Abstract Interpretation of Computations

Patrick COUSOT
École Normale Supérieure
45 rue d’Ulm
75230 Paris cedex 05, France
Patrick.Cousot@ens.fr
www.di.ens.fr/~cousot

Workshop on Robustness, Abstractions and Computations
University of Pennsylvania, Philadelphia
March 28, 2004
Talk Outline

- A few elements of abstract interpretation (10 mn) .................................................. 3
- Applications of abstract interpretation (1 mn) ...... 28
- Application to the verification of embedded, real-time, synchronous, safety super-critical software (8 mn) .................................................. 31
- Examples of abstractions (5 mn) ......................... 45
- Conclusion (1 mn) ............................................. 54
A Few Elements of Abstract Interpretation

Reference

A Model of Computer Programs

- **Syntax**: a well-founded set of programs \( \langle \mathbb{P}, \prec \rangle \) where \( \prec \) is the “strict immediate subcomponent” relation;

- **Semantics of** \( P \in \mathbb{P} \):
  - **Semantic domain**: a complete lattice/cpo \( \langle \mathcal{D}[P], \sqsubseteq, \bot, \sqcup \rangle \)
  - **Compositional Fixpoint Semantics**:

  \[
  S\![P] \overset{\text{def}}{=} \text{lfp}_{\bot} \mathcal{F}[P] \left( \prod_{P' \prec P} S\![P'] \right)
  \]

  \( \text{lfp}_{\bot} f \) is the limit of \( X^0 = \bot, X^{\delta+1} = f(X^{\delta}), X^\lambda = \sqcup \beta < \lambda X^\lambda \), \( \lambda \) limit ordinal, if any. Existence e.g. monotony (by Tarski).
Example: Syntax of Programs

\[ X \] \hspace{0.5cm} \text{variables } X \in X
\[ T \] \hspace{0.5cm} \text{types } T \in T
\[ E \] \hspace{0.5cm} \text{arithmetic expressions } E \in E
\[ B \] \hspace{0.5cm} \text{boolean expressions } B \in B
\[ D ::= T X ; \] \hspace{0.5cm} \text{declarations } D \in D, \ \text{vars}(D) = \{X\}
\[ | \ T X ; D' \] \hspace{0.5cm} X \notin \text{vars}(D'), \ \text{vars}(D) = \{X\} \cup \text{vars}(D')
\[ C ::= X = E ; \] \hspace{0.5cm} \text{commands } C \in C \ (E < C)
\[ | \ \text{while } B C' \] \hspace{0.5cm} (B < C, C' < C)
\[ | \ \text{if } B C' \] \hspace{0.5cm} (B < C, C' < C)
\[ | \ \text{if } B C' \ \text{else } C'' \] \hspace{0.5cm} (B < C, C' < C, C'' < C)
\[ | \ \{ \ C_1 \ldots C_n \}, (n \geq 0) \] \hspace{0.5cm} (C_1 < C, \ldots, C_n < C)
\[ P ::= D C \] \hspace{0.5cm} \text{program } P \in P \ (C < P)
Example: Concrete Semantic Domain of Programs

Reachability properties:

\[ \Sigma[D \ C] \overset{\text{def}}{=} \Sigma[D] \]
\[ \Sigma[T \ X \ ;] \overset{\text{def}}{=} \{X\} \rightarrow T \]
\[ \Sigma[T \ X \ ; \ D] \overset{\text{def}}{=} (\{X\} \rightarrow T) \cup \Sigma[D] \]

\[ \mathcal{D}[P] \overset{\text{def}}{=} \wp(\Sigma[P]) \]
\[ \subset \overset{\text{def}}{=} \subseteq \]
\[ \bot \overset{\text{def}}{=} \emptyset \]
\[ \cup \overset{\text{def}}{=} \cup \]

states \( \rho \)

(\( \rho(X) \) is the value of \( X \))

sets of states

implication

false

disjunction
Example: Concrete Semantics of Programs (Reachability)

\[ S[X = E;] R \overset{\text{def}}{=} \{ \rho[X \leftarrow \mathcal{E}[E]\rho] \mid \rho \in R \cap \text{dom}(E) \} \]
\[ \rho[X \leftarrow v](X) \overset{\text{def}}{=} v, \quad \rho[X \leftarrow v](Y) \overset{\text{def}}{=} \rho(Y) \]
\[ S[\text{if } B C'] R \overset{\text{def}}{=} S[C'](\mathcal{B}[B]R) \cup \mathcal{B}[\neg B] R \]
\[ \mathcal{B}[B] R \overset{\text{def}}{=} \{ \rho \in R \cap \text{dom}(B) \mid B \text{ holds in } \rho \} \]
\[ S[\text{if } B C' \text{ else } C''] R \overset{\text{def}}{=} S[C'](\mathcal{B}[B]R) \cup S[C''](\mathcal{B}[\neg B] R) \]
\[ S[\text{while } B C'] R \overset{\text{def}}{=} \text{let } \mathcal{W} = \text{if } \rho_0 \subseteq \mathcal{L} \chi . R \cup S[C'](\mathcal{B}[B]\chi) \text{ in } (\mathcal{B}[\neg B]\mathcal{W}) \]
\[ S[\{}\} R \overset{\text{def}}{=} R \]
\[ S[\{C_1 \ldots C_n\}] R \overset{\text{def}}{=} S[C_n] \circ \ldots \circ S[C_1] \quad n > 0 \]
\[ S[D C] R \overset{\text{def}}{=} S[C](\Sigma[D]) \quad \text{(uninitialized variables)} \]

Not computable (undecidability).
Abstraction

A reasoning/computation such that:

- only some properties can be used;
- the properties that can be used are called “abstract”;
- so, the (other concrete) properties must be approximated by the abstract ones;
Abstract Properties

• **Abstract Properties**: a set $\mathcal{A} \subseteq \varphi(\Sigma)$ of properties of interest (the only one which can be used to approximate others).

**Direction of Approximation**

• **Approximation from above**: approximate $\overline{P}$ by $\overline{P}$ such that $\overline{P} \subseteq \overline{P}$;

• **Approximation from below**: approximate $\underline{P}$ by $\underline{P}$ such that $\underline{P} \subseteq \underline{P}$ (dual).
Best Abstraction

- We require that all concrete property $P \in \wp(\Sigma)$ have a best abstraction $P \in A$:

  $P \subseteq \bigwedge \{P' \in A : (P \subseteq P') \Rightarrow (P \subseteq P')\}$

- So, by definition of the greatest lower bound/meet $\bigwedge$:

  $P = \bigwedge \{P' \in A \mid P \subseteq P\} \in A$

(Otherwise see [JLC ’92].)

---

Reference

Moore Family

- This hypothesis that any concrete property \( P \in \wp(\Sigma) \) has a best abstraction \( P \in \mathcal{A} \) implies that:
  \[ \mathcal{A} \text{ is a Moore family} \]
  i.e. it is closed under intersection \( \cap \):
  \[ \forall S \subseteq \mathcal{A} : \cap S \in \mathcal{A} \]

- In particular \( \cap \emptyset = \Sigma \in \mathcal{A} \) is “I don’t know”.

Example of Moore Family-Based Abstraction
Closure Operator Induced by an Abstraction

The map \( \rho \bar{\mathcal{A}} \) mapping a concrete property \( P \in \wp(\Sigma) \) to its best abstraction \( \rho \bar{\mathcal{A}}(P) \) in \( \bar{\mathcal{A}} \):

\[
\rho \bar{\mathcal{A}}(P) = \bigcap \{ P \in \bar{\mathcal{A}} \mid P \subseteq P \}
\]

is a closure operator:

- extensive,
- idempotent,
- isotone/monotonic;

such that \( P \in \bar{\mathcal{A}} \iff P = \rho \bar{\mathcal{A}}(P) \)

hence \( \bar{\mathcal{A}} = \rho \bar{\mathcal{A}}(\wp(\Sigma)) \).
Example of Closure Operator-Based Abstraction
The Lattice of Abstract Interpretations

- The set of all possible abstractions that is of all upper closure operators on the complete lattice

\[ \langle \mathcal{D}[P], \sqsubseteq, \bot, \top, \sqcup, \sqcap \rangle \]

is a complete lattice

\[ \langle \text{uco}(\mathcal{D}[P] \hookrightarrow \mathcal{D}[P]), \sqsubseteq, \lambda x \cdot x, \lambda x \cdot \top, \lambda R \cdot \text{uco}(\sqcup R), \sqcap \rangle \]

- The meet of abstractions called the reduced product \( \left( \sqcap_{i \in \Delta} \rho_i \right) \) is that most abstract abstraction more precise than all \( \rho_i \), \( i \in \Delta \)
Galois Connection Between Concrete and Abstract Properties

- For closure operators \( \rho \), we have:

\[
\rho(P) \subseteq \rho(P') \iff P \subseteq \rho(P')
\]

written:

\[
\langle \wp(\Sigma), \subseteq \rangle \xleftarrow{\rho} \langle \wp(\wp(\Sigma)), \subseteq \rangle
\]

where \( \rho \) is the identity and:

\[
\langle \wp(\Sigma), \subseteq \rangle \xleftarrow{\gamma} \langle \mathcal{D}, \sqsubseteq \rangle
\]

means that \( \langle \alpha, \gamma \rangle \) is a **Galois connection**:

\[
\forall P \in \wp(\Sigma), P \in \mathcal{D} : \alpha(P) \sqsubseteq P \iff P \subseteq \gamma(P);
\]

- A Galois connection defines a closure operator \( \rho = \alpha \circ \gamma \), hence a best abstraction.
Example of Galois Connection-Based Abstraction
Example: abstract semantic domain of programs

\[ \langle D^\#, [P], \sqsubseteq, \bot, \sqcup \rangle \]

such that:

\[ \langle D, \subseteq \rangle \leftrightarrow \frac{\gamma}{\alpha} \langle D^\#, [P], \subseteq \rangle \]

hence \( \langle D^\#, [P], \sqsubseteq, \bot, \sqcup \rangle \) is a complete lattice such that \( \bot = \alpha(\emptyset) \) and \( \sqcup X = \alpha(\sqcup \gamma(X)) \)
Abstract domain

$F \# \rightarrow \alpha$

Concrete domain

$\gamma \rightarrow F$

\[\langle P, \subseteq \rangle \xleftrightarrow{\gamma} \langle Q, \sqsubseteq \rangle \Rightarrow\]

\[
\langle P \xrightarrow{\text{mon}} P, \subseteq \rangle \xleftrightarrow{\lambda F \# \cdot \gamma \circ F \# \circ \alpha} \langle Q \xrightarrow{\text{mon}} Q, \sqsubseteq \rangle
\]

\[F \# = \alpha \circ F \circ \gamma \]

i.e. \[F \# = \rho \circ F\]
Approximate Fixpoint Abstraction

Abstract domain

Concrete domain

Approximation relation $\sqsubseteq$

$F \circ \gamma \sqsubseteq \gamma \circ F^\# \Rightarrow \text{lfp } F \sqsubseteq \gamma(\text{lfp } F^\#)$
Example: abstract semantics of programs (reachability)

\[ S^\sharp[X = E;] R \overset{\text{def}}{=} \alpha(\{ \rho[X \leftarrow E] \rho \mid \rho \in \gamma(R) \cap \text{dom}(E) \}) \]

\[ S^\sharp[\text{if } B \ C'] R \overset{\text{def}}{=} S^\sharp[C'](B^\sharp[B] R) \sqcup B^\sharp[\neg B] R \]

\[ B^\sharp[B] R \overset{\text{def}}{=} \alpha(\{ \rho \in \gamma(R) \cap \text{dom}(B) \mid B \text{ holds in } \rho \}) \]

\[ S^\sharp[\text{if } B \ C' \text{ else } C''] R \overset{\text{def}}{=} S^\sharp[C'](B^\sharp[B] R) \sqcup S^\sharp[C''](B^\sharp[\neg B] R) \]

\[ S^\sharp[\text{while } B \ C'] R \overset{\text{def}}{=} \text{let } W = \text{lpf}_{\sqsubseteq} \lambda X. R \sqcup S^\sharp[C'](B^\sharp[B] X) \text{ in } (B^\sharp[\neg B] W) \]

\[ S^\sharp[\{} R \overset{\text{def}}{=} R \]

\[ S^\sharp[\{ C_1 \ldots C_n \}] R \overset{\text{def}}{=} S^\sharp[C_n] \circ \ldots \circ S^\sharp[C_1] \quad n > 0 \]

\[ S^\sharp[D \ C] R \overset{\text{def}}{=} S^\sharp[C](\top) \quad (\text{uninitialized variables}) \]
Convergence Acceleration with Widening

Abstract domain

Concrete domain

Approximation relation ⊇
Widening Operator

A widening operator $\nabla \in \bar{L} \times \bar{L} \mapsto \bar{L}$ is such that:

- **Correctness:**
  - $\forall x, y \in \bar{L} : \gamma(x) \sqsubseteq \gamma(x \nabla y)$
  - $\forall x, y \in \bar{L} : \gamma(y) \sqsubseteq \gamma(x \nabla y)$

- **Convergence:**
  - for all increasing chains $x^0 \sqsubseteq x^1 \sqsubseteq \ldots$, the increasing chain defined by $y^0 = x^0$, $y^{i+1} = y^i \nabla x^{i+1}$, $\ldots$ is not strictly increasing.
Fixpoint Approximation with Widening

**Convergence Theorem:**

The upward iteration sequence with widening:

- $X^0 = \bot$ (infimum)
- $X^{i+1} = X^i$ if $F^\#(X^i) \subseteq X^i$
  
  $= X^i \triangledown F(X^i)$ otherwise

is ultimately stationary and its limit $A$ is a sound upper approximation of $\lfp F^\#$:

\[
\lfp F^\# \subseteq A
\]
Example: Abstract Semantics with Convergence Acceleration

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

\footnote{Note: $\mathcal{F}^\#$ not monotonic!}
Extrapolation by Widening is Essentially Not Monotone

Proof by contradiction:

- Let $\triangledown$ be a widening operator
- Define $x \triangledown' y = \text{if } y \subseteq x \text{ then } x \text{ else } x \triangledown y$
- Assume $x \subseteq y = F(x)$ (during iteration)

then: $x \triangledown' y = x \triangledown y \supseteq y$

$\subseteq \subseteq \subseteq$

$y \triangledown' y = y$

$\Rightarrow x \triangledown y = y$, by antisymmetry!

$\Rightarrow x \triangledown F(x) = F(x)$ during iteration $\Rightarrow$ convergence cannot be enforced with monotone widening (so widening by finite abstraction is less powerful!)
Soundness Theorem

- Convergence by extensivity (no longer monotone)
- Improvement by narrowing [POPL ’77]
- **Soundness Corollary**: any abstract safety proof is valid in the concrete in that:

\[ S^\#[P] \subseteq Q \implies S[P] \subseteq \gamma(Q) \]

- Example: \( \gamma(Q) \) expresses the absence of run-time errors.

Reference

Applications of Abstract Interpretation
Applications of Abstract Interpretation

• **Static Program Analysis** [POPL ’77, ’78, ’79] including **Dataflow Analysis** [POPL ’79, ’00], **Set-based Analysis** [FPCA ’95], **Predicate Abstraction** [Manna’s festschrift ’04]

• **Syntax Analysis** [TCS 290(1) 2002]

• **Hierarchies of Semantics (including Proofs)** [POPL ’92, TCS 277(1–2) 2002]

• **Typing** [POPL ’97]

• **Model Checking** [POPL ’00]
Applications of Abstract Interpretation (Cont’d)

- Program Transformation [POPL ’02]
- Software watermarking [POPL ’02]

All these techniques involve sound approximations that can be formalized by abstract interpretation.
A Practical Application of Abstract Interpretation to the Verification of Safety Critical Embedded Software

Reference


Static Program Analysis

Program

Specification

Generator

System of fixpoint equations/constraints

Solver

(Approximate) solution

Diagnoser

Program checker

Diagnosis
ASTRÉE: A Sound, Automatic, Specializable, Domain-Aware, Parametric, Modular, Efficient and Precise Static Program Analyzer

www.astree.ens.fr

- C programs:
  - structured C programs;
  - no dynamic memory allocation;
  - no recursion.

- Application Domain: safety critical embedded real-time synchronous software for non-linear control of very complex control/command systems.
Concrete Operational Semantics

- International norm of C (ISO/IEC 9899:1999)
- restricted by implementation-specific behaviors depending upon the machine and compiler (e.g. representation and size of integers, IEEE 754-1985 norm for floats and doubles)
- restricted by user-defined programming guidelines (such as no modular arithmetic for signed integers, even though this might be the hardware choice)
- restricted by program specific user requirements (e.g. assert)
Abstract Semantics

- **Reachable states** for the concrete operational semantics
- **Volatile environment** is specified by a *trusted* configuration file.
Implicit Specification: Absence of Runtime Errors

- No violation of the norm of C (e.g. array index out of bounds)
- No implementation-specific undefined behaviors (e.g. maximum short integer is 32767)
- No violation of the programming guidelines (e.g. static variables cannot be assumed to be initialized to 0)
- No violation of the programmer assertions (must all be statically verified).
Example application

- **Primary flight control software** of the Airbus A340/A380 fly-by-wire system

- C program, automatically generated from a proprietary high-level specification

- A340: 132,000 lines, 75,000 LOCs after preprocessing, 10,000 global variables, over 21,000 after expansion of small arrays.
The Class of Considered Periodic Synchronous Programs

declare volatile input, state and output variables;
initialize state and output variables;
loop forever
  - read volatile input variables,
  - compute output and state variables,
  - write to volatile output variables;
  wait_for_clock ();
end loop

• **Requirements:** the only interrupts are clock ticks;
• **Execution time of loop body less than a clock tick** [3].

---

Reference

Characteristics of the ASTRÉE Analyzer

**Static:** compile time analysis (≠ run time analysis Rational Purify, Parasoft Insure++)

**Program Analyzer:** analyzes programs not micromodels of programs (≠ PROMELA in SPIN or Alloy in the Alloy Analyzer)

**Automatic:** no end-user intervention needed (≠ ESC Java, ESC Java 2)

**Sound:** covers the whole state space (≠ MAGIC, CBMC) so never omit potential errors (≠ UNO, CMC from coverity.com) or sort most probable ones (≠ Splint)
Characteristics of the ASTRÉE Analyzer (Cont’d)

**Multiabstraction:** uses many numerical/symbolic abstract domains (≠ symbolic constraints in Bane)

**Infinitary:** all abstractions use infinite abstract domains with widening/narrowing (≠ model checking based analyzers such as VeriSoft, Bandera, Java PathFinder)

**Efficient:** always terminate (≠ counterexample-driven automatic abstraction refinement BLAST, SLAM)

**Specializable:** can easily incorporate new abstractions (and reduction with already existing abstract domains) (≠ general-purpose analyzers PolySpace Verifier)
Characteristics of the ASTRÉE Analyzer (Cont’d)

**Domain-Aware:** knows about control/command (e.g. digital filters) (as opposed to specialization to a mere programming style in C Global Surveyor)

**Parametric:** the precision/cost can be tailored to user needs by options and directives in the code

**Automatic Parametrization:** the generation of parametric directives in the code can be programmed (to be specialized for a specific application domain)
Characteristics of the ASTRÉE Analyzer (Cont’d)

**Modular:** an analyzer instance is built by selection of O-CAML modules from a collection each implementing an abstract domain

**Precise:** few or no false alarm when adapted to an application domain → VERIFIER!
Example of Analysis Session
Benchmarks for the Primary Flight Control Software of the Airbus A340

- Comparative results (commercial software):
  4,200 (false?) alarms,
  3.5 days;

- Our results:
  0 alarm,
  1h20 on 2.8 GHz PC,
  300 Megabytes

→ A world première!
Examples of Abstractions
General-Purpose Abstract Domains: Intervals and Octagons

Intervals:
\[ 1 \leq x \leq 9 \]
\[ 1 \leq y \leq 20 \]

Octagons [4]:
\[ 1 \leq x \leq 9 \]
\[ x + y \leq 78 \]
\[ 1 \leq y \leq 20 \]
\[ x - y \leq 03 \]

Difficulties: many global variables, IEEE 754 floating-point arithmetic (in program and analyzer)

Reference


Floating-Point Computations

- Code Sample:

```c
/* float-error.c */
int main () {
    float x, y, z, r;
    x = 1.000000019e+38;
    y = x + 1.0e21;
    z = x - 1.0e21;
    r = y - z;
    printf("%f\n", r);
} % gcc float-error.c % ./a.out 0.000000
```

```c
/* double-error.c */
int main () {
    double x; float y, z, r;
    /* x = ldexp(1.,50)+ldexp(1.,26); */
    x = 1125899973951488.0;
    y = x + 1;
    z = x - 1;
    r = y - z;
    printf("%f\n", r);
} % gcc double-error.c % ./a.out 134217728.000000
```

\[(x + a) - (x - a) \neq 2a\]
Clock Abstract Domain for Counters

- Code Sample:

```c
R = 0;
while (1) {
    if (I)
        { R = R+1; }
    else
        { R = 0; }
    T = (R>=n);
    wait_for_clock();
}
```

- Output T is true iff the volatile input I has been true for the last n clock ticks.
- The clock ticks every s seconds for at most h hours, thus R is bounded.
- To prove that R cannot overflow, we must prove that R cannot exceed the elapsed clock ticks (impossible using only intervals).

- Solution:
  - We add a phantom variable clock in the concrete user semantics to track elapsed clock ticks.
  - For each variable X, we abstract three intervals: X, X+clock, and X-clock.
  - If X+clock or X-clock is bounded, so is X.
Boolean Relations for Boolean Control

- Code Sample:

```c
/* boolean.c */
typedef enum {F=0,T=1} BOOL;
BOOL B;
void main () {
    unsigned int X, Y;
    while (1) {
        ...
        B = (X == 0);
        ...
        if (!B) {
            Y = 1 / X;
        }
        ...
    }
}
```

The boolean relation abstract domain is parameterized by the height of the decision tree (an analyzer option) and the abstract domain at the leaves.
Control Partitionning for Case Analysis

- Code Sample:

```c
/* trace_partitionning.c */
void main() {
    float t[5] = {-10.0, -10.0, 0.0, 10.0, 10.0};
    float c[4] = {0.0, 2.0, 2.0, 0.0};
    float d[4] = {-20.0, -20.0, 0.0, 20.0};
    float x, r;
    int i = 0;

    ... found invariant -100 ≤ x ≤ 100 ...

    while ((i < 3) && (x >= t[i+1])) {
        i = i + 1;
    }
    r = (x - t[i]) * c[i] + d[i];
}
```

Control point partitionning:

Trace partitionning:
Ellipsoid Abstract Domain for Filters

- 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.
Reference

The main loop invariant

A textual file over 4.5 Mb with

- 6,900 boolean interval assertions \((x \in [0; 1])\)
- 9,600 interval assertions \((x \in [a; b])\)
- 25,400 clock assertions \((x + \text{clk} \in [a; b] \land x - \text{clk} \in [a; b])\)
- 19,100 additive octagonal assertions \((a \leq x + y \leq b)\)
- 19,200 subtractive octagonal assertions \((a \leq x - y \leq b)\)
- 100 decision trees
- 60 ellipse invariants, etc . . .

involving over 16,000 floating point constants (only 550 appearing in the program text) \(\times 75,000\) LOCs.
Conclusion
Conclusion

- Most applications of abstract interpretation tolerate a small rate (typically 5 to 15%) of false alarms:
  - Program transformation → do not optimize,
  - Typing → reject some correct programs, etc,
  - WCET analysis → overestimate;
- Some applications require no false alarm at all:
  - Program verification.
- Theoretically possible [SARA ’00], practically feasible [PLDI ’03]

Reference


THE END, THANK YOU

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