Abstract Interpretation: Achievements and Perspectives

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

SSGRR-2000, L'Aquila, Italy   July 31st – August 6th, 2000

Motivations & Introduction

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 (such as 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.

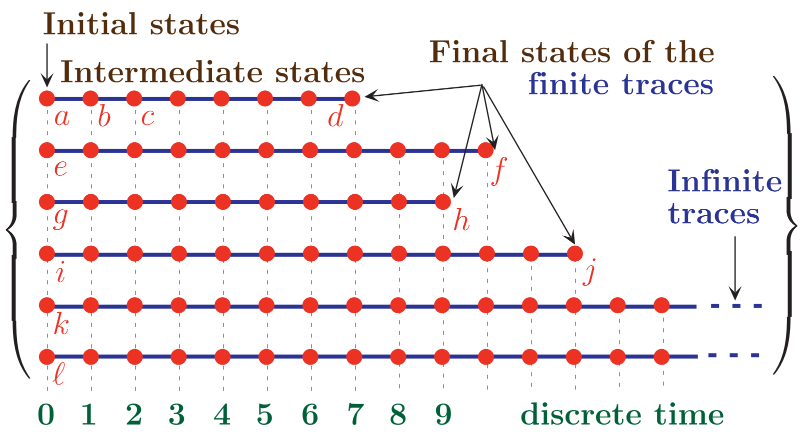
Content

Semantics

Semantics: intuition

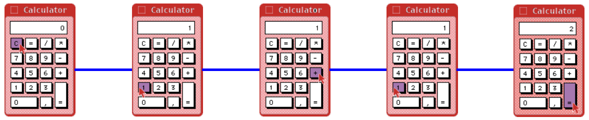
- The semantics of a language defines the semantics of any program written in this language;
- The semantics of a program provides a formal mathematical model of all possible behaviors of a computer system executing this program (interacting with any possible environment);
- Any semantics of a program can be defined as the solution of a fixpoint equation;
- All semantics of a program can be organized in a hierarchy by abstraction.

Example: trace semantics [4, 6]



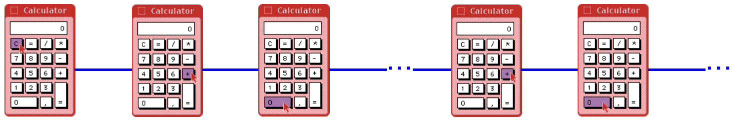
Examples of computation traces

- Finite (C1+1=):



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

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



Least Fixpoints: intuition [4, 6]

$$
\begin{aligned}
\textbf{Behaviors} = \; & \{\bullet \mid \bullet \text{ is a final state}\} \\
\cup \; & \{\bullet\!\!-\!\!\bullet\!\!-\!\!\ldots\!\!-\!\!\bullet \mid \bullet\!\!-\!\!\bullet \text{ is an elementary step } \& \\
& \quad \bullet\!\!-\!\!\ldots\!\!-\!\!\bullet \in \textbf{Behaviors}^{+}\} \\
\cup \; & \{\bullet\!\!-\!\!\bullet\!\!-\!\!\ldots\!\!-\!\!\ldots \mid \bullet\!\!-\!\!\bullet \text{ is an elementary step } \& \\
& \quad \bullet\!\!-\!\!\ldots\!\!-\!\!\ldots \in \textbf{Behaviors}^{\infty}\}
\end{aligned}
$$

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

« more finite traces & less infinite traces ».

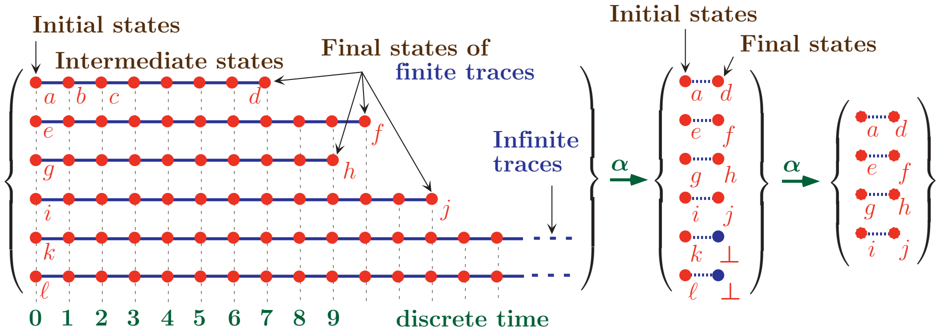
Abstraction

Abstraction: intuition

- Abstract interpretation is a theory of the approximation of the behavior of discrete dynamic 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 [1]

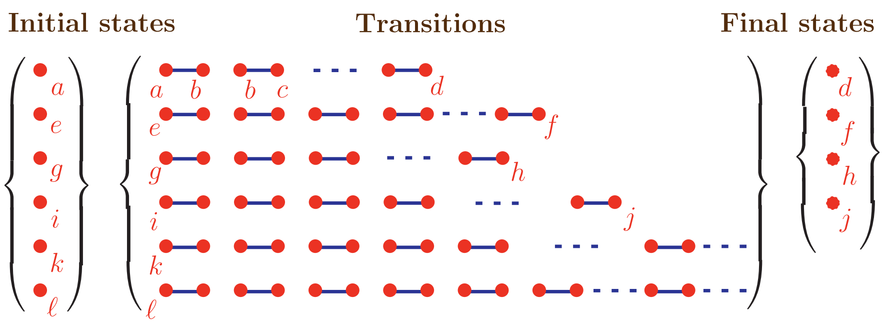


Trace semantics

Denotational semantics

Natural semantics

Example 2 of abstraction [2]

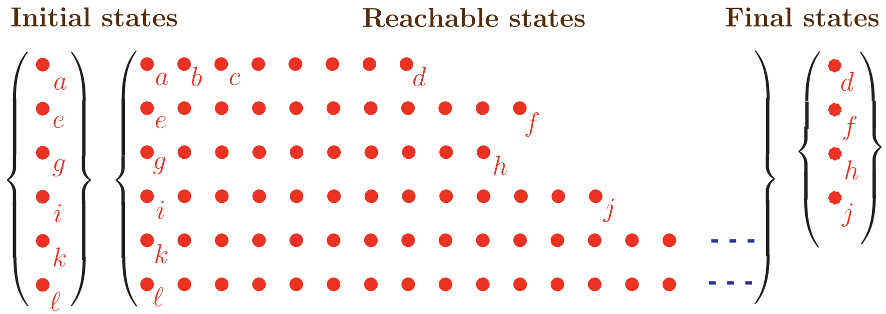


(Small-Step) Operational Semantics

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

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

Example 3 of abstraction [3]



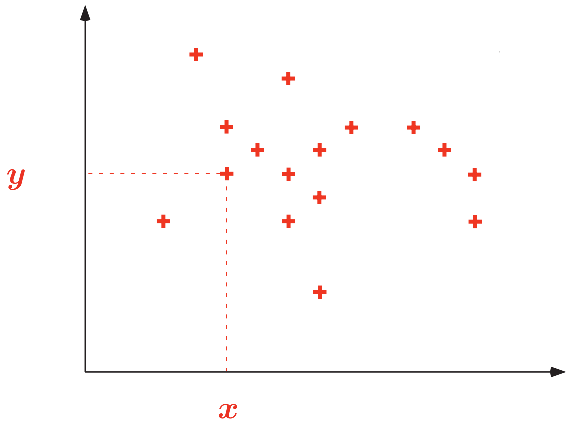
Collecting Semantics

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;
- The computation of this abstract semantics amounts to the effective iterative resolution of fixpoint equations;
- 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].

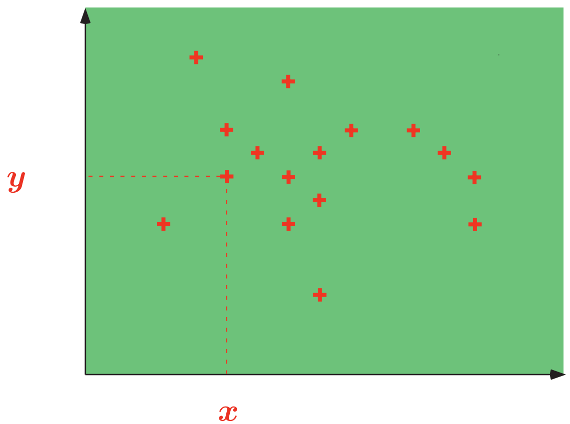
[3] P. Cousot. *Constructive design of a hierarchy of semantics of a transition system by abstract interpretation.* To appear in TCS (2000).

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



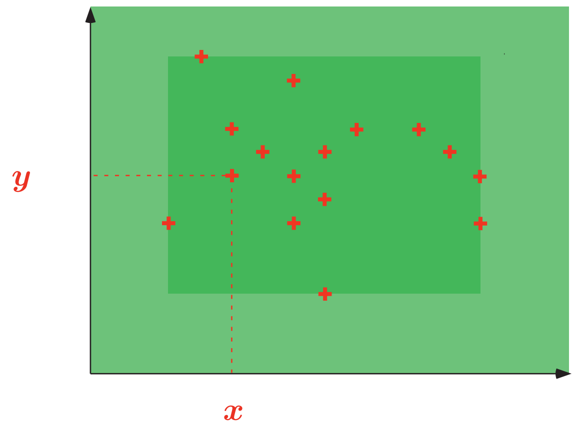
$$\{\ldots, \langle 19,\ 88\rangle, \ldots,\ \langle 19,\ 99\rangle, \ldots\}$$

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



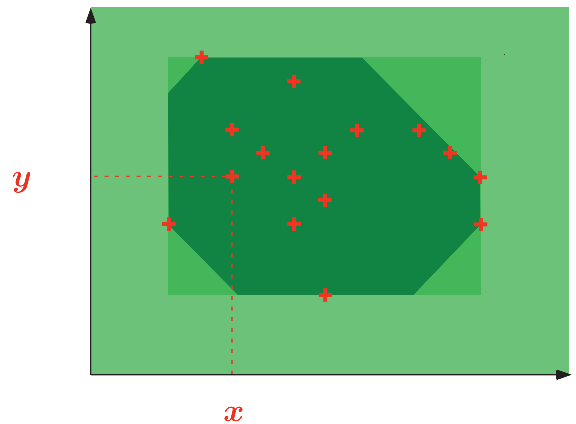
$$\begin{cases} x \geq 0 \\ y \geq 0 \end{cases}$$

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



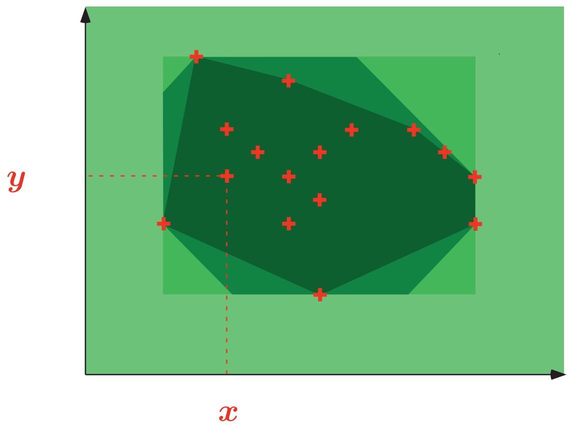
$$\begin{cases} x \in [19,\ 88] \\ y \in [19,\ 99] \end{cases}$$

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



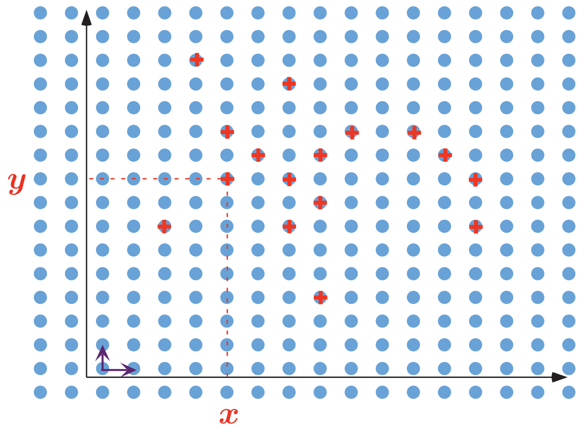
$$\begin{cases} 1 \leq x \leq 9 \\ x + y \leq 88 \\ 1 \leq y \leq 9 \\ x - y \leq 99 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 4: polyhedra [13]



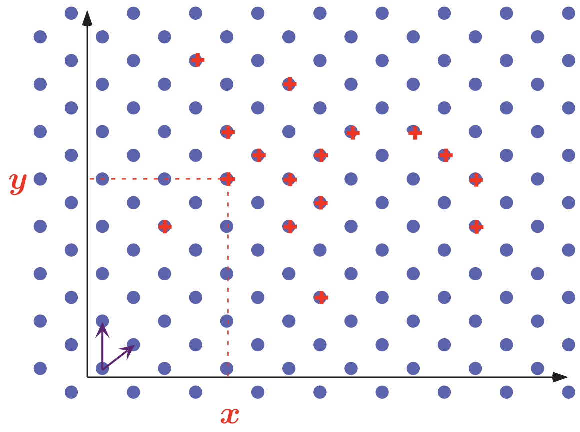
$$\begin{cases} 19x + 88y \leq 2000 \\ 19x + 99y \geq 0 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 5: simple congruences [15]



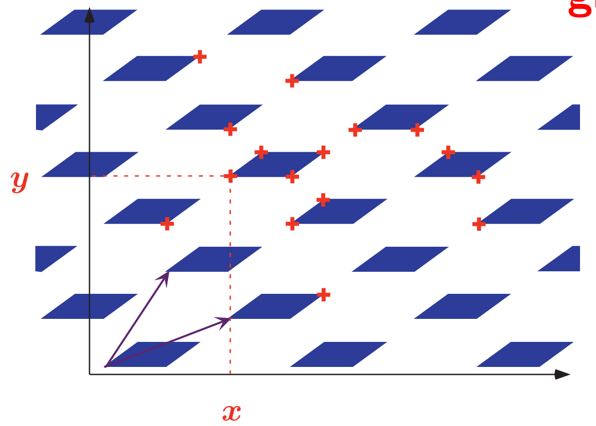
$$\begin{cases} x = 19 \bmod 88 \\ y = 19 \bmod 99 \end{cases}$$

Computable abstractions of an [in]finite set of points; Example 6: linear congruences [16]



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

Computable abstractions of an [in]finite set of points; Example 7: trapezoidal linear congruences [18, 19]



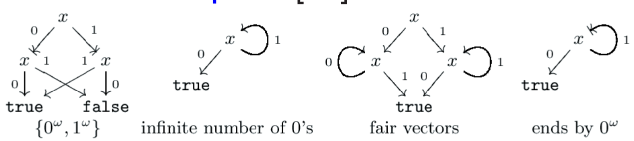
$$\begin{cases} 1x + 9y \in [0, 88] \bmod 10 \\ 1x - 9y \in [0, 99] \bmod 11 \end{cases}$$

Computable abstractions of symbolic structures

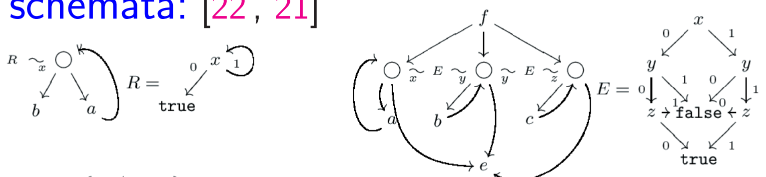
- Most structures manipulated by programs are *symbolic structures* such as control structures (call graphs), data structures (search trees), communication structures (distributed & mobile programs), etc;
- It is very difficult to find compact and expressive abstractions of such sets of objects (languages,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

Example of abstractions of infinite sets of infinite trees

Binary Decision Graphs: [20]



Tree schemata: [22, 21]



$\{a^n b \mid n \in \mathbb{N}\}$

$\{f(a^n e, b^n e, c^n e) \mid n \in \mathbb{N}\}$

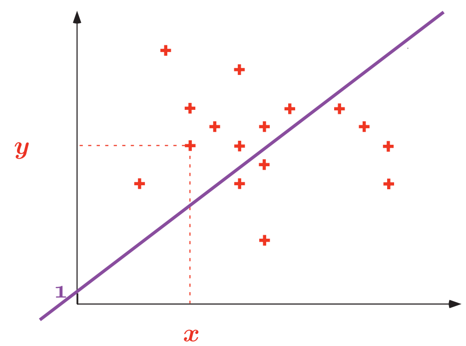
Note that E is the equality relation.

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.

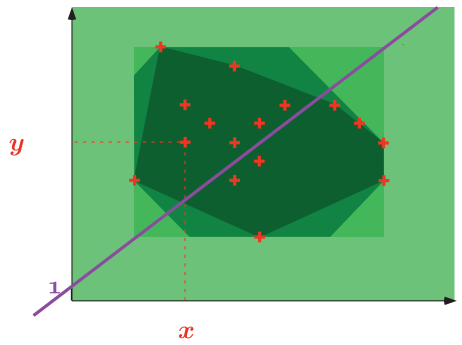
Example of information loss

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



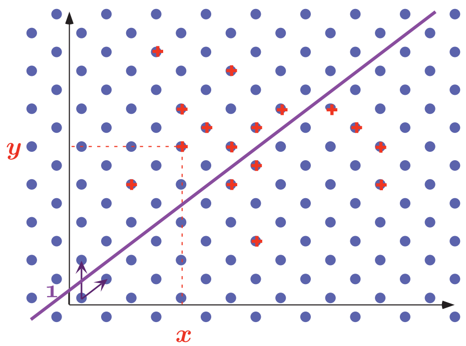
Example of information loss

- 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!
- Fully automatic bug detection is impossible (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) [4]

Program to be analyzed:

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


Example: interval analysis (1975)

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.

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

Example: interval analysis (1975)

Constraints (abstract interpretation of the semantics):

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

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

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

Example: interval analysis (1975)

Increasing chaotic iteration, initialization:

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

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

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

Example: interval analysis (1975) [4]

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)

Increasing chaotic iteration:

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

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

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

Example: interval analysis (1975)

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)

Increasing chaotic iteration:

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

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

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

Example: interval analysis (1975)

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)

Increasing chaotic iteration: **convergence??**

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

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

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

Example: interval analysis (1975)

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)

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

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

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

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

Example: interval analysis (1975)

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)

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

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

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

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

Example: interval analysis (1975)

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)

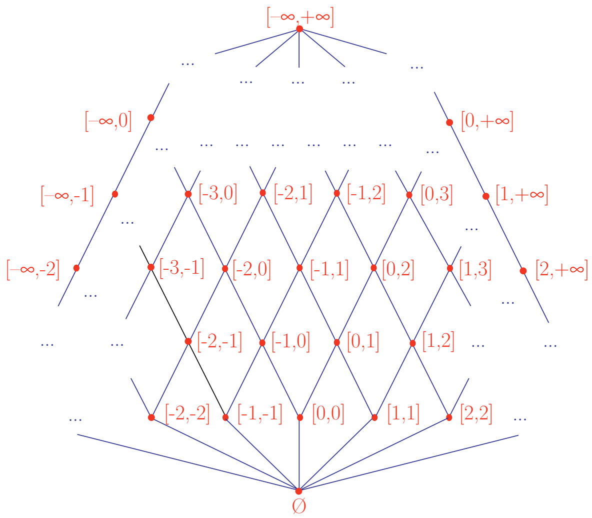
Convergence speed-up by extrapolation:

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

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

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

Widening



Example: interval analysis (1975)

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)

Decreasing chaotic iteration:

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

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

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

Example: interval analysis (1975)

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)

Final solution:

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

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

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

Example: interval analysis (1975)

Result of the interval analysis:

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

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

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

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

Example: interval analysis (1975)

Exploitation of the result of the interval analysis:

```
   x := 1;
1: {x = 1}
   while x < 10000 do
2: {x ∈ [1,9999]}
       x := x + 1          ←—— no overflow
3: {x ∈ [2,10000]}
   od;
4: {x = 10000}
```


Some applications of static analysis by abstract interpretation

- data flow and set-based analysis for program optimization & transformation (including partial evaluation) [9, 12];
- type inference (including undecidable systems)/soft typing [5];
- abstract model-checking of infinite systems [11, 12];
- abstract debugging & testing [2, 1];
- probabilistic analysis [24];
- communication topology analysis for mobile/distributed code [25];
- ...

Some other recent applications of abstract interpretation

- Fundamental applications:
  - design of hierarchies of semantics [10, 3, 6], ...;
- Practical applications:
  - generation of heuristics for search problems in AI;
  - automatic differentiation of numerical programs;
  - security (analysis of cryptographic protocols [23], mobile code [14]);
  - semantic tattooing/watermarking of software, ...;

Present-day and forthcoming research

A lot of fundamental research remains to be done:

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

Industrialization of Program Static Analysis

An impressive application (1996/97)

- *Abstract interpretation* has been used (including interval analysis) for the static analysis of the embedded ADA software of the Ariane 5 launcher [5]; [17]
- Automatic detection of the definiteness, potentiality, impossibility or inaccessibility of run-time errors [6];
- Automatic discovery of the 501 flight error;
- Success for the 502 & 503 flights and the ARD [7].

[4] Flight software (60,000 lines of Ada code) and Inertial Measurement Unit (30,000 lines of Ada code).
[5] 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.
[6] Atmospheric Reentry Demonstrator: module coming back to earth.

Industrialization of static analysis by abstract interpretation

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

References

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/

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. 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.

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

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] L. Mauborgne. Tree schemata and fair termination. 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 302–321. Springer-Verlag, Berlin, Germany, 29 June – 1 July 2000.

[22] 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.

[23] 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.

[24] 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-Verlag, Berlin, Germany, 29 June – 1 July 2000.

[25] 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.