.
2. γ is monotone.
3. $\forall q \in Q, \quad q \sqsubseteq_Q \alpha(\gamma(q))$.
4. $\forall p \in P, \quad \gamma(\alpha(p)) \sqsubseteq_P p$.

Remark: Conditions 3 and 4 express that we preserve safety but we may lose precision.

Notation

$$\langle P, \sqsubseteq_P \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle Q, \sqsubseteq_Q \rangle$$

Correctnees of abstraction

Abstract semantics is defined on the **abstraction** of transfer functions using the Galois connection (α, γ).

For a concrete transfer function $\mathbf{f}$, we consider $\mathbf{f}^\sharp$ as a sound abstract version iff

$$(\gamma \circ \mathbf{f}^\sharp)(v^\sharp) \sqsupseteq (\mathbf{f} \circ \gamma)(v^\sharp) \ .$$

Lemma of correctness

Let $\langle P, \sqsubseteq \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle P^\sharp, \sqsubseteq^\sharp \rangle$ a Galois connection. For all $p \in P$, for all $p^\sharp \in P^\sharp$ if $p \in \gamma(p^\sharp)$ then

$$\mathbf{f}(p) \sqsubseteq \gamma\left(\mathbf{f}^\sharp(p^\sharp)\right) \ .$$

Abstract interpretation and interval analysis

Question

Where are the inclusion functions in all that?
In the interpretation of **arithmetic expressions**

$$[\![\mathtt{var} = \mathtt{exp}]\!] = \begin{cases} (\mathtt{Var} \to \mathcal{R}) & \to (\mathtt{Var} \to \mathcal{R}) \\ \sigma & \mapsto \sigma\,[\mathtt{var} \leftarrow [\![\mathtt{exp}]\!](\sigma)] \end{cases}$$

Question

Where are the set theoretic operations in all that?
In the **interpretation of conditionals**
and in the **unions of paths** in control flow graph.

$$[\![x < 0]\!]^\sharp = \begin{cases} (\mathtt{Var} \to \mathtt{Sign}) & \to (\mathtt{Var} \to \mathtt{Sign}) \\ \sigma^\sharp & \mapsto \sigma[\mathtt{var} \leftarrow \mathtt{s}] \text{ s.t. } \mathtt{s} = \sigma^\sharp(x) \cap^\sharp \mathtt{Neg} \end{cases}$$

Conlusion

Both interval analysis and abstract interepretation talk about set of values but

- the mathematical objects considered are not exactly the same.
- usually the goal of the computations is not the same;

Locally, a combination of methods and techniques is possible e.g., Ruher et al.