Challenges in Abstract Interpretation for Software Safety

Patrick Cousot

École normale supérieure, Paris, France

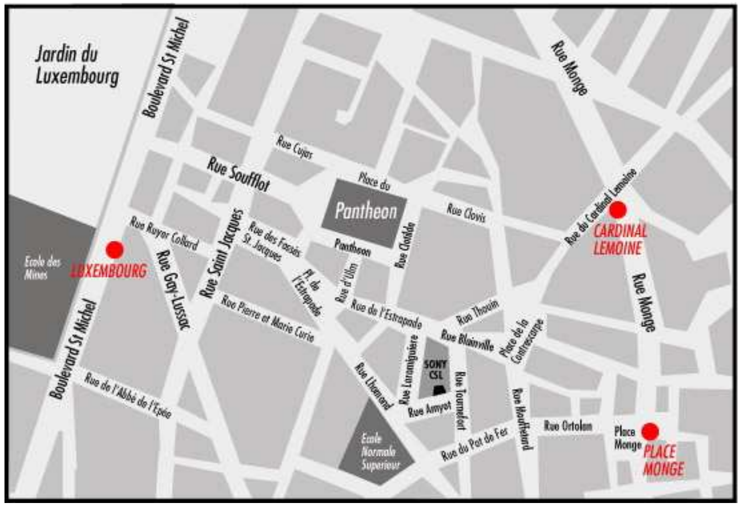
cousot@ens.fr   www.di.ens.fr/~cousot

Franco-Japanese Workshop on Security

Keio University, Mita Campus, September 5–7, 2005

ENS

École Normale Supérieure (ENS)



Normale Sup. (ENS)

A few former students: Évariste Galois, Louis Pasteur, ...; Nobel prizes: Claude Cohen-Tannoudji, Pierre-Gilles de Gennes, Gabriel Lippmann, Louis Néel, Jean-Baptiste Perrin, Paul Sabatier, ...; Fields Medal holders: Laurent Schwartz, Jean-Pierre Serre (1st Abel Prize), René Thom, Alain Connes, Pierre-Louis Lions, Jean-Christophe Yoccoz, Laurent Lafforgue; Fictious mathematicians: Nicolas Bourbaki; Philosophers: Henri Bergson (Nobel Prize), Louis Althusser, Simone de Beauvoir, Emile Auguste Chartier "Alain", Raymond Aron, Jean-Paul Sartre, Maurice Merleau-Ponty, Michel Foucault, Jacques Derrida, Bernard-Henri Lévy...; Politicians: Jean Jaurès, Léon Blum, Édouard Herriot, Georges Pompidou, Alain Juppé, Laurent Fabius, Léopold Sédar Senghor,...; Sociologists: Émile Durkheim, Pierre Bourdieu, ...; Writers: Romain Rolland (Nobel Prize), Jean Giraudoux, Charles Péguy, Julien Gracq, ...;

State of Practice in Software Engineering

An example among many others (Matlab code)

```
» h=get(gca,'children');

apple.awt.EventQueueExceptionHandler Caught Throwable : java.lang.ArrayIndexOutOfBoundsException: 2 >= 2
java.lang.ArrayIndexOutOfBoundsException: 2 >= 2
at java.util.Vector.elementAt(Vector.java:431)
at com.mathworks.mde.help.IndexItem.getFilename(IndexItem.java:100)
at com.mathworks.mde.help.Index.getFilenameForLocation(Index.java:706)
at com.mathworks.mde.help.Index.access$3100(Index.java:29)
at com.mathworks.mde.help.Index$IndexMouseMotionAdapter.mouseMoved(Index.java:768)
at java.awt.AWTEventMulticaster.mouseMoved(AWTEventMulticaster.java:272)
at java.awt.AWTEventMulticaster.mouseMoved(AWTEventMulticaster.java:271)
at java.awt.Component.processMouseMotionEvent(Component.java:5211)
at javax.swing.JComponent.processMouseMotionEvent(JComponent.java:2779)
at com.mathworks.mwswing.MJTable.processMouseMotionEvent(MJTable.java:725)
at java.awt.Component.processEvent(Component.java:4967)
at java.awt.Container.processEvent(Container.java:1613)
at java.awt.Component.dispatchEventImpl(Component.java:3681)
at java.awt.Container.dispatchEventImpl(Container.java:1671)
at java.awt.Component.dispatchEvent(Component.java:3543)
at java.awt.LightweightDispatcher.retargetMouseEvent(Container.java:3527)
at java.awt.LightweightDispatcher.processMouseEvent(Container.java:3255)
at java.awt.LightweightDispatcher.dispatchEvent(Container.java:3172)
at java.awt.Container.dispatchEventImpl(Container.java:1657)
at java.awt.Window.dispatchEventImpl(Window.java:1606)
at java.awt.Component.dispatchEvent(Component.java:3543)
at java.awt.EventQueue.dispatchEvent(EventQueue.java:456)
at java.awt.EventDispatchThread.pumpOneEventForHierarchy(EventDispatchThread.java:234)
at java.awt.EventDispatchThread.pumpEventsForHierarchy(EventDispatchThread.java:184)
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:178)
at java.awt.EventDispatchThread.pumpEvents(EventDispatchThread.java:170)
at java.awt.EventDispatchThread.run(EventDispatchThread.java:100)
»
```


The software safety challenge for next 10 years

- Present-day software engineering is almost exclusively manual, with very few automated tools;
- Trust and confidence in specifications and software can no longer be entirely based on the development process (e.g. DO178B in aerospace software);
- In complement, quality assurance must be ensured by new design, modeling, checking, verification and certification tools based on the product itself.

Abstract Interpretation

Reference

[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 4^{th} *ACM POPL*.

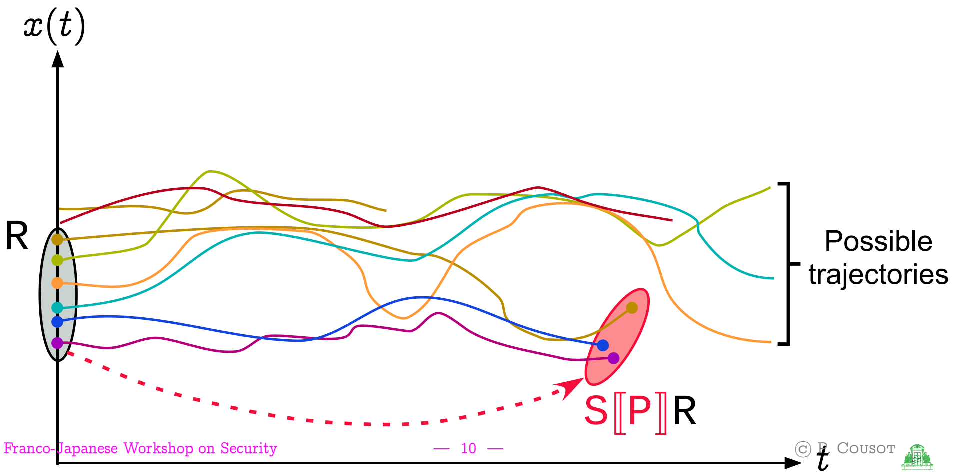
[Thesis '78] P. Cousot. Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes. Thèse ès sci. math. Grenoble, march 1978.

[POPL '79] P. Cousot & R. Cousot. Systematic design of program analysis frameworks. In 6^{th} *ACM POPL*.

Syntax of programs

X	variables $X \in \mathbb{X}$
T	types $T \in \mathbb{T}$
E	arithmetic expressions $E \in \mathbb{E}$
B	boolean expressions $B \in \mathbb{B}$
$D ::= T\ X\,;$ $\quad\mid\ T\ X\ ;\ D'$	
$C ::= X = E;$ $\quad\mid\ \texttt{while}\ B\ C'$ $\quad\mid\ \texttt{if}\ B\ C'\ \texttt{else}\ C''$ $\quad\mid\ \{\ \mathrm{C}_1\ \ldots\ \mathrm{C}_n\ \},\ (n \geq 0)$	commands $C \in \mathbb{C}$
$P ::= D\ C$	program $P \in \mathbb{P}$

Postcondition semantics



States

Values of given type:

$$\mathcal{V}[\![T]\!] \ :\ \text{values of type } T \in \mathbb{T}$$
$$\mathcal{V}[\![\texttt{int}]\!] \stackrel{\text{def}}{=} \{z \in \mathbb{Z} \mid \texttt{min\_int} \leq z \leq \texttt{max\_int}\}$$

Program states $\Sigma[\![P]\!]$ [1]:

$$\Sigma[\![D\ C]\!] \stackrel{\text{def}}{=} \Sigma[\![D]\!]$$
$$\Sigma[\![T\ X\,;]\!] \stackrel{\text{def}}{=} \{X\} \mapsto \mathcal{V}[\![T]\!]$$
$$\Sigma[\![T\ X\,;\ D]\!] \stackrel{\text{def}}{=} (\{X\} \mapsto \mathcal{V}[\![T]\!]) \cup \Sigma[\![D]\!]$$

Concrete Semantic Domain of Programs

Concrete semantic domain for reachability properties:

$$\mathcal{D}[\![P]\!] \stackrel{\text{def}}{=} \wp(\Sigma[\![P]\!]) \qquad \text{sets of states}$$

i.e. program properties where $\subseteq$ is implication, $\emptyset$ is false, $\cup$ is disjunction.

[1] States $\rho \in \Sigma[\![P]\!]$ of a program P map program variables X to their values $\rho(X)$

Concrete Reachability Semantics of Programs

$$\mathcal{S}[\![X = E;]\!]R \stackrel{\text{def}}{=} \{\rho[X \leftarrow \mathcal{E}[\![E]\!]\rho] \mid \rho \in R \cap \text{dom}(E)\}$$

$$\rho[X \leftarrow v](X) \stackrel{\text{def}}{=} v, \qquad \rho[X \leftarrow v](Y) \stackrel{\text{def}}{=} \rho(Y)$$

$$\mathcal{S}[\![\texttt{if } B\ C']\!]R \stackrel{\text{def}}{=} \mathcal{S}[\![C']\!](\mathcal{B}[\![B]\!]R) \cup \mathcal{B}[\![\neg B]\!]R$$

$$\mathcal{B}[\![B]\!]R \stackrel{\text{def}}{=} \{\rho \in R \cap \text{dom}(B) \mid B \text{ holds in } \rho\}$$

$$\mathcal{S}[\![\texttt{if } B\ C' \texttt{ else } C'']\!]R \stackrel{\text{def}}{=} \mathcal{S}[\![C']\!](\mathcal{B}[\![B]\!]R) \cup \mathcal{S}[\![C'']\!](\mathcal{B}[\![\neg B]\!]R)$$

$$\mathcal{S}[\![\texttt{while } B\ C']\!]R \stackrel{\text{def}}{=} \text{let } \mathcal{W} = \mathsf{lfp}^{\subseteq}_{\emptyset}\ \lambda \mathcal{X} \cdot R \cup \mathcal{S}[\![C']\!](\mathcal{B}[\![B]\!]\mathcal{X}) \text{ in } (\mathcal{B}[\![\neg B]\!]\mathcal{W})$$

$$\mathcal{S}[\![\{\}]\!]R \stackrel{\text{def}}{=} R$$

$$\mathcal{S}[\![\{C_1 \ldots C_n\}]\!]R \stackrel{\text{def}}{=} \mathcal{S}[\![C_n]\!] \circ \ldots \circ \mathcal{S}[\![C_1]\!] \quad n > 0$$

$$\mathcal{S}[\![D\ C]\!]R \stackrel{\text{def}}{=} \mathcal{S}[\![C]\!](\Sigma[\![D]\!]) \quad \text{(uninitialized variables)}$$

Not computable (undecidability).

Abstract Semantic Domain of Programs

$$\langle \mathcal{D}^\sharp[\![P]\!],\ \sqsubseteq,\ \bot,\ \sqcup \rangle$$

such that:

$$\langle \mathcal{D},\ \subseteq \rangle \underset{\alpha}{\overset{\gamma}{\longleftrightarrow}} \langle \mathcal{D}^\sharp[\![P]\!],\ \sqsubseteq \rangle$$

hence $\langle \mathcal{D}^\sharp[\![P]\!],\ \sqsubseteq,\ \bot,\ \sqcup \rangle$ is a complete lattice such that $\bot = \alpha(\emptyset)$ and $\sqcup X = \alpha(\cup \gamma(X))$

Reduced Product of Abstract Domains

To combine abstractions

$$\langle \mathcal{D},\ \subseteq \rangle \underset{\alpha_1}{\overset{\gamma_1}{\longleftrightarrow}} \langle \mathcal{D}^\sharp_1,\ \sqsubseteq_1 \rangle \text{ and } \langle \mathcal{D},\ \subseteq \rangle \underset{\alpha_2}{\overset{\gamma_2}{\longleftrightarrow}} \langle \mathcal{D}^\sharp_2,\ \sqsubseteq_2 \rangle$$

the reduced product is

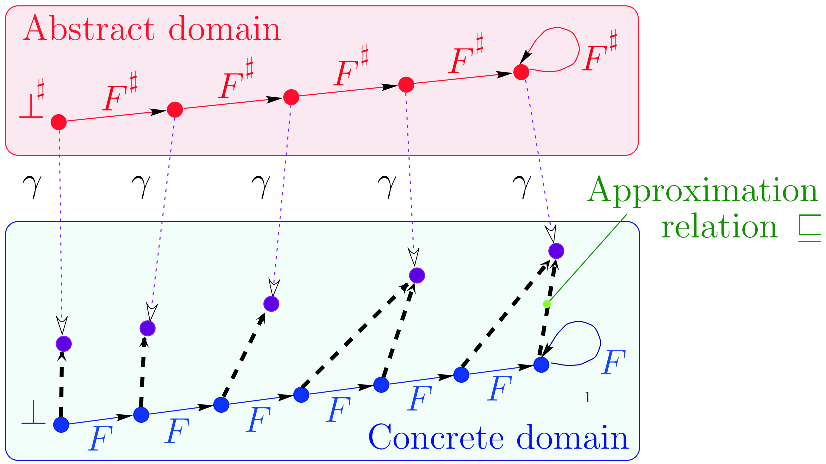
$$\alpha(X) \stackrel{\text{def}}{=} \sqcap \{\langle x,\ y \rangle \mid X \subseteq \gamma_1(X) \wedge X \subseteq \gamma_2(X)\}$$

such that $\sqsubseteq \stackrel{\text{def}}{=} \sqsubseteq_1 \times \sqsubseteq_2$ and

$$\langle \mathcal{D},\ \subseteq \rangle \underset{\alpha}{\overset{\gamma_1 \times \gamma_2}{\longleftrightarrow}} \langle \alpha(\mathcal{D}),\ \sqsubseteq \rangle$$

Example: $\mathtt{x} \in [1, 9] \wedge \mathtt{x} \bmod 2 = 0$ reduces to $\mathtt{x} \in [2, 8] \wedge \mathtt{x} \bmod 2 = 0$

Approximate Fixpoint Abstraction

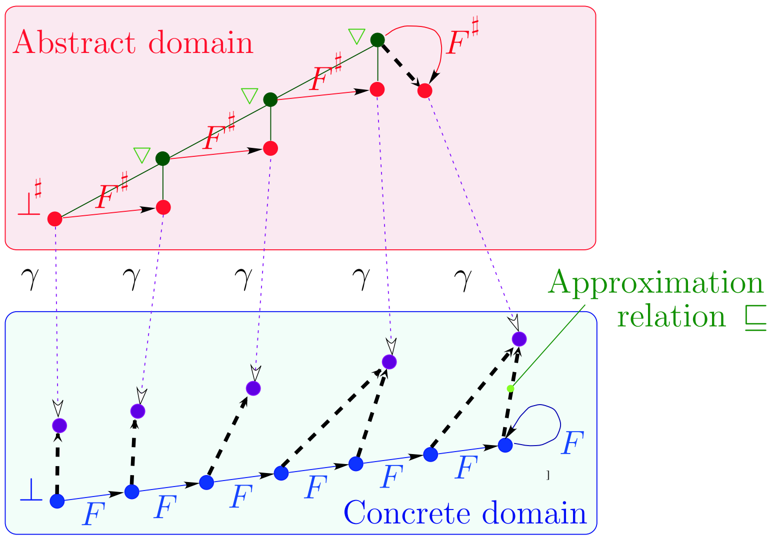


$$F \circ \gamma \sqsubseteq \gamma \circ F^\sharp \Rightarrow \mathsf{lfp}\, F \sqsubseteq \gamma(\mathsf{lfp}\, F^\sharp)$$

Abstract Reachability Semantics of Programs

$$\mathcal{S}^\sharp[\![X = E;]\!]R \stackrel{\text{def}}{=} \alpha(\{\rho[X \leftarrow \mathcal{E}[\![E]\!]\rho] \mid \rho \in \gamma(R) \cap \text{dom}(E)\})$$
$$\mathcal{S}^\sharp[\![\texttt{if } B\ C']\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C']\!](\mathcal{B}^\sharp[\![B]\!]R) \sqcup \mathcal{B}^\sharp[\![\neg B]\!]R$$
$$\mathcal{B}^\sharp[\![B]\!]R \stackrel{\text{def}}{=} \alpha(\{\rho \in \gamma(R) \cap \text{dom}(B) \mid B \text{ holds in } \rho\})$$
$$\mathcal{S}^\sharp[\![\texttt{if } B\ C' \texttt{ else } C'']\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C']\!](\mathcal{B}^\sharp[\![B]\!]R) \sqcup \mathcal{S}^\sharp[\![C'']\!](\mathcal{B}^\sharp[\![\neg B]\!]R)$$
$$\mathcal{S}^\sharp[\![\texttt{while } B\ C']\!]R \stackrel{\text{def}}{=} \text{let } \mathcal{W} = \textsf{lfp}^{\sqsubseteq}_{\bot}\ \lambda \mathcal{X} \cdot R \sqcup \mathcal{S}^\sharp[\![C']\!](\mathcal{B}^\sharp[\![B]\!]\mathcal{X}) \text{ in } (\mathcal{B}^\sharp[\![\neg B]\!]\mathcal{W})$$
$$\mathcal{S}^\sharp[\![\{\}]\!]R \stackrel{\text{def}}{=} R$$
$$\mathcal{S}^\sharp[\![\{C_1 \ldots C_n\}]\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C_n]\!] \circ \ldots \circ \mathcal{S}^\sharp[\![C_1]\!] \quad n > 0$$
$$\mathcal{S}^\sharp[\![D\ C]\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C]\!](\top) \quad \text{(uninitialized variables)}$$

Convergence Acceleration with Widening



Abstract Semantics with Convergence Acceleration [2]

$$\mathcal{S}^\sharp[\![X = E;]\!]R \stackrel{\text{def}}{=} \alpha(\{\rho[X \leftarrow \mathcal{E}[\![E]\!]\rho] \mid \rho \in \gamma(R) \cap \text{dom}(E)\})$$
$$\mathcal{S}^\sharp[\![\texttt{if } B\ C']\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C']\!](\mathcal{B}^\sharp[\![B]\!]R) \sqcup \mathcal{B}^\sharp[\![\neg B]\!]R$$
$$\mathcal{B}^\sharp[\![B]\!]R \stackrel{\text{def}}{=} \alpha(\{\rho \in \gamma(R) \cap \text{dom}(B) \mid B \text{ holds in } \rho\})$$
$$\mathcal{S}^\sharp[\![\texttt{if } B\ C' \texttt{ else } C'']\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C']\!](\mathcal{B}^\sharp[\![B]\!]R) \sqcup \mathcal{S}^\sharp[\![C'']\!](\mathcal{B}^\sharp[\![\neg B]\!]R)$$
$$\mathcal{S}^\sharp[\![\texttt{while } B\ C']\!]R \stackrel{\text{def}}{=} \text{let } \mathcal{F}^\sharp = \lambda \mathcal{X} \cdot \text{let } \mathcal{Y} = R \sqcup \mathcal{S}^\sharp[\![C']\!](\mathcal{B}^\sharp[\![B]\!]\mathcal{X}) \text{ in if } \mathcal{Y} \sqsubseteq \mathcal{X} \text{ then } \mathcal{X} \text{ else } \mathcal{X} \triangledown \mathcal{Y} \text{ and } \mathcal{W} = \textsf{lfp}^{\sqsubseteq}_{\bot}\ \mathcal{F}^\sharp \quad \text{in } (\mathcal{B}^\sharp[\![\neg B]\!]\mathcal{W})$$
$$\mathcal{S}^\sharp[\![\{\}]\!]R \stackrel{\text{def}}{=} R$$
$$\mathcal{S}^\sharp[\![\{C_1 \ldots C_n\}]\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C_n]\!] \circ \ldots \circ \mathcal{S}^\sharp[\![C_1]\!] \quad n > 0$$
$$\mathcal{S}^\sharp[\![D\ C]\!]R \stackrel{\text{def}}{=} \mathcal{S}^\sharp[\![C]\!](\top) \quad \text{(uninitialized variables)}$$

Applications of Abstract Interpretation

[2] Note: $\mathcal{F}^\sharp$ not monotonic!

Applications of Abstract Interpretation

- **Static Program Analysis** [POPL '77], [POPL '78], [POPL '79] including **Dataflow Analysis** [POPL '79], [POPL '00], **Set-based Analysis** [FPCA '95], **Predicate Abstraction** [Manna's festschrift '03], ...
- **Syntax Analysis** [TCS 290(1) 2002]
- **Hierarchies of Semantics (including Proofs)** [POPL '92], [TCS 277(1–2) 2002]
- **Typing & Type Inference** [POPL '97]

Applications of Abstract Interpretation (Cont'd)

- **(Abstract) Model Checking** [POPL '00]
- **Program Transformation** [POPL '02]
- **Software Watermarking** [POPL '04]
- **Bisimulations** [RT-ESOP '04]

All these techniques involve sound approximations that can be formalized by abstract interpretation

Static Analysis

Reference

[1] 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.

State of the Art in Automatic Static Program Analysis

Static analysis tools

- Determine automatically from the program text program properties of a certain class that do hold at runtime (e.g. absence of runtime error);
- Based on the automatic computation of machine representable abstractions[3] of all possible executions of the program in any possible environment;
- Scales up to hundreds of thousands lines;
- Undecidable whence false alarms are possible[4]

Degree of specialization

- Specialization for a class of runtime properties (e.g. absence of runtime errors)
- Specialization for a programming language (e.g. PolySpace Suite for Ada, C or C++)
- Specialization for a programming style (e.g. C Global Surveyor)
- Specialization for an application type (e.g. ASTRÉE for embedded real-time synchronous[5] autocodes)

⇒ The more specialized, the less false alarms[6]!

[3] sound but (in general) uncomplete approximations.
[4] cases when a question on the program runtime behavior cannot be answered automatically for sure
[5] deterministic
[6] but the less specialized, the larger commercial market (and the less client satisfaction)!

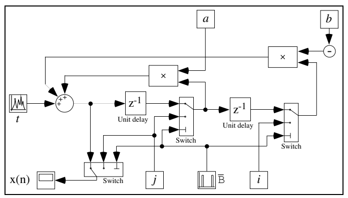
The ASTRÉE static analyzer (www.astree.ens.fr)

- ASTRÉE is a static program analyzer aiming at proving the absence of Run Time Errors (started Nov. 2001)
- C programs, no dynamic memory allocation and recursion
- Encompass many (automatically generated) synchronous, time-triggered, real-time, safety critical, embedded software
- automotive, energy and aerospace applications

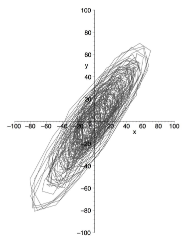
⇒ e.g. No false alarm on the electric flight control codes for the A340 (Nov. 2003) and A380 (Nov. 2004) generated from SAO/SCADE.

Ellipsoid Abstract Domain for Filters

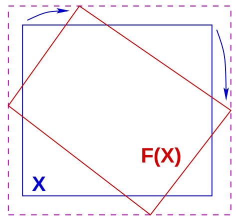
2[d] Order Digital Filter:



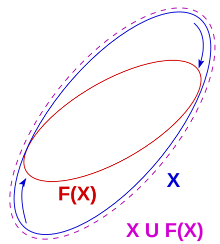
- Computes $X_n = \begin{cases} \alpha X_{n-1} + \beta X_{n-2} + Y_n \\ I_n \end{cases}$
- The concrete computation is bounded, which must be proved in the abstract.
- There is no stable interval or octagon.
- The simplest stable surface is an ellipsoid.



execution trace



unstable interval



stable ellipsoid

Filter Example

```
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
BOOLEAN INIT; float P, X;
void filter () {
  static float E[2], S[2];
  if (INIT) { S[0] = X; P = X; E[0] = X; }
  else { P = (((((0.5 * X) - (E[0] * 0.7)) + (E[1] * 0.4))
             + (S[0] * 1.5)) - (S[1] * 0.7)); }
  E[1] = E[0]; E[0] = X; S[1] = S[0]; S[0] = P;
  /* S[0], S[1] in [-1327.02698354, 1327.02698354] */
}
void main () { X = 0.2 * X + 5; INIT = TRUE;
  while (1) {
    X = 0.9 * X + 35; /* simulated filter input */
    filter (); INIT = FALSE; }
}
```

Reference

see http://www.astree.ens.fr/

Arithmetic-geometric progressions

– Abstract domain: $(\mathbb{R}^+)^5$ [7]

– Concretization (any function bounded by the arithmetic-geometric progression):

$$\gamma \in (\mathbb{R}^+)^5 \longmapsto \wp(\mathbb{N} \mapsto \mathbb{R})$$

$$\gamma(M, a, b, a', b') =$$

$$\{f \mid \forall k \in \mathbb{N} : |f(k)| \leq \left(\lambda x \cdot ax + b \circ (\lambda x \cdot a'x + b')^k\right)(M)\}$$

Reference

see http://www.astree.ens.fr/

[7] here in $\mathbb{R}$

Arithmetic-Geometric Progressions (Example 1)

```
% cat count.c
typedef enum {FALSE = 0, TRUE = 1} BOOLEAN;
volatile BOOLEAN I; int R; BOOLEAN T;
void main() {
  R = 0;
  while (TRUE) {
    __ASTREE_log_vars((R));
    if (I) { R = R + 1; }                ← potential overflow!
    else { R = 0; }
    T = (R >= 100);
    __ASTREE_wait_for_clock(());
  }}
% cat count.config
__ASTREE_volatile_input((I [0,1]));
__ASTREE_max_clock((3600000));
% astree -exec-fn main -config-sem count.config count.c|grep '|R|'
|R| <= 0. + clock *1. <= 3600001.
```


Arithmetic-geometric progressions (Example 2)

```
% cat retro.c
typedef enum {FALSE=0, TRUE=1} BOOL;
BOOL FIRST;
volatile BOOL SWITCH;
volatile float E;
float P, X, A, B;

void dev( )
{ X=E;
  if (FIRST) { P = X; }
  else
    { P = (P - ((((2.0 * P) - A) - B)
            * 4.491048e-03)); };
  B = A;
  if (SWITCH) {A = P;}
  else {A = X;}
}
```

```
void main()
{ FIRST = TRUE;
  while (TRUE) {
    dev( );
    FIRST = FALSE;
    __ASTREE_wait_for_clock(());
  }}
% cat retro.config
__ASTREE_volatile_input((E [-15.0, 15.0]));
__ASTREE_volatile_input((SWITCH [0,1]));
__ASTREE_max_clock((3600000));
|P| <= (15. + 5.87747175411e-39
/ 1.19209290217e-07) * (1
+ 1.19209290217e-07)^clock
- 5.87747175411e-39 /
1.19209290217e-07 <=
23.0393526881
```

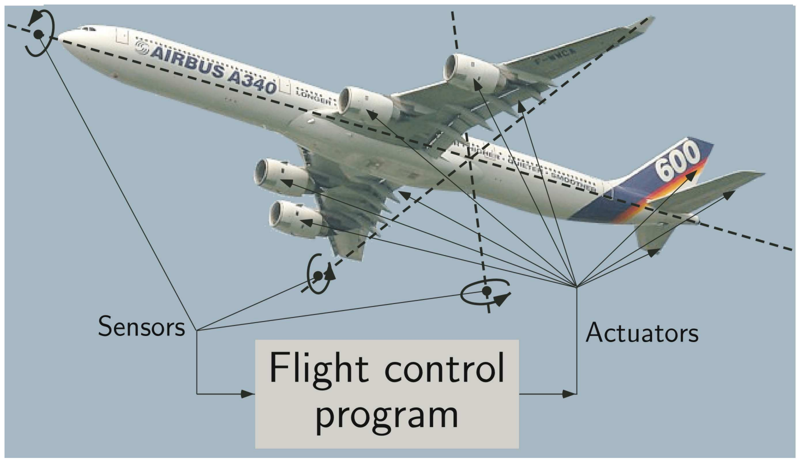
Towards System Verification Tools

Reference

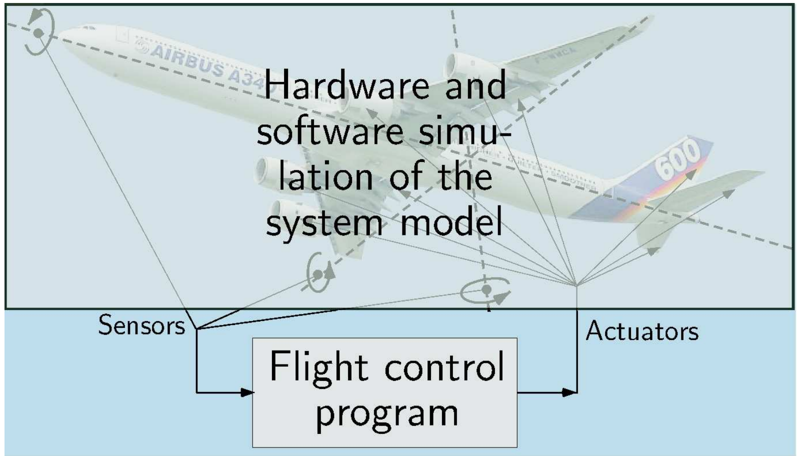
[2] P. Cousot. Proving Program Invariance and Termination by Parametric Abstraction, Lagrangian Relaxation and Semidefinite Programming, invited paper. In *Sixth International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'05)*, pages 1–24, Paris, France, January 17-19, 2005. Lecture Notes in Computer Science, volume 3385, Springer, Berlin.

[APLAS'06] P. Cousot. Integrating Physical Systems in the Static Analysis of Embedded Control Software., invited talk In *APLAS'06*, Tokyo, Nov. 2005, to appear (LNCS).

Computer controlled systems



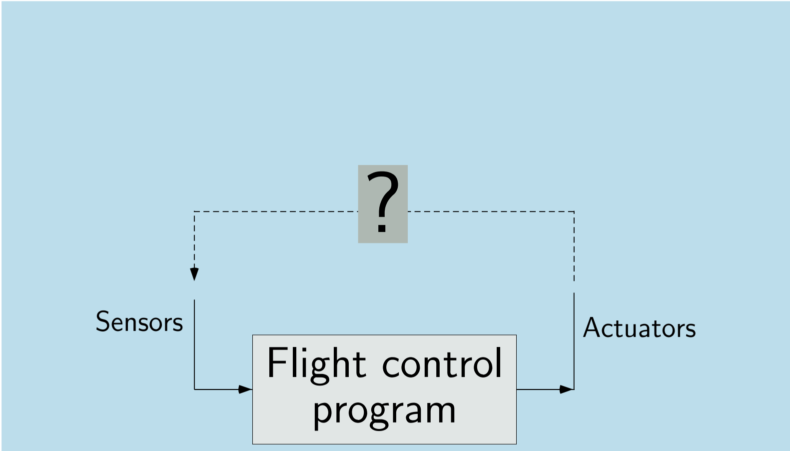
Software test



Abstractions: program $\longrightarrow$ none, system $\longrightarrow$ precise

- Very expensive
- Not exhaustive
- Extended during flight test period
- Late discovery of errors can delay the program by months (the whole software development process must be rechecked)

Software analysis & verification with ASTRÉE



Abstractions: program $\longrightarrow$ precise, system $\longrightarrow$ coarse

– Exhaustive

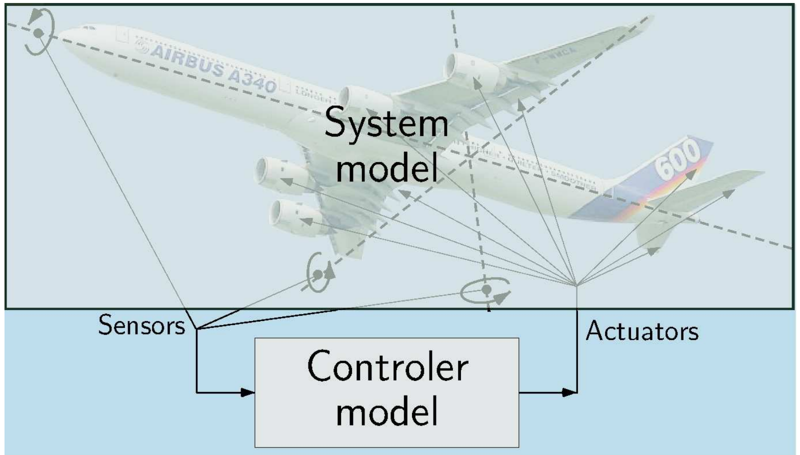
– Can be made precise by specialization[8] to get no false alarm

– No specification of the controlled system (but for ranges of values of a few sensors)

– Impossible to prove essential properties of the controlled system (e.g. controlability, robustness, stability)

[8] To specific families of properties and programs

System analysis & verification by control engineers



Abstractions: program $\longrightarrow$ imprecise, system $\longrightarrow$ precise

– The controler model is a rough abstraction of the control program:

  - Continuous, not discrete
  - Limited to control laws
  - Does not take into account fault-tolerance to failures and computer-related system dependability.

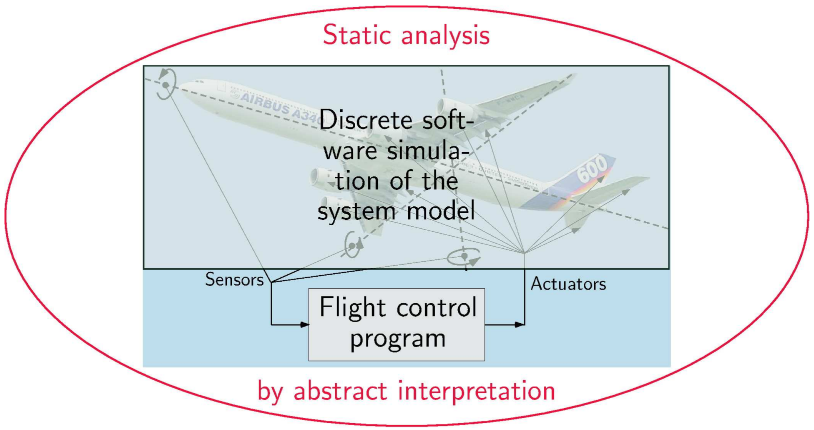
– In theory, SDP-based search of system invariants (Lyapunov-like functions) can be used to prove reachability and inevitability properties

– Problems to scale up (e.g. over long periods of time)

– In practice, the system/controler model is explored by discrete simulations (testing)

Exploring new avenues in static analysis

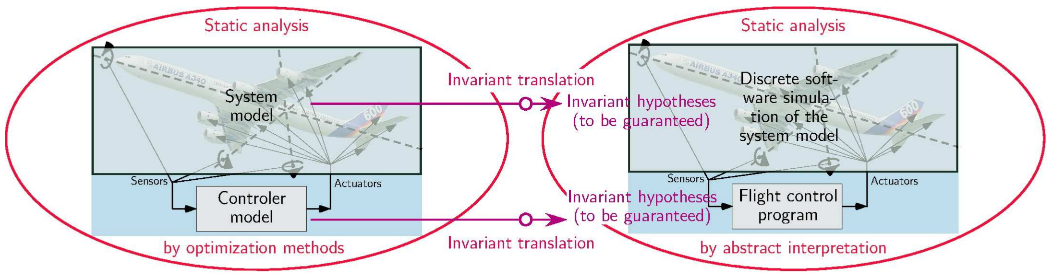
System analysis & verification, Avenue 1



Abstractions: program → precise, system → precise

– Exhaustive (contrary to current simulations)
– Traditional abstractions (e.g. polyhedral abstraction with widening) seem to be too imprecise
– Currently exploring new abstractions (issued from control theory like ellipsoidal calculus using SDP)
– Prototype implementation in construction!

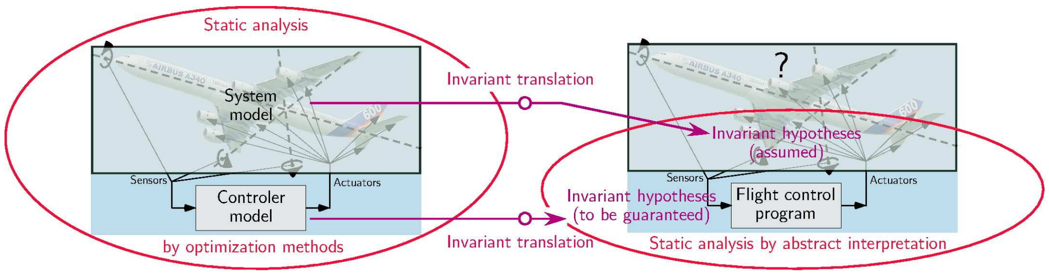
System analysis & verification, Avenue 2



Abstractions: program → precise, system → precise

– Example of invariant translation: ellipsoidal $\longrightarrow$ polyhedral[9]
– The static analysis is easier on the system/controller model using continuous optimization methods
– The translated invariants can be checked for the system simulator/control program (easier than invariant discovery)
– Should scale up since these complex invariants are relevant to a small part of the control program only

System analysis & verification, Avenue 3



Abstractions: program $\rightarrow$ precise, system $\rightarrow$ precise

[9] For which floating point computations can be taken into account

– The invariant hypotheses on the controlled system are assumed to be true
– It remains to perform the control program analysis under these hypothesis
– The results can then be checked on the whole system (as in case 2, but now using refined invariants on the control program!)
– Iterating this process leads to *static analysis by refinement of specifications*

Conclusion

Scientific and technologic objective

To develop formal tools to answer questions about software:

– from control model design to software implementation,

– for a wide range of design and software properties,

which would be general enough to benefit all software-intensive industries, and can be adapted to specific application domains.

THE END, THANK YOU

More references at URL www.di.ens.fr/~cousot
www.astree.ens.fr.

References

[3] www.astree.ens.fr [5, 6, 7, 8, 9, 10, 11, 12]

[4] P. Cousot. *Méthodes itératives de construction et d'approximation de points fixes d'opérateurs monotones sur un treillis, analyse sémantique de programmes.* Thèse d'État ès sciences mathématiques, Université scientifique et médicale de Grenoble, Grenoble, France, 21 March 1978.

[5] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Design and implementation of a special-purpose static program analyzer for safety-critical real-time embedded software. *The Essence of Computation: Complexity, Analysis, Transformation. Essays Dedicated to Neil D. Jones*, LNCS 2566, pp. 85–108. Springer, 2002.

[6] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. *PLDI'03*, San Diego, pp. 196–207, ACM Press, 2003.

[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 *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, NY, USA.

[PACJM '79] P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. Pacific Journal of Mathematics 82(1):43–57 (1979).

[POPL '78] 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, NY, U.S.A.

[POPL '79] 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, NY, U.S.A.

[POPL '92] P. Cousot and R. Cousot. Inductive Definitions, Semantics and Abstract Interpretation. In Conference Record of the 19th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Programming Languages, pages 83–94, Albuquerque, New Mexico, 1992. ACM Press, New York, U.S.A.

[FPCA '95] P. Cousot and R. Cousot. Formal Language, Grammar and Set-Constraint-Based Program Analysis by Abstract Interpretation. In *SIGPLAN/SIGARCH/WG2.8 7th Conference on Functional Programming and Computer Architecture, FPCA'95*. La Jolla, California, U.S.A., pages 170–181. ACM Press, New York, U.S.A., 25-28 June 1995.

[POPL '97] P. Cousot. Types as Abstract Interpretations. In Conference Record of the 24th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Programming Languages, pages 316–331, Paris, France, 1997. ACM Press, New York, U.S.A.

[POPL '00] 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, Mass., January 2000. ACM Press, New York, NY.

[POPL '02] P. Cousot and R. Cousot. Systematic Design of Program Transformation Frameworks by Abstract Interpretation. In *Conference Record of the Twentyninth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 178–190, Portland, Oregon, January 2002. ACM Press, New York, NY.

[TCS 277(1–2) 2002] P. Cousot. Constructive Design of a Hierarchy of Semantics of a Transition System by Abstract Interpretation. *Theoretical Computer Science* 277(1–2):47–103, 2002.

[TCS 290(1) 2002] P. Cousot and R. Cousot. Parsing as abstract interpretation of grammar semantics. *Theoret. Comput. Sci.*, 290:531–544, 2003.

[Manna's festschrift '03] P. Cousot. Verification by Abstract Interpretation. *Proc. Int. Symp. on Verification – Theory & Practice – Honoring Zohar Manna's 64th Birthday*, N. Dershowitz (Ed.), Taormina, Italy, June 29 – July 4, 2003. Lecture Notes in Computer Science, vol. 2772, pp. 243–268. © Springer-Verlag, Berlin, Germany, 2003.

[7] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The ASTRÉE analyser. *ESOP 2005*, Edinburgh, LNCS 3444, pp. 21–30, Springer, 2005.

[8] J. Feret. Static analysis of digital filters. *ESOP'04*, Barcelona, LNCS 2986, pp. 33—-48, Springer, 2004.

[9] J. Feret. The arithmetic-geometric progression abstract domain. In *VMCAI'05*, Paris, LNCS 3385, pp. 42–58, Springer, 2005.

[10] Laurent Mauborgne & Xavier Rival. Trace Partitioning in Abstract Interpretation Based Static Analyzers. *ESOP'05*, Edinburgh, LNCS 3444, pp. 5–20, Springer, 2005.

[11] A. Miné. A New Numerical Abstract Domain Based on Difference-Bound Matrices. *PADO'2001*, LNCS 2053, Springer, 2001, pp. 155–172.

[12] A. Miné. Relational abstract domains for the detection of floating-point run-time errors. *ESOP'04*, Barcelona, LNCS 2986, pp. 3—17, Springer, 2004.

[POPL '04] P. Cousot and R. Cousot. An Abstract Interpretation-Based Framework for Software Watermarking. In *Conference Record of the Thirtyfirst Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 173–185, Venice, Italy, January 14-16, 2004. ACM Press, New York, NY.

[DPG-ICALP '05] M. Dalla Preda and R. Giacobazzi. Semantic-based Code Obfuscation by Abstract Interpretation. In Proc. 32nd Int. Colloquium on Automata, Languages and Programming (ICALP'05 – Track B). LNCS, 2005 Springer-Verlag. July 11-15, 2005, Lisboa, Portugal. To appear.

[EMSOFT '01] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm. Reliable and precise WCET determination for a real-life processor. *ESOP (2001)*, LNCS 2211, 469–485.

[RT-ESOP '04] F. Ranzato and F. Tapparo. Strong Preservation as Completeness in Abstract Interpretation. ESOP 2004, Barcelona, Spain, March 29 - April 2, 2004, D.A. Schmidt (Ed), LNCS 2986, Springer, 2004, pp. 18–32.