« A Tutorial on Abstract Interpretation »

Patrick Cousot
École normale supérieure
45 rue d’Ulm
75230 Paris cedex 05, France

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

VMCAI’05 Industrial Day
Static analysis by abstract interpretation
Example of static analysis (input)

\[ n := n0; \]
\[ i := n; \]
\[ \text{while (} i <> 0 \text{) do} \]
\[ \quad j := 0; \]
\[ \quad \text{while (} j <> i \text{) do} \]
\[ \quad \quad j := j + 1 \]
\[ \quad \text{od;} \]
\[ \quad i := i - 1 \]
\[ \text{od} \]
Example of static analysis (output)

\{n0 \geq 0\}
\hspace{1em} n := n0;
\{n0 = n, n0 \geq 0\}
\hspace{1em} i := n;
\{n0 = i, n0 = n, n0 \geq 0\}
\hspace{1em} while (i \neq 0) do
\{n0 = n, i \geq 1, n0 \geq i\}
\hspace{1em} j := 0;
\{n0 = n, j = 0, i \geq 1, n0 \geq i\}
\hspace{1em} while (j \neq i) do
\{n0 = n, j \geq 0, i \geq j+1, n0 \geq i\}
\hspace{1em} j := j + 1
\{n0 = n, j \geq 1, i \geq j, n0 \geq i\}
\hspace{1em} od;
\{n0 = n, i = j, i \geq 1, n0 \geq i\}
\hspace{1em} i := i - 1
\{i+1 = j, n0 = n, i \geq 0, n0 \geq i+1\}
\hspace{1em} od
\{n0 = n, i = 0, n0 \geq 0\}
Example of static analysis (safety)

n0 must be initially nonnegative
(otherwise the program does not terminate properly)

\{n0=0\}
\text{n := n0;}
\{n0=n, n0=0\}
\text{i := n;}
\{n0=i, n0=n, n0=0\}
while (i <> 0) do
\{n0=n, i=1, n0=i\}
\text{j := 0;}
\{n0=n, j=0, i=1, n0=i\}
while (j <> i) do
\{n0=n, j=0, i=1, n0=i\}
\text{j := j + 1}
\{n0=n, j=1, i=j, n0=i\}
\text{od;}
\{n0=n, i=j, i=1, n0=i\}
\text{i := i - 1}
\{i+1=j, n0=n, i=0, n0=i+1\}
\text{od}
\{n0=n, i=0, n0=0\}
Static analysis by abstract interpretation

**Verification**: define and prove automatically a property of the possible behaviors of a complex computer program (example: program semantics);

**Abstraction**: the reasoning/calculus can be done on an abstraction of these behaviors dealing only with those elements of the behaviors related to the considered property;

**Theory**: abstract interpretation.
Example of static analysis

**Verification**: absence of runtime errors;
**Abstraction**: polyhedral abstraction (affine inequalities);
**Theory**: abstract interpretation.
A very informal introduction to the principles of abstract interpretation
Semantics

The *concrete semantics* of a program formalizes (is a mathematical model of) the set of all its possible executions in all possible execution environments.
Graphic example: Possible behaviors

\[ x(t) \]

Possible trajectories
Undecidability

– The concrete mathematical semantics of a program is an “t-infinitesimal” mathematical object, *not computable*;
– All non-trivial questions on the concrete program semantics are *undecidable*.

Example: termination
– Assume `termination(P)` would always terminates and returns `true` *iff* *P* always terminates on all input data;
– The following program yields a contradiction

\[ P \equiv \text{while} \ \text{termination}(P) \ \text{do} \ \text{skip} \ \text{od}. \]
Graphic example: Safety properties

The safety properties of a program express that no possible execution in any possible execution environment can reach an erroneous state.
Graphic example: Safety property

Forbidden zone

Possible trajectories
Safety proofs

– A safety proof consists in proving that the intersection of the program concrete semantics and the forbidden zone is empty;

– Undecidable problem (the concrete semantics is not computable);

– Impossible to provide completely automatic answers with finite computer resources and neither human interaction nor uncertainty on the answer.¹

¹ e.g. probabilistic answer.
Test/debugging

- consists in considering a subset of the possible executions;
- not a correctness proof;
- absence of coverage is the main problem.
Graphic example: Property test/simulation

\[ x(t) \]

Forbidden zone

Error !!!

Possible trajectories

Test of a few trajectories

VMCAI’05 Industrial Day, Paris, France, January 20, 2005
Abstract interpretation

- consists in considering an abstract semantics, that is to say a superset of the concrete semantics of the program;
- hence the abstract semantics covers all possible concrete cases;
- correct: if the abstract semantics is safe (does not intersect the forbidden zone) then so is the concrete semantics
Graphic example: Abstract interpretation

Forbidden zone

Abstraction of the trajectories

Possible trajectories

$x(t)$

$t$
Formal methods

Formal methods are abstract interpretations, which differ in the way to obtain the abstract semantics:

– “model checking”:
  - the abstract semantics is given manually by the user;
  - in the form of a finitary model of the program execution;
  - can be computed automatically, by techniques relevant to static analysis.
– “deductive methods”:
  - the abstract semantics is specified by verification conditions;
  - the user must provide the abstract semantics in the form of inductive arguments (e.g. invariants);
  - can be computed automatically by methods relevant to static analysis.
– “static analysis”: the abstract semantics is computed automatically from the program text according to pre-defined abstractions (that can sometimes be tailored automatically/manually by the user).
Required properties of the abstract semantics

- **sound** so that no possible error can be forgotten;
- **precise** enough (to avoid false alarms);
- as **simple/abstract** as possible (to avoid combinatorial explosion phenomena).
Graphic example: The most abstract correct and precise semantics

$x(t)$

Forbidden zone

Possible trajectories

$t$
Graphic example: Erroneous abstraction — I

Forbidden zone

Error !!!

Erroneous trajectory abstraction

x(t)

t

Possible trajectories
Graphic example: **Erroneous abstraction — II**

- Forbidden zone
- Error !!!
- Erroneous trajectory abstraction
- Possible trajectories

\[ x(t) \]
Graphic example: Imprecision $\Rightarrow$ false alarms

Forbidden zone

False alarm

Imprecise trajectory abstraction

Possible trajectories

$x(t)$

t
Abstract domains

Standard abstractions

- that serve as a basis for the design of static analyzers:
  - abstract program data,
  - abstract program basic operations;
  - abstract program control (iteration, procedure, concurrency, ...);
- can be parametrized to allow for manual adaptation to the application domains.
Graphic example: Standard abstraction by intervals

\[ x(t) \]

Forbidden zone

False alarms

Imprecise trajectory abstraction by intervals

Possible trajectories

\[ t \]
Graphic example: A more refined abstraction

Forbidden zone

Refinement of intervals

Possible trajectories

\( x(t) \)
A very informal introduction to static analysis algorithms
Standard operational semantics
Standard semantics

- Start from a **standard operational semantics** that describes formally:
  - **states** that is data values of program variables,
  - **transitions** that is elementary computation steps;
- Consider **traces** that is successions of states corresponding to executions described by transitions (possibly infinite).
Graphic example: Small-steps transition semantics
Example: Small-steps transition semantics of an assignment

```plaintext
int x;
...
l:
    x := x + 1;
l':

\{ l : x = \nu \rightarrow l' : x = \nu + 1 \mid \nu \in [\text{min\_int, max\_int} - 1] \}
\cup \{ l : x = \text{max\_int} \rightarrow l' : x = \Omega \} \quad \text{(runtime error)}
```
Example: Small-steps transition semantics of a loop

\begin{align*}
\text{l1:} & \quad x := 1; \\
\text{l2:} & \quad \text{while } x < 10 \text{ do} \\
\text{l3:} & \quad x := x + 1 \\
\text{l4:} & \quad \text{od} \\
\text{l5:} & \quad \text{end}
\end{align*}

\begin{align*}
\text{l1:} & \quad \ldots \quad l1 : x = -1 \\
\text{l1:} & \quad \ldots \quad l1 : x = 0 \quad \rightarrow \quad l2 : x = 1 \\
\text{l1:} & \quad \ldots \quad l1 : x = 1 \\
\text{l1:} & \quad \ldots \quad l1 : \ldots \\
\text{l2:} & \quad \ldots \quad l2 : x = 1 \rightarrow l3 : x = 1 \\
\text{l3:} & \quad \ldots \quad l3 : x = 1 \rightarrow l4 : x = 2 \\
\text{l4:} & \quad \ldots \quad l4 : x = 2 \rightarrow l3 : x = 2 \\
\text{l3:} & \quad \ldots \quad l3 : x = 2 \rightarrow l4 : x = 3 \\
\text{l4:} & \quad \ldots \quad l4 : x = 10 \rightarrow l5 : x = 10
\end{align*}
Example: Trace semantics of loop

```
l1: x := 1;
l2: while x < 10 do
l3: x := x + 1
l4: od
l5:
```

\[
\begin{align*}
\text{l1:} & \quad x := 1; \\
\text{l2:} & \quad \text{while } x < 10 \text{ do} \\
\text{l3:} & \quad x := x + 1 \\
\text{l4:} & \quad \text{od} \\
\text{l5:} & \\
\end{align*}
\]
Transition systems

- $\langle S, t \rightarrow \rangle$ where:
  - $S$ is a set of states/vertices/...
  - $t \rightarrow \in \wp(S \times S)$ is a transition relation/set of arcs/...
Collecting semantics in fixpoint form
Collecting semantics

- consider all traces simultaneously;
- collecting semantics:
  - sets of states that describe data values of program variables on all possible trajectories;
  - set of states transitions that is simultaneous elementary computation steps on all possible trajectories;
Graphic example: sets of states

$x(t)$

Possible discrete trajectories
Graphic example: set of states transitions
Example: Reachable states of a transition system
Reachable states in fixpoint form

\[ F(X) = \mathcal{I} \cup \{ s' \mid \exists s \in X : s \xrightarrow{t} s' \} \]

\[ \mathcal{R} = \text{lfp} \subseteq F = \bigcup_{n=0}^{+\infty} F^n(\emptyset) \]

where

\[ f^0(x) = x \]

\[ f^{n+1}(x) = f(f^n(x)) \]
Example of fixpoint iteration for reachable states \( \text{lfp}_\emptyset \subseteq \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\} \)
Example of fixpoint iteration for reachable states \( \text{fixpoint} \subseteq \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\} \)
Example of fixpoint iteration
for reachable states $lfp_{\emptyset} \subseteq \lambda X. \mathcal{I} \cup \{s' | \exists s \in X : s \xrightarrow{t} s'\}$
Example of fixpoint iteration for reachable states \( \text{ifp}_\emptyset \subseteq \lambda X. \mathcal{I} \cup \{s' | \exists s \in X : s \xrightarrow{t} s'\} \)
Example of fixpoint iteration for reachable states $\text{lfp}_\emptyset \subseteq \lambda X. \mathcal{I} \cup \{s' \mid \exists s \in X : s \xrightarrow{t} s'\}$
Abstraction by Galois connections
Abstracting sets (i.e. properties)

- Choose an abstract domain, replacing sets of objects (states, traces, ...) $S$ by their abstraction $\alpha(S)$
- The abstraction function $\alpha$ maps a set of concrete objects to its abstract interpretation;
- The inverse concretization function $\gamma$ maps an abstract set of objects to concrete ones;
- Forget no concrete objects: (abstraction from above) $S \subseteq \gamma(\alpha(S))$. 
Interval abstraction $\alpha$

\[ x : [1, 99], y : [2, 77] \]
Interval concretization $\gamma$

\[ x : [1, 99], y : [2, 77] \]
The abstraction $\alpha$ is monotone

\[
\{ x : [33, 89], y : [48, 61] \} \sqsubseteq \{ x : [1, 99], y : [2, 90] \}
\]

\[
X \subseteq Y \Rightarrow \alpha(X) \sqsubseteq \alpha(Y)
\]
The concretization $\gamma$ is monotone

\[ x : [33, 89], y : [48, 61] \]
\[ \subseteq \]
\[ x : [1, 99], y : [2, 90] \]

\[ X \subseteq Y \Rightarrow \gamma(X) \subseteq \gamma(Y) \]
The $\gamma \circ \alpha$ composition is extensive

$X \subseteq \gamma \circ \alpha(X)$
The $\alpha \circ \gamma$ composition is reductive

\[ \{ x : [1, 99], y : [2, 77] \} \subseteq \{ x : [1, 99], y : [2, 77] \} \]

\[ \alpha \circ \gamma(Y) = \subseteq Y \]
Correspondance between concrete and abstract properties

- The pair \( \langle \alpha, \gamma \rangle \) is a Galois connection:

\[
\langle \mathcal{G}(S), \subseteq \rangle \leftrightarrow_{\alpha} \gamma \leftrightarrow \langle \mathcal{D}, \sqsubseteq \rangle
\]

- \( \langle \mathcal{G}(S), \subseteq \rangle \leftrightarrow_{\alpha} \gamma \leftrightarrow \langle \mathcal{D}, \sqsubseteq \rangle \) when \( \alpha \) is onto (equivalently \( \alpha \circ \gamma = 1 \) or \( \gamma \) is one-to-one).
Galois connection

\[ \langle D, \subseteq \rangle \leftrightarrow_{\alpha} \gamma \leftrightarrow \langle \overline{D}, \subseteq \rangle \]

iff

\[ \forall x, y \in D : x \subseteq y \implies \alpha(x) \subseteq \alpha(y) \]
\[ \land \forall x, y \in \overline{D} : x \subseteq y \implies \gamma(x) \subseteq \gamma(y) \]
\[ \land \forall x \in D : x \subseteq \gamma(\alpha(x)) \]
\[ \land \forall y \in \overline{D} : \alpha(\gamma(y)) \subseteq x \]

iff

\[ \forall x \in D, \overline{y} \in \overline{D} : \alpha(x) \subseteq \overline{y} \iff x \subseteq \gamma(y) \]
Graphic example: Interval abstraction

Possible discrete trajectories

Interval with spurious states
Graphic example: Abstract transitions

\[ x(t) \]

Possible discrete trajectories

Interval transition
Example: Interval transition semantics of assignments

```
int x;
...
l:
    x := x + 1;
```

\[ l' : \{ l : x \in [\ell, h] \rightarrow l' : x \in [\ell + 1, \min(h + 1, \max\_int)] \cup \{ \Omega \mid h = \max\_int \} \mid \ell \leq h \} \]

where \([\ell, h] = \emptyset\) when \(h < \ell\).
Function abstraction

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

i.e. \[ F^\# = \rho \circ F \]

\[
\begin{align*}
\langle P, \subseteq \rangle & \xrightarrow{\gamma} \langle Q, \sqsubseteq \rangle \\
\langle P \xrightarrow{\text{mon}} P, \subseteq \rangle & \xrightarrow{\lambda F^\# \cdot \gamma \circ F^\# \circ \alpha} \langle Q \xrightarrow{\text{mon}} Q, \sqsubseteq \rangle
\end{align*}
\]
Example: Set of traces to trace of intervals abstraction

Set of traces:

\[ \alpha_1 \downarrow \]

Trace of sets:

\[ \alpha_2 \downarrow \]

Trace of intervals
Example: Set of traces to reachable states abstraction

Set of traces:

\[ \alpha_1 \downarrow \]

Trace of sets:

\[ \alpha_3 \downarrow \]

Reachable states
Composition of Galois Connections

The composition of Galois connections:

\[ \langle L, \leq \rangle \overset{\gamma_1}{\underset{\alpha_1}{\leftrightarrow}} \langle M, \sqsubseteq \rangle \]

and:

\[ \langle M, \sqsubseteq \rangle \overset{\gamma_2}{\underset{\alpha_2}{\leftrightarrow}} \langle N, \preceq \rangle \]

is a Galois connection:

\[ \langle L, \leq \rangle \overset{\gamma_1 \circ \gamma_2}{\underset{\alpha_2 \circ \alpha_1}{\leftrightarrow}} \langle N, \preceq \rangle \]
Abstract semantics in fixpoint form
Graphic example: traces of sets of states in fixpoint form

$x(t)$

Possible discrete trajectories
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form
Graphic example: *traces of sets of states in fixpoint form*
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form

\[ x(t) \]
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form

$$x(t)$$

Possible discrete trajectories
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form

\[ x(t) \]

Possible discrete trajectories
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form
Graphic example: traces of sets of states in fixpoint form.

$x(t)$
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form

Graph showing traces of intervals over time (x(t) vs. t) with possible discrete trajectories.
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form

\[ x(t) \]
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Graphic example: traces of intervals in fixpoint form
Approximate fixpoint abstraction

Abstract domain

Concrete domain

\[ \alpha(\text{lfp } F) \sqsubseteq \text{lfp } F^\# \]
approximate/exact fixpoint abstraction

Exact Abstraction:
\[ \alpha(\text{lfp } F) = \text{lfp } F'\]

Approximate Abstraction:
\[ \alpha(\text{lfp } F) \Box \text{lfp } F'\]
Convergence acceleration by widening/narrowing
Graphic example: upward iteration with widening

\[ x(t) \]
Graphic example: upward iteration with widening
Graphic example: upward iteration with widening
Graphic example: upward iteration with widening

Interval transition with widening

Possible discrete trajectories

$\mathbf{x(t)}$
Graphic example: stability of the upward iteration
Convergence acceleration with widening
Widening operator

A widening operator \( \nabla \in \overline{L} \times \overline{L} \mapsto \overline{L} \) is such that:

- **Correctness:**
  - \( \forall x, y \in \overline{L} : \gamma(x) \subseteq \gamma(x \nabla y) \)
  - \( \forall x, y \in \overline{L} : \gamma(y) \subseteq \gamma(x \nabla y) \)

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

The upward iteration sequence with widening:

\[ \hat{X}^0 = \perp \text{ (infimum) } \]
\[ \hat{X}^{i+1} = \hat{X}^i \quad \text{if } F(\hat{X}^i) \subseteq \hat{X}^i \]
\[ = \hat{X}^i \triangledown F(\hat{X}^i) \quad \text{otherwise} \]

is ultimately stationary and its limit \( \hat{A} \) is a sound upper approximation of \( \lfp F \):

\[ \lfp F \subseteq \hat{A} \]
Interval widening

- $\overline{L} = \{ \bot \} \cup \{ [l, u] \mid l, u \in \mathbb{Z} \cup \{-\infty\} \land u \in \mathbb{Z} \cup \{\} \land l \leq u \}$

- The **widening** extrapolates unstable bounds to infinity:
  \[
  \bot \nabla X = X \\
  X \nabla \bot = X \\
  [l_0, u_0] \nabla [l_1, u_1] = \begin{cases} 
  -\infty & \text{if } l_1 < l_0 \\
  l_0, & \text{if } u_1 > u_0 \\
  +\infty & \text{else}
  \end{cases}
  \]

Not monotone. For example $[0, 1] \subseteq [0, 2]$ but $[0, 1] \nabla [0, 2] = [0, +\infty] \not\subseteq [0, 2] = [0, 2] \nabla [0, 2]$
Example: Interval analysis (1975)

Program to be analyzed:

\[
\begin{align*}
x & := 1; \\
1: & \\
& \text{while } x < 10000 \text{ do} \\
2: & \\
& x := x + 1 \\
3: & \\
& \text{od}; \\
4: & 
\end{align*}
\]
Example: Interval analysis (1975)

Equations (abstract interpretation of the semantics):

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

x := 1;
1:
while x < 10000 do
2:
    x := x + 1
3:
od;
4:
Example: Interval analysis (1975)

Resolution by chaotic increasing iteration:

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

\[
\begin{align*}
X_1 &= \emptyset \\
X_2 &= \emptyset \\
X_3 &= \emptyset \\
X_4 &= \emptyset
\end{align*}
\]

\[
x := 1; \\
1: \quad \text{while } x < 10000 \text{ do} \\
2: \quad x := x + 1 \\
3: \quad \text{od;} \\
4: \\
\]
Example: Interval analysis (1975)

Increasing chaotic iteration:

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

\[
\begin{align*}
x &:= 1; \\
\text{1:} & \quad \text{while } x < 10000 \text{ do} \\
\text{2:} & \quad x := x + 1 \\
\text{3:} & \quad \text{od}; \\
\text{4:}
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration:

\[
\begin{align*}
x &:= 1; \\
\text{while } x < 10000 \text{ do} \\
& \quad x := x + 1 \\
\text{od;}
\end{align*}
\]

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= (X_1 \cup X_3) \cap (-\infty, 9999] \\
X_3 &= X_2 \oplus [1, 1] \\
X_4 &= (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration:

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

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= (X_1 \cup X_3) \cap [-\infty, 9999] \\
X_3 &= X_2 \oplus [1, 1] \\
X_4 &= (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration:

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

\[ \begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, 2] \\
X_3 &= [2, 2] \\
X_4 &= \emptyset
\end{align*} \]
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence!

\[
\begin{align*}
x &:= 1; \\
1: & \text{ while } x < 10000 \text{ do} \\
2: & \quad x := x + 1 \\
3: & \quad \text{od;} \\
4: &
\end{align*}
\]

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= (X_1 \cup X_3) \cap [-\infty, 9999] \\
X_3 &= X_2 \oplus [1, 1] \\
X_4 &= (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence !!

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

\[
\begin{align*}
x &= 1; \\
1: & \text{ while } x < 10000 \text{ do} \\
2: & \quad x := x + 1 \\
3: & \text{ od;} \\
4: & \text{}
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence !!!

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

\[
\begin{align*}
x &:= 1; \\
1: &\quad \text{while } x < 10000 \text{ do} \\
2: &\quad \text{\hspace{1cm} } x := x + 1 \\
3: &\quad \text{od;} \\
4: &
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence !!!!

\begin{align*}
    x &:= 1; \\
1: &\quad \text{while } x < 10000 \text{ do} \\
2: &\quad x := x + 1 \\
3: &\quad \text{od;}
\end{align*}

\begin{align*}
    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{align*}
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence !!!!!

\[
x := 1;
\]
\[
1: \quad \text{while } x < 10000 \text{ do}
\]
\[
2: \quad x := x + 1
\]
\[
3: \quad \text{od;}
\]
\[
4: \quad \begin{cases}
X_1 = [1, 1] \\
X_2 = (X_1 \cup X_3) \cap [-\infty, 9999] \\
X_3 = X_2 \oplus [1, 1] \\
X_4 = (X_1 \cup X_3) \cap [10000, +\infty]
\end{cases}
\]

\[
\begin{cases}
X_1 = [1, 1] \\
X_2 = [1, 4] \\
X_3 = [2, 5] \\
X_4 = \emptyset
\end{cases}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence !!!!!!

\[
\begin{align*}
\text{x} &:= 1; \\
\text{1:} &\quad \text{while } \text{x} < 10000 \text{ do} \\
\text{2:} &\quad \quad \text{x} := \text{x} + 1 \\
\text{3:} &\quad \text{od;} \\
\text{4:} &
\end{align*}
\]

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

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, 5] \\
X_3 &= [2, 5] \\
X_4 &= \emptyset
\end{align*}
\]
Example: Interval analysis (1975)

Increasing chaotic iteration: convergence !!!!!!!

\[
\begin{align*}
x & := 1; \\
\text{while } x < 10000 \text{ do} \\
x & := x + 1 \\
\text{od;}
\end{align*}
\]

\[
\begin{align*}
X_1 & = [1, 1] \\
X_2 & = (X_1 \cup X_3) \cap [-\infty, 9999] \\
X_3 & = X_2 \oplus [1, 1] \\
X_4 & = (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]
Example: Interval analysis (1975)

Convergence speed-up by widening:

\[
\begin{align*}
x &:= 1; \\
1: &\quad \text{while } x < 10000 \text{ do} \\
2: &\quad x := x + 1 \\
3: &\quad \text{od;} \\
4: &\quad \text{od;}
\end{align*}
\]

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= (X_1 \cup X_3) \cap [\infty, 9999] \\
X_3 &= X_2 \oplus [1, 1] \\
X_4 &= (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, +\infty] \quad \Leftarrow \text{widening} \\
X_3 &= [2, 6] \\
X_4 &= \emptyset
\end{align*}
\]
Example: Interval analysis (1975)

Decreasing chaotic iteration:

\[
\begin{align*}
x & := 1; \\
\text{while } x < 10000 \text{ do} & \\
\text{end while;}
\end{align*}
\]

\[
\begin{align*}
X_1 & = [1, 1] \\
X_2 & = (X_1 \cup X_3) \cap [-\infty, 9999] \\
X_3 & = X_2 \oplus [1, 1] \\
X_4 & = (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]
Example: Interval analysis (1975)

Decreasing chaotic iteration:

\[
\begin{align*}
&x := 1; \\
&1: \\
&\quad \text{while } x < 10000 \text{ do} \\
&\quad \quad x := x + 1 \\
&3: \\
&\quad \text{od}; \\
&4: \\
&\end{align*}
\]

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

\[
\begin{align*}
\begin{cases}
X_1 &= [1, 1] \\
X_2 &= [1, 9999] \\
X_3 &= [2, +\infty] \\
X_4 &= \emptyset
\end{cases}
\]
Example: Interval analysis (1975)

Decreasing chaotic iteration:

\[
\begin{align*}
x & := 1; \\
\text{while } x < 10000 \text{ do} \\
x & := x + 1 \\
\text{od;}
\end{align*}
\]

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= (X_1 \cup X_3) \cap [\neg \infty, 9999] \\
X_3 &= X_2 \oplus [1, 1] \\
X_4 &= (X_1 \cup X_3) \cap [10000, +\infty]
\end{align*}
\]
Example: Interval analysis (1975)

Final solution:

\[
\begin{align*}
x & := 1; \\
1: & \quad \text{while } x < 10000 \text{ do} \\
2: & \quad x := x + 1 \\
3: & \quad \text{od;} \\
4: &
\end{align*}
\]

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

\[
\begin{cases}
X_1 = [1, 1] \\
X_2 = [1, 9999] \\
X_3 = [2, +10000] \\
X_4 = [+10000, +10000]
\end{cases}
\]
Example: Interval analysis (1975)

Result of the interval analysis:

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

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

\[
\begin{aligned}
X_1 &= [1, 1] \\
X_2 &= [1, 9999] \\
X_3 &= [2, +10000] \\
X_4 &= [+10000, +10000]
\end{aligned}
\]
Example: Interval analysis (1975)

Checking absence of runtime errors with interval analysis:

\[
\begin{align*}
\text{x} & := 1; \\
1: & \{x = 1\} \\
\text{while } & x < 10000 \text{ do} \\
2: & \{x \in [1, 9999]\} \\
\text{\quad x} & := x + 1 \\
3: & \{x \in [2, +10000]\} \\
\text{\quad od;} \\
4: & \{x = 10000\}
\end{align*}
\]
Refinement of abstractions
Approximations of an \( [\text{in}]\)finite set of points:

\[
\{\ldots, \langle 19, 77 \rangle, \ldots, \\
\langle 20, 03 \rangle, \ldots \}
\]
Approximations of an [in]finite set of points:
from above

\[ \{ \ldots, \langle 19, 77 \rangle, \ldots, \langle 20, 03 \rangle, \langle ?, ? \rangle, \ldots \} \]

From Below: dual² + combinations.

---

² Trivial for finite states (liveness model-checking), more difficult for infinite states (variant functions).
Effective computable approximations of an infinite set of points; Signs

\[ \begin{align*}
  x &\geq 0 \\
  y &\geq 0
\end{align*} \]

---

Effective computable approximations of an [in]finite set of points; Intervals

\[ \begin{align*}
  x & \in [19, 77] \\
  y & \in [20, 03]
\end{align*} \]

---

Effective computable approximations of an [in]finite set of points; Octagons

\[
\begin{align*}
1 & \leq x \leq 9 \\
x + y & \leq 77 \\
1 & \leq y \leq 9 \\
x - y & \leq 99
\end{align*}
\]

Effective computable approximations of an [in]finite set of points; Polyhedra

\[
\begin{align*}
19x + 77y & \leq 2004 \\
20x + 03y & \geq 0
\end{align*}
\]

---

Effective computable approximations of an [in]finite set of points; Simple congruences

\[
\begin{align*}
\begin{cases}
  x &= 19 \mod 77 \\
  y &= 20 \mod 99
\end{cases}
\end{align*}
\]

---

Effective computable approximations of an [in]finite set of points; Linear congruences

\[
\begin{align*}
\begin{cases}
1x + 9y &= 7 \mod 8 \\
2x - 1y &= 9 \mod 9
\end{cases}
\end{align*}
\]

---

Effective computable approximations of an [in]finite set of points; Trapezoidal linear congruences

\[
\begin{align*}
1x + 9y &\in [0, 77] \mod 10 \\
2x - 1y &\in [0, 99] \mod 11
\end{align*}
\]

---

Refinement of iterates
Graphic example: Refinement required by false alarms
Graphic example: Partitionning
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening

$x(t)$

Possible discrete trajectories
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening

\[ x(t) \]
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening
Graphic example: partitionned upward iteration with widening

$\mathbf{x}(t)$

Possible discrete trajectories
Graphic example: safety verification

Forbidden zone

Possible discrete trajectories
Examples of partitionnings

- **sets of control states**: attach local information to program points instead of global information for the whole program/procedure/loop
- **sets of data states**:
  - case analysis (test, switches)
- **fixpoint iterates**:
  - widening with threshold set
Interval widening with threshold set

- The threshold set $T$ is a finite set of numbers (plus $+\infty$ and $-\infty$),
- $[a, b] \nabla_T [a', b'] = \begin{cases} 
  \text{if } a' < a \text{ then } \max\{l \in T \mid l \leq a'\} & \text{else } a, \\
  \text{if } b' > b \text{ then } \min\{h \in T \mid h \geq b'\} & \text{else } b 
\end{cases}$.

Examples (intervals):
- sign analysis: $T = \{-\infty, 0, +\infty\}$;
- strict sign analysis: $T = \{-\infty, -1, 0, +1, +\infty\}$;
- $T$ is a parameter of the analysis.
Combinations of abstractions
Forward/reachability analysis
Backward/ancestry analysis
Iterated forward/backward analysis
Example of iterated forward/backward analysis

Arithmetical mean of two integers $x$ and $y$:

$$\{x \geq y\}$$
while (x <> y) do
$$\{x \geq y + 2\}$$
    x := x - 1;
$$\{x \geq y + 1\}$$
y := y + 1
$$\{x \geq y\}$$
  od
$$\{x = y\}$$

Necessarily $x \geq y$ for proper termination
Example of iterated forward/backward analysis

Adding an auxiliary counter $k$ decremented in the loop body and asserted to be null on loop exit:

\[
\begin{align*}
&\{x=y+2k,x\geq y\} \\
&\text{while (x <> y) do} \\
&\quad \{x=y+2k,x\geq y+2\} \\
&\quad k := k - 1; \\
&\quad \{x=y+2k+2,x\geq y+2\} \\
&\quad x := x - 1; \\
&\quad \{x=y+2k+1,x\geq y+1\} \\
&\quad y := y + 1 \\
&\quad \{x=y+2k,x\geq y\} \\
&\text{od} \\
&\{x=y,k=0\} \\
&\text{assume (k = 0)} \\
&\{x=y,k=0\}
\end{align*}
\]

Moreover the difference of $x$ and $y$ must be even for proper termination.
Bibliography
Seminal papers


Recent surveys


Conclusion
Theoretical applications of abstract interpretation

- **Static Program Analysis** [POPL ’77,78,79] including **Data-flow Analysis** [POPL ’79,00], **Set-based Analysis** [FPCA ’95], etc
- **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]
- **Program Transformation** [POPL ’02]
- **Software watermarking** [POPL ’04]
Practical applications of abstract interpretation

- **Program analysis and manipulation**: a small rate of false alarms is acceptable
  - **AiT**: worst case execution time – Christian Ferdinard

- **Program verification**: no false alarms is acceptable
  - **TVLA**: A system for generating abstract interpreters
    – Mooly Sagiv
  - **Astrée**: verification of absence of run-time errors – Laurent Mauborgne
Industrial applications of abstract interpretation

- Both to Program analysis and verification
- Experience with the industrial use of abstract interpretation-based static analysis tools – Jean Souyris (Airbus France)
THE END

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