An Introduction to Abstract Interpretation (with a Look at Abstract Fixpoint Checking)

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

SARA'2000, Austin, TX July 28th, 2000

An Informal Introduction to Abstract Interpretation

Organization of the talk

- 45 mn: informal introduction to abstract interpretation;
- 10 mn: informal sketch of the proceedings paper content (*"Partial completeness of abstract fixpoint checking"* [1]);
- 5 mn: left for questions.

[1] May be of interest to specialists only!

The initial application: program analysis

- Prove automatically that:
  for all programs P of a given programming language $\mathcal{L}$:
  for all possible executions of that program P in any conceivable environment:
  a given specification S is always satisfied.
- Initially the considered specifications S were simple safety specifications (e.g. absence of runtime errors).

The methodology [CC-POPL'77]

- Define formally the program executions by a fixpoint semantics of the programs of the language $\mathcal{L}$;
- Since the semantics of a program is **not** computable, use a manually designed approximation/abstraction of that semantics to check the specification.

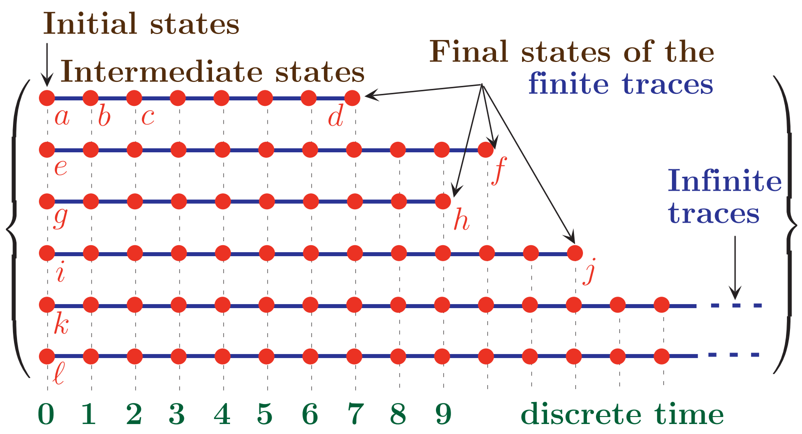
Reference

[CC-POPL'77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conf. Record of the 4th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages POPL'77*, Los Angeles, CA, 1977. ACM Press, pp. 238–252.

Semantics: intuition

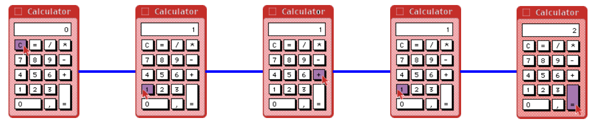
- The semantics of a program provides a formal mathematical model of all possible behaviors of a computer system executing this program (interacting with any possible environment);
- The semantics of a language defines the semantics of any program written in this language.

Example 1: trace semantics [4, 6]

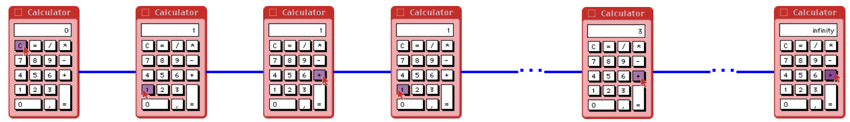


Examples of computation traces

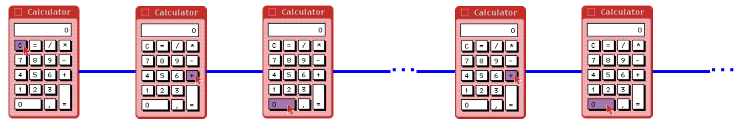
- Finite (C1+1=):



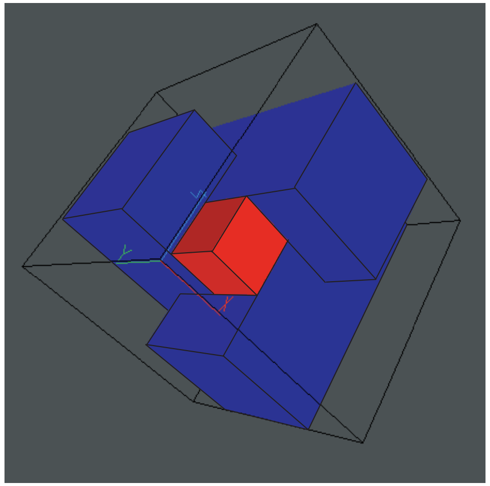
- Erroneous  (C1+1+1+1...):



- Infinite (C+0+0+0...):



Example 2: geometric semantics [18] (deadlock)

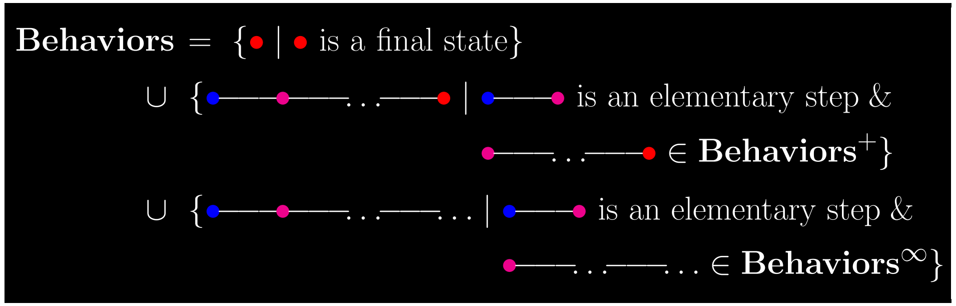


$[\![$ Pa.Pb.Va.Vb
$\|$ Pb.Pc.Vb.Vc
$\|$ Pc.Pa.Vc.Va $]\!]$

inaccessible

deadlock

Least Fixpoints: intuition [4, 6]



In general, the equation has multiple solutions. Choose the least one for the partial ordering:

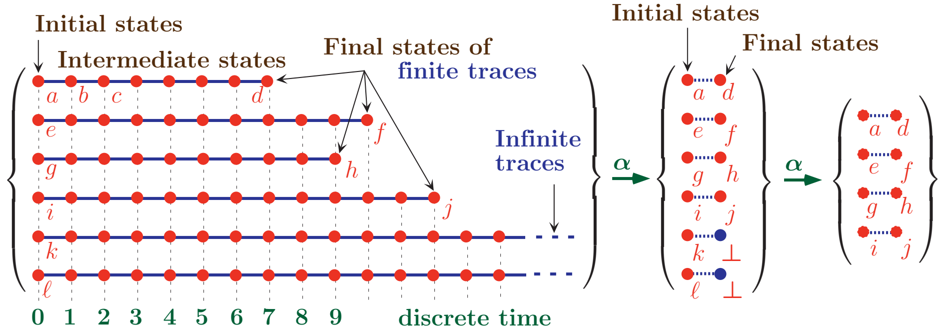
« *more finite traces & less infinite traces* ».

Abstraction: intuition

- Abstract interpretation is a theory of the approximation of the behavior of discrete systems, including the semantics of (programming or specification) languages [8, 9, 2];

- Abstract interpretation formalizes the intuitive idea that a semantics is more or less precise according to the considered observation level.

Example 1 of abstraction [2]



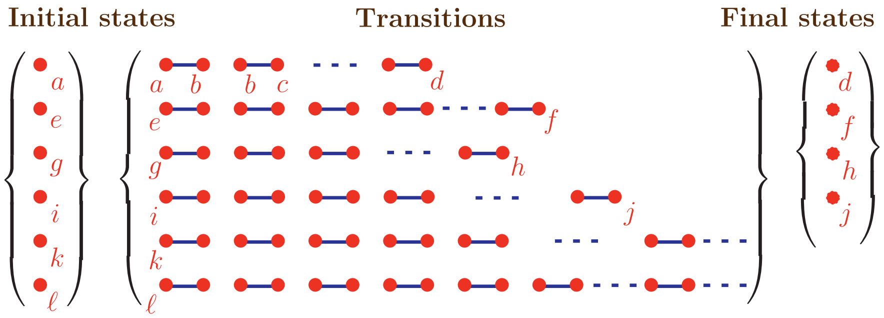
Trace semantics

Denotational semantics

Natural semantics

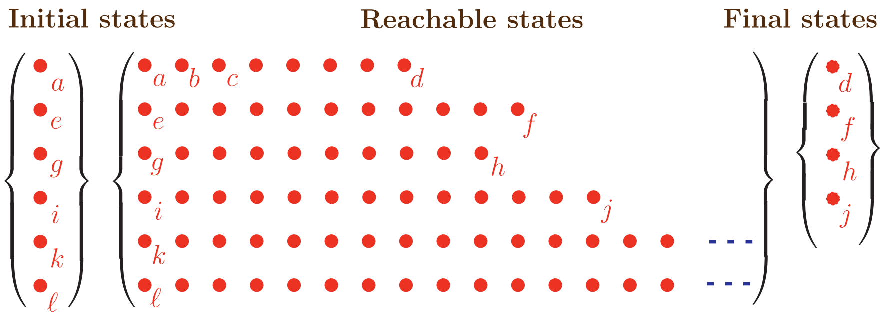
[2] P. Cousot. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation.* To appear in TCS (2000).

Example 2 of abstraction [3]



(Small-Step) Operational Semantics

Example 3 of abstraction [4]



Collecting Semantics

[3] P. Cousot. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation*. To appear in TCS (2000).

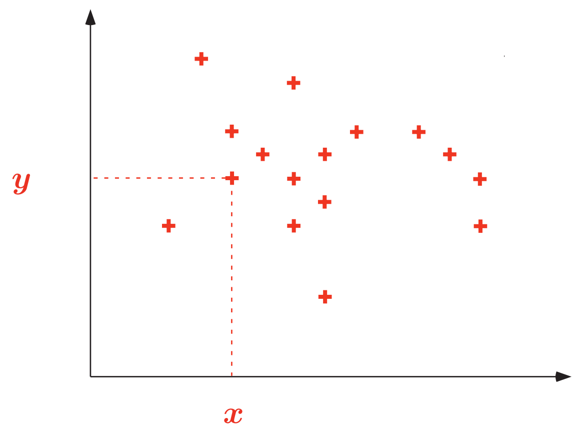
[4] P. Cousot. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation*. To appear in TCS (2000).

Computable abstractions

- If the approximation is rough enough, the abstraction of a semantics can lead to a version which is less precise but is effectively computable by a computer;

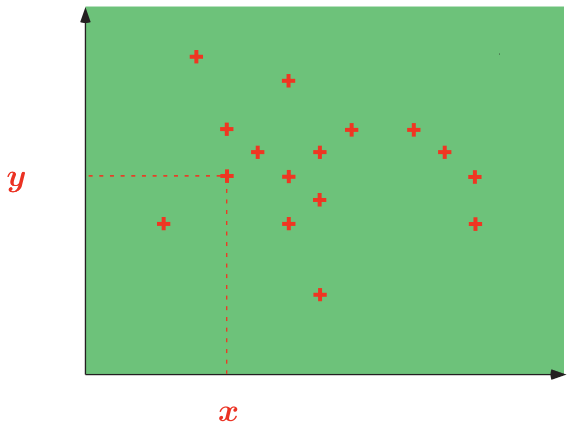
- By effective computation of the abstract semantics, the computer is able to analyze the behavior of programs and of software before and without executing them [7].

Computable abstractions of an [in]finite set of points;



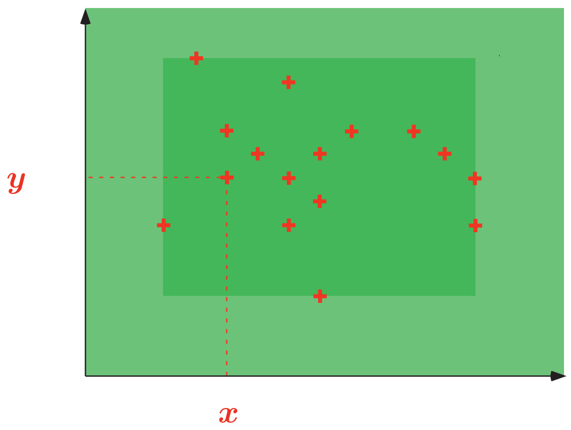
$$\{\ldots, \langle 19,\ 88\rangle, \ldots,\ \langle 19,\ 99\rangle, \ldots\}$$

Computable abstractions of an [in]finite set of points; Example 1: signs [9]



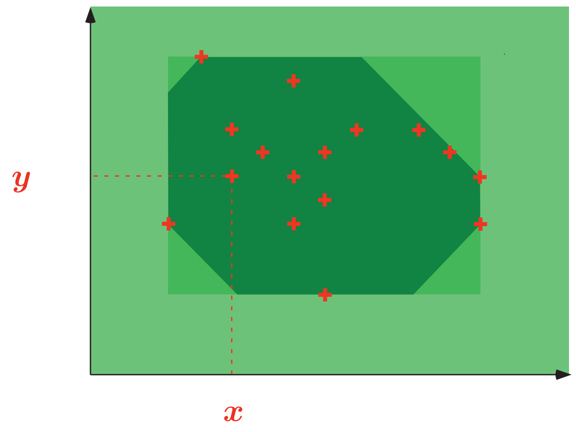
$$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 2: intervals [7, 8]



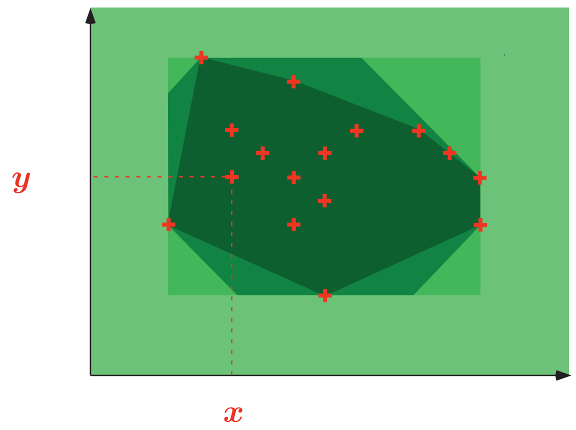
$$\begin{cases} x \in [19,\ 88] \\ y \in [19,\ 99] \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 3: octagons



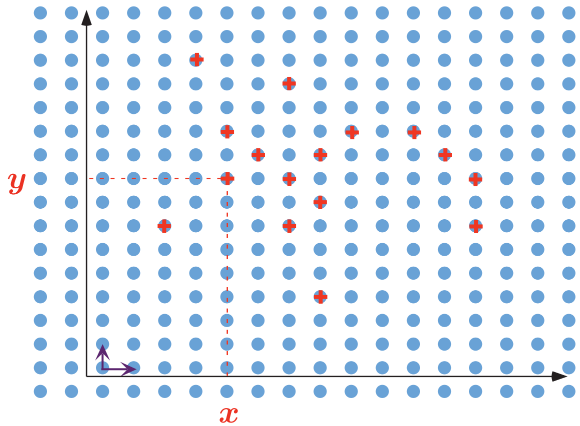
$$\begin{cases} 1 \leq x \leq 9 \\ x + y \leq 88 \\ 1 \leq y \leq 9 \\ x - y \leq 99 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 4: polyhedra [16]



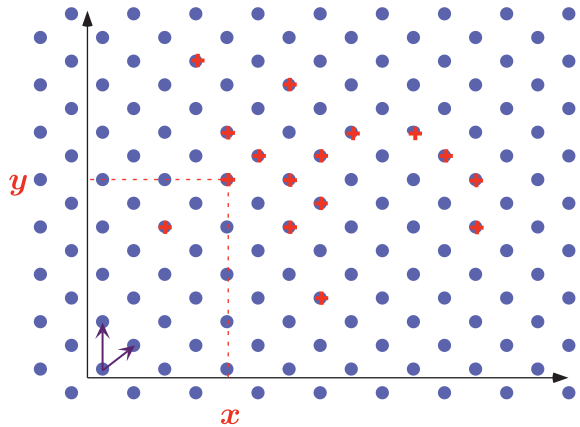
$$\begin{cases} 19x + 88y \leq 2000 \\ 19x + 99y \geq 0 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 5: simple congruences [19]



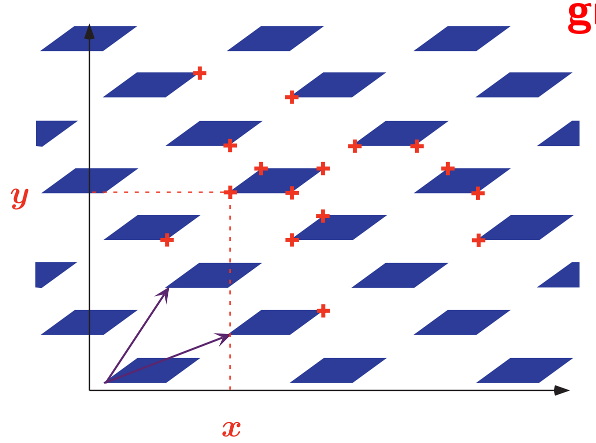
$$\begin{cases} x = 19 \bmod 88 \\ y = 19 \bmod 99 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 6: linear congruences [20]



$$\begin{cases} 1x + 9y = 8 \bmod 8 \\ 1x - 9y = 9 \bmod 9 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 7: trapezoidal linear congruences [22, 23]



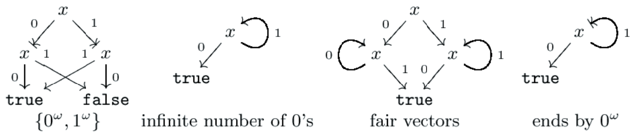
$$\begin{cases} 1x + 9y \in [0, 88] \bmod 10 \\ 1x - 9y \in [0, 99] \bmod 11 \end{cases}$$

Computable abstractions of symbolic structures

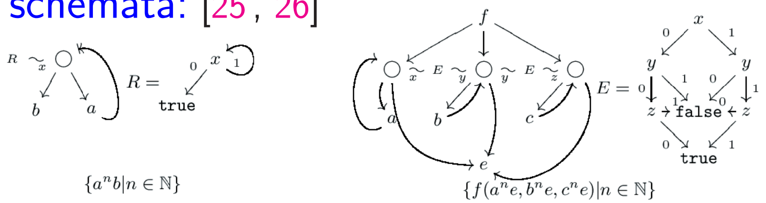
- Most structures manipulated by programs are *symbolic structures* such as control structures (call graphs), data structures (search trees), communication structures (distributed & mobile programs), etc;
- It is very difficult to find compact and expressive abstractions of such sets of objects (languages, automata, trees, graphs, etc.).

Example of abstractions of infinite sets of infinite trees

Binary Decision Graphs: [24]



Tree schemata: [25, 26]



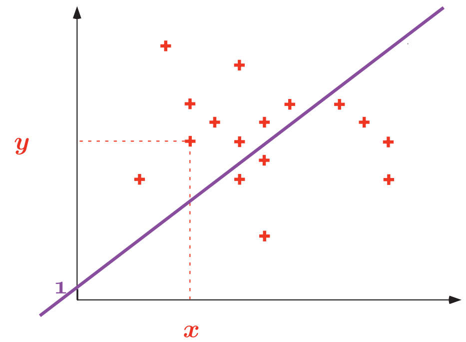
Note that E is the equality relation.

Information loss

- All answers given by the abstract semantics are always correct with respect to the concrete semantics;
- Because of the information loss, not all questions can be definitely answered with the abstract semantics;
- The more concrete semantics can answer more questions;
- The more abstract semantics are more simple.

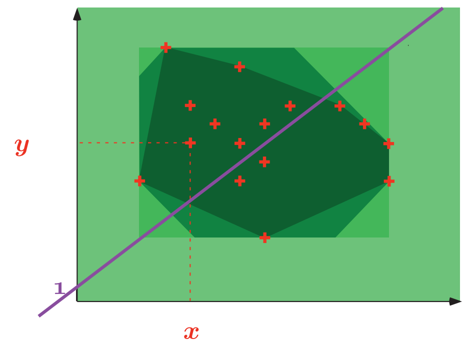
Example of information loss

- Is the operation `1/(x+1-y)` well defined at run-time?
- Concrete semantics: **yes**



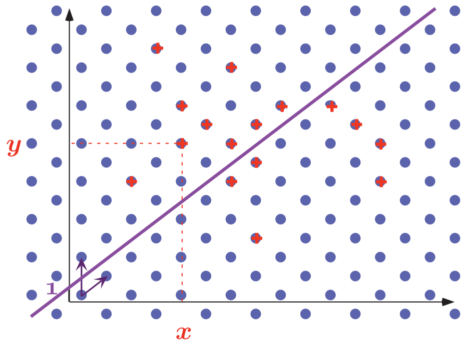
Example of information loss

- Is the operation `1/(x+1-y)` well defined at run-time?
- Abstract semantics 1: **I don't know**



Example of information loss

- Is the operation `1/(x+1-y)` well defined at run-time?
- Abstract semantics 2: **yes**



Program Static Analysis

Objective of program static analysis

- Programming bugs should be eradicated before they lead to disastrous catastrophes!
- Full automation is necessarily limited (undecidability);
- Program static analysis uses *abstract interpretation* to derive, from a standard semantics, an approximate and computable semantics. This derivation is itself not (fully) mechanizable;
- It follows that the computer is able to analyze the behavior of software before and without executing it;
- This is essential for computer-based safety-critical systems (for example: planes, trains, launchers, nuclear plants, etc.).

Example: interval analysis (1975) [5]

Program to be analyzed:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Equations (abstract interpretation of the semantics):

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

The analyzer reads the program text and produces (a representation of) the above equations and then solve them iteratively. The equations are an abstraction of the trace semantics of the program. The formal derivation of the algorithm producing the equation by abstract interpretation of the program trace semantics is (mainly) manual.

Example: interval analysis (1975) [5]

Increasing chaotic iteration, initialization:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = \emptyset \\ X_2 = \emptyset \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Increasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = \emptyset \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Increasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,1] \\ X_3 = \emptyset \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Increasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
      x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,1] \\ X_3 = [2,2] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Increasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
      x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,2] \\ X_3 = [2,2] \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence?**

```
   x := 1;
1:
   while x < 10000 do
2:
      x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,2] \\ X_3 = [2,3] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence??**

```
   x := 1;
1:
   while x < 10000 do
2:
      x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,3] \\ X_3 = [2,3] \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence???**

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,3] \\ X_3 = [2,4] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence????**

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,4] \\ X_3 = [2,4] \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.
[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence?????**

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,4] \\ X_3 = [2,5] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence??????**

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1,5] \\ X_3 = [2,5] \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.
[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Increasing chaotic iteration: **convergence???????**

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, 5] \\ X_3 = [2, 6] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Convergence speed-up by extrapolation:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, +\infty] \quad \Leftarrow \text{widening} \\ X_3 = [2, 6] \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.
[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Decreasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, +\infty] \\ X_3 = [2, +\infty] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Decreasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1, 1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1, 1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1, 1] \\ X_2 = [1, 9999] \\ X_3 = [2, +\infty] \\ X_4 = \emptyset \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.
[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Decreasing chaotic iteration:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1, 9999] \\ X_3 = [2, 10000] \\ X_4 = \emptyset \end{cases}$$

Example: interval analysis (1975) [5]

Final solution:

```
   x := 1;
1:
   while x < 10000 do
2:
       x := x + 1
3:
   od;
4:
```

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1, 9999] \\ X_3 = [2, 10000] \\ X_4 = [10000, 10000] \end{cases}$$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.
[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

Example: interval analysis (1975) [5]

Result of the interval analysis:

`   x := 1;`
`1:` $\{x = 1\}$
`   while x < 10000 do`
`2:` $\{x \in [1, 9999]\}$
`       x := x + 1`
`3:` $\{x \in [2, 10000]\}$
`   od;`
`4:` $\{x = 10000\}$

$$\begin{cases} X_1 = [1,1] \\ X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\ X_3 = X_2 \oplus [1,1] \\ X_4 = (X_1 \cup X_3) \cap [10000, +\infty] \end{cases}$$

$$\begin{cases} X_1 = [1,1] \\ X_2 = [1, 9999] \\ X_3 = [2, 10000] \\ X_4 = [10000, 10000] \end{cases}$$

Example: interval analysis (1975) [5]

Exploitation of the result of the interval analysis:

`   x := 1;`
`1:` $\{x = 1\}$
`   while x < 10000 do`
`2:` $\{x \in [1, 9999]\}$
`       x := x + 1` $\longleftarrow$ **no overflow**
`3:` $\{x \in [2, 10000]\}$
`   od;`
`4:` $\{x = 10000\}$

[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.
[5] P. Cousot & R. Cousot, ISOP'1976, POPL'77.

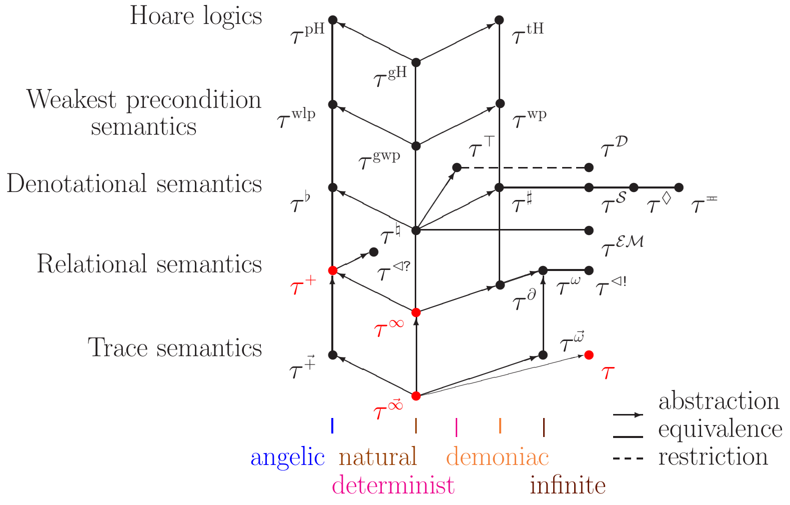
Some applications of static analysis by abstract interpretation

- data flow analysis for program optimization & transformation (including partial evaluation) [9, 15];
- type inference (including undecidable systems)/soft typing [5];
- abstract model-checking of infinite systems [14, 15];
- abstract debugging & testing [2, 1];
- probabilistic analysis [28];
- ...

Some other recent applications of abstract interpretation

- Fundamental applications:
  - design of hierarchies of semantics [13, 3, 6], ...;
- Practical applications:
  - communication topology of mobile/distributed code [29];
  - automatic differentiation of numerical programs;
  - security (analysis of cryptographic protocols [27], mobile code [17]);
  - semantic tattooing/watermarking of software, ...

Lattice of semantics [6]



Forthcoming research

A lot of fundamental research remains to be one:

- modularity,
- higher order functions & modules,
- floating point numbers,
- probabilistic analyses,
- liveness properties with fairness,
- ...;

An impressive application (1996/97)

- Abstract interpretation is used (including interval analysis) for the static analysis of the embedded ADA software of the Ariane 5 launcher [6]; [21]
- Automatic detection of the definiteness, potentiality, impossibility or inaccessibility of run-time errors [7];
- Automatic discovery of the 501 flight error;
- Success for the 502 & 503 flights and the ARD [8].

Industrialization of static analysis by abstract interpretation

- Connected Components Corporation (U.S.A.), L. Harrison, 1993;
- AbsInt Angewandte Informatik GmbH (Germany), R. Wilhelm, 1998;
- PolySpace Polyspace Technologies (France), A. Deutsch & D. Pilaud, 1999.

[5] Flight software (60,000 lines of Ada code) and Inertial Measurement Unit (30,000 lines of Ada code).
[6] such as scalar and floating-point overflows, array index errors, divisions by zero and related arithmetic exceptions, uninitialized variables, data races on shared data structures, etc.
[7] Atmospheric Reentry Demonstrator: module coming back to earth.

An Few Elements of Abstract Interpretation Theory

Seminal reference

– P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conf. Record of the 6th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages*, San Antonio, TX, 1979. ACM Press, pp. 269–282.

See also an introduction in [11] and variants in [12].

Concrete properties

- The semantic definition S of a program P associates a semantics $S[\![P]\!] \in D$ to the program describing its possible executions (e.g. a set of traces);
- A property is represented by the set of semantics which have this property;
- The set of properties form a complete boolean lattice:

$$\langle \wp(D),\ \subseteq,\ \emptyset,\ D,\ \cup,\ \cap,\ \neg \rangle \tag{1}$$

Example: state properties of transition systems

- Σ is a set of states;
- $t \in \wp(\Sigma \times \Sigma)$ is the transition relation between a state and its possible successors;
- $\wp(\Sigma)$ is the set of state properties;
- Example:

  $I \subseteq \Sigma$ is the set of initial states.

Concrete fixpoint semantics

- The concrete semantics $S[\![P]\!]$ of a given program P is defined in fixpoint form:

$$S[\![P]\!] \stackrel{\triangle}{=} \mathit{lfp}^{\subseteq} F \qquad (2)$$

- The semantic transformer F is monotonic:

$$F \in \wp(D) \stackrel{\text{mon}}{\longmapsto} \wp(D) \qquad (3)$$

- In general the semantic transformer of a program P is defined by structural induction on the syntax of P.

Example: reachability analysis

- The reachable states of a transition system $\langle \Sigma, t, I\rangle$ is

$$R \stackrel{\triangle}{=} \mathit{post}[t^\star](I) \qquad (4)$$

  where:
  - $t^\star$ is the reflexive transitive closure of the transition relation t,
  - $\mathit{post}[r](X) = \{y \mid \exists x \in X : \langle x, y\rangle \in r\}$ is the right image of the set X by relation r;
- In fixpoint form:

$$R = \mathit{lfp}^{\subseteq} F \quad \text{where} \quad F(X) = I \cup \mathit{post}[t](X)\ . \qquad (5)$$

Abstract properties

- The abstract properties form a complete boolean lattice:

$$\langle L, \sqsubseteq, \bot, \top, \sqcup, \sqcap\rangle \qquad (6)$$

Galois connection between concrete and abstract properties

- The correspondence between concrete and abstract properties is given by a Galois connection:

$$\langle \wp(D),\ \subseteq\rangle \xleftarrow[\alpha]{\gamma} \langle L,\ \sqsubseteq\rangle \qquad (7)$$

- $\alpha(P)$ is the abstraction of the concrete property P;
- $\gamma(Q)$ is the concretization of the abstract property Q;

- $\forall P, Q : \alpha(P) \sqsubseteq Q \iff P \subseteq \gamma(Q)$.

Example 1: state abstraction

- If
  - $h \in D \mapsto \bar{D}$,
  - $\alpha(P) \triangleq \{\ h(x) \mid x \in P\}$,
  - $\gamma(Q) \triangleq \{\ x \mid h(x) \in Q\}$.

  then:

$$\langle \wp(D),\ \subseteq\rangle \xleftarrow[\alpha]{\gamma} \langle \wp(\bar{D}),\ \subseteq\rangle \qquad (8)$$

- Not all abstractions are of that form, a counter-example .../...

Example 2: intervals

- Concrete properties: $\wp(\mathbb{Z})$ (sets of integers);
- Abstract properties: $[a, b]$ ($a \leq b$, intervals),
  $\perp$ empty interval;

- Abstraction: $\alpha(\emptyset) \triangleq \perp$,
  $\alpha(P) \triangleq [\min P, \max P]$, $P \neq \emptyset$;

- Concretization: $\gamma(\perp) \triangleq \emptyset$,
  $\gamma([a, b]) \triangleq \{x \in \mathbb{Z} \mid a \leq x \leq b\}$.

Intuition behind Galois connections

- $\alpha(P)$ is the best possible approximation of P in the abstract domain:
  - $P \subseteq \gamma \circ \alpha(P)$, it's an upper approximation
  - $P \subseteq \gamma(Q) \Longrightarrow \gamma \circ \alpha(P) \subseteq \gamma(Q)$,
    it's the best upper approximation

- logical implication is preserved by the abstraction:
  - $P \subseteq P' \Longrightarrow \alpha(P) \sqsubseteq \alpha(P')$,
  - $Q \sqsubseteq Q' \Longrightarrow \gamma(Q) \subseteq \gamma(Q')$.

Composing abstractions

- The composition of two Galois connections:

$$\langle \wp(D),\ \subseteq\rangle \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} \langle L_1,\ \sqsubseteq_1\rangle$$

$$\langle L_1,\ \sqsubseteq_1\rangle \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} \langle L_2,\ \sqsubseteq_2\rangle$$

is a Galois connection:

$$\langle \wp(D),\ \subseteq\rangle \underset{\alpha_2 \circ \alpha_1}{\overset{\gamma_1 \circ \gamma_2}{\leftrightarrows}} \langle L_2,\ \sqsubseteq_2\rangle \tag{9}$$

Approximating functions

- If:

$$\langle \wp(D),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle \tag{10}$$

then:

$$\langle \wp(D) \xrightarrow{\text{mon}} \wp(D),\ \dot{\subseteq}\rangle \underset{\dot{\alpha}}{\overset{\dot{\gamma}}{\leftrightarrows}} \langle L \xrightarrow{\text{mon}} L,\ \dot{\sqsubseteq}\rangle \tag{11}$$

where:

$$f \mathrel{\dot{\preceq}} g \iff \forall x : f(x) \preceq g(x)\,,$$
$$\dot{\alpha}(F) \triangleq \alpha \circ F \circ \gamma\,,$$
$$\dot{\gamma}(G) \triangleq \gamma \circ G \circ \alpha.$$

Approximating fixpoints

- If:

$$\langle \wp(D),\ \subseteq\rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle L,\ \sqsubseteq\rangle \tag{12}$$

then:

$$\mathsf{lfp}^{\sqsubseteq} F \sqsubseteq \gamma\ (\mathsf{lfp}^{\sqsubseteq} \dot{\alpha}(F)) \tag{13}$$

- So $\mathsf{lfp}^{\dot{\sqsubseteq}} \dot{\alpha}(F) \sqsubseteq S$ implies $\mathsf{lfp}^{\subseteq} F \subseteq \gamma(S)$.

Computing fixpoints

- The transfinite iteration sequence:
  - $X^0 \triangleq \bot$,
  - $X^{\delta+1} \triangleq \bar{F}\ (X^{\delta})$ for successor ordinals $\delta + 1$,
  - $X^{\lambda} \triangleq \bigsqcup_{\delta<\lambda} X^{\delta}$ for limit ordinals λ

converges to $\mathsf{lfp}^{\sqsubseteq} \bar{F}$:

$$\exists \epsilon : \forall \delta \geq \epsilon : X^{\delta} = \mathsf{lfp}^{\sqsubseteq} \bar{F} \tag{14}$$

Fixpoint checking algorithm

- Check that $\mathit{lfp}^{\sqsubseteq}\ \bar{F} \sqsubseteq S$ where $\bar{F} \triangleq \dot{\alpha}(F)$:

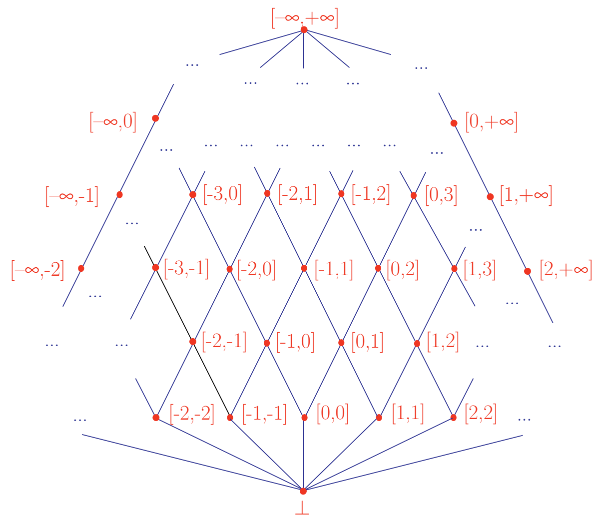
Algorithm 1

$X := \bot;$
repeat
  $X' := \bar{F}(X);$
  $\mathit{Stop} := (X = X') \vee (X' \not\sqsubseteq S);$
  $X := X';$
until $\mathit{Stop};$
return $(X \sqsubseteq S);$

Convergence

- The iterates $X^\delta, \delta \in \mathbb{N}$ form an increasing chain;
- The algorithm 1 terminates if the abstract lattice L satisfies the ascending chain condition (or is finite).

Example: lattice of intervals



Speeding up convergence

- In case of possible divergence use a widening [8, 10]:

Algorithm 2

$X := \bot;$ *(See more precise algorithm in [9])*
repeat
  $X' := X \,\nabla\, \bar{F}(X);$
  $\mathit{Stop} := (X = X') \vee (X' \not\sqsubseteq S);$
  $X := X';$
until $\mathit{Stop};$
return $(X \sqsubseteq S);$

- Example: $[1, 1] \,\nabla\, [1, 2] = [1, +\infty]$.

Pointers to references

Starter:

P. Cousot. Abstract interpretation. *ACM Computing Surveys* 28 (2), 1996, 324–328.

On the web:

http://www.di.ens.fr/~cousot/

SARA'2000 paper content sketch:

"Partial Completeness of Abstract Fixpoint Checking"

Approaches to program verification

Deductive methods: The proof size is exponential in the program size!

Model-checking: Restricted to finite models. Gained only a factor of 100 in 10 years. The limit seems to be reached!

Program static analysis: Can analyze large programs (220 000 lines of C) but specifications are simple and the abstraction is manual!

Can abstract interpretation be automatized?

Abstraction for finite fixpoint checking

- A finite abstraction to prove a given class of specifications (such as safety specifications) does not exist for a given programming language;
- However, such a finite abstraction always exists to prove a given specification for a given program;
- This SARA'2000 paper characterizes all such finite abstractions for a given program and specification.

Reference

- P. Cousot and R. Cousot. Comparing the Galois connection and widening/narrowing approaches to abstract interpretation. In M. Bruynooghe and M. Wirsing, editors, *Proc. Int. Workshop Programming Language Implementation and Logic Programming, PLILP '92*, Leuven, Belgium, 13–17 Aug. 1992, LNCS 631, p. 269–295. Springer, 1992.

The problem: abstract fixpoint checking

- The fixpoint checking problem for $\langle F,\ I,\ S\rangle$:

$$\mathit{lfp}^{\leq}\ \boldsymbol{\lambda} X \cdot I \vee F(X) \leq S\ ?$$

- Definition: $A \in L$ an *invariant* for $\langle F,\ I,\ S\rangle$ if and only if $I \leq A \;\&\; F(A) \leq A \;\&\; A \leq S$;

- In practice, $\langle F,\ I,\ \gamma(S)\rangle$ has to be checked in the abstract:

$$\mathit{lfp}^{\sqsubseteq}\ \boldsymbol{\lambda} X \cdot \alpha(I \vee F(\gamma(X))) \sqsubseteq S\ ?$$

Definition: partially complete fixpoint checking

- An abstract safety specification checking algorithm for checking a specification $\langle F,\ I,\ \gamma(S)\rangle$ is partially complete if and only if it misses no positive answer, up to termination;

The two main results in the SARA'2000 paper

1. The various (abstract) safety specification checking algorithms, whether forward (as used in program static analysis) or backward (as used in model-checking) are all equivalent (provide the same answers), up to termination;
2. A safety specification checking algorithm is partially complete if if and only if the abstract domain contains the image of an invariant.

Intuition: the design of the abstraction is as difficult as the design of an invariant.

References

[1] F. Bourdoncle. Abstract debugging of higher-order imperative languages. In *Proceedings of the ACM-SIGPLAN Conference on Programming Language Design and Implementation*, pages 46–55. ACM Press, New York, New York, United States, 1993.

[2] 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, Inc., Englewood Cliffs, New Jersey, United States, 1981.

[3] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Electronic Notes in Theoretical Computer Science*, 6, 1997. URL: http://www.elsevier.nl/locate/entcs/volume6.html, 25 pages.

[4] P. Cousot. Design of semantics by abstract interpretation, invited address. In *Mathematical Foundations of Programming Semantics, Thirteenth Annual Conference (MFPS XIII)*, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States, 23–26 March 1997.

[5] P. Cousot. Types as abstract interpretations, invited paper. In *Conference Record of the Twentyfourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 316–331, Paris, France, January 1997. ACM Press, New York, New York, United States.

[6] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation. *Theoretical Computer Science*, To appear (Preliminary version in [3]).

[7] P. Cousot and R. Cousot. Static determination of dynamic properties of programs. In *Proceedings of the Second International Symposium on Programming*, pages 106–130. Dunod, Paris, France, 1976.

[8] P. Cousot and R. 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 Press, New York, New York, United States.

[9] P. Cousot and R. 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, New York, United States.

[10] P. Cousot and R. Cousot. Comparison of the Galois connection and widening/narrowing approaches to abstract interpretation. JTASPEFL '91, Bordeaux. *BIGRE*, 74:107–110, October 1991.

[11] P. Cousot and R. Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Programming* [9], 13(2–3):103–179, 1992.

[12] P. Cousot and R. Cousot. Abstract interpretation frameworks. *Journal of Logic and Computation*, 2(4):511–547, August 1992.

[13] P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *Conference Record of the Ninthteenth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 83–94, Albuquerque, New Mexico, 1992. ACM Press, New York, New York, United States.

[14] P. Cousot and R. Cousot. Refining model checking by abstract interpretation. *Automated Software Engineering*, 6:69–95, 1999.

[15] P. Cousot and R. Cousot. Temporal abstract interpretation. In *Conference Record of the Twentyseventh Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 12–25, Boston, Massachusetts, January 2000. ACM Press, New York, New York, United States.

[16] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 84–97, Tucson, Arizona, 1978. ACM Press, New York, New York, United States.

[8] The editor of Journal of Logic Programming has mistakenly published the unreadable galley proof. For a correct version of this paper, see http://www.di.ens.fr/~cousot.

[17] J. Feret. Confidentiality analysis for mobiles systems. In J. Palsberg, editor, *Proceedings of the Seventh International Symposium on Static Analysis, SAS '2000*, Santa Barbara, California, United States, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 29 June – 1 July 2000.

[18] É. Goubault. Schedulers as abstract interpretations of higher-dimensional automata. In *Proceedings of the ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation, PEPM '95*, La Jolla, California, pages 134–145. ACM Press, New York, New York, United States, 21–23 June 1995.

[19] P. Granger. Static analysis of arithmetical congruences. *Int. J. Comput. Math.*, 30:165–190, 1989.

[20] P. Granger. Static analysis of linear congruence equalities among variables of a program. In S. Abramsky and T.S.E. Maibaum, editors, *Proceedings of the International Joint Conference on Theory and Practice of Software Development, TAPSOFT '91, Volume 1 (CAAP '91)*, Brighton, Great Britain, Lecture Notes in Computer Science 493, pages 169–192. Springer-Verlag, Berlin, Germany, 1991.

[21] P. Lacan, J.N. Monfort, L.V.Q. Ribal, A. Deutsch, and G. Gonthier. The software reliability verification process: The Ariane 5 example. In *Proceedings DASIA 98 – DAta Systems In Aerospace*, Athens, Greece. ESA Publications, SP-422, 25–28 May 1998.

[22] F. Masdupuy. Array operations abstraction using semantic analysis of trapezoid congruences. In *Proceedings of the ACM International Conference on Supercomputing, ICS '92*, pages 226–235, Washington D.C., July 1992.

[23] F. Masdupuy. Semantic analysis of interval congruences. In D. Bjørner, M. Broy, and I.V. Pottosin, editors, *Proceedings of the International Conference on Formal Methods in Programming and their Applications*, Academgorodok, Novosibirsk, Russia, Lecture Notes in Computer Science 735, pages 142–155. Springer-Verlag, Berlin, Germany, 28 June – 2 July 1993.

[24] L. Mauborgne. Binary decision graphs. In A. Cortesi and G. Filé, editors, *Proceedings of the Sixth International Symposium on Static Analysis, SAS '99*, Venice, Italy, 22–24 september 1999, Lecture Notes in Computer Science 1694, pages 101–116. Springer-Verlag, Berlin, Germany, 1999.

[25] L. Mauborgne. Improving the representation of infinite trees to deal with sets of trees. In G. Smolka, editor, *Programming Languages and Systems, Proceedings of the Ninth European Symposium on Programming, ESOP '2000*, Berlin, Germany, Lecture Notes in Computer Science 1782, pages 275–289. Springer-Verlag, Berlin, Germany, March – April 2000.

[26] L. Mauborgne. In J. Palsberg, editor, *Proceedings of the Seventh International Symposium on Static Analysis, SAS '2000*, Santa Barbara, California, United States, Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 29 June – 1 July 2000.

[27] D. Monniaux. Abstracting cryptographic protocols with tree automata. In A. Cortesi and G. Filé, editors, *Proceedings of the Sixth International Symposium on Static Analysis, SAS '99*, Venice, Italy, 22–24 september 1999, Lecture Notes in Computer Science 1694, pages 149–163. Springer-Verlag, Berlin, Germany, 1999.

[28] D. Monniaux. Abstract interpretation of probabilistic semantics. In J. Palsberg, editor, *Proceedings of the Seventh International Symposium on Static Analysis, SAS '2000*, Santa Barbara, California, United States, Lecture Notes in Computer Science 1824, pages 322–339. SPRINGER, 29 June – 1 July 2000.

[29] A. Venet. Automatic determination of communication topologies in mobile systems. In G. Levi, editor, *Proceedings of the Fifth International Symposium on Static Analysis, SAS '98*, Pisa, Italy, 14–16 september 1998, Lecture Notes in Computer Science 1503, pages 152–167. Springer-Verlag, Berlin, Germany, 1998.