What is static analysis by abstract interpretation?
Example of static analysis (input)

\begin{align*}
n &:= n_0; \\
i &:= n; \\
\text{while} \ (i <> 0) \ \text{do} \\
&\quad j := 0; \\
&\quad \text{while} \ (j <> i) \ \text{do} \\
&\quad \quad j := j + 1 \\
&\quad \quad od; \\
&\quad i := i - 1 \\
&\quad od
\end{align*}
Example of static analysis (output)

\{n0 \geq 0\}
\begin{align*}
  & n \ := \ n0; \\
  & \{n0=n, n0 \geq 0\} \\
  & i \ := \ n; \\
  & \{n0=i, n0=n, n0 \geq 0\} \\
  \text{while} \ (i \not= 0) \ \text{do} \\
  & \{n0=n, i \geq 1, n0 \geq i\} \\
  & j \ := \ 0; \\
  & \{n0=n, j=0, i \geq 1, n0 \geq i\} \\
  \text{while} \ (j \not= i) \ \text{do} \\
  & \{n0=n, j \geq 0, i \geq j+1, n0 \geq i\} \\
  & \quad j \ := \ j + 1 \\
  & \{n0=n, j \geq 1, i \geq j, n0 \geq i\} \\
  \text{od;} \\
  & \{n0=n, i=j, i \geq 1, n0 \geq i\} \\
  & i \ := \ i - 1 \\
  & \{i+1=j, n0=n, i \geq 0, n0 \geq i+1\} \\
  \text{od} \\
  & \{n0=n, i=0, n0 \geq 0\} \\
\end{align*}
Example of static analysis (safety)

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

```
{n0>=0}
  n := n0;
{n0=n,n0>=0}
  i := n;
{n0=i,n0=n,n0>=0}
while (i <> 0 ) do
  {n0=n,i>=1,n0>=i}
    j := 0;
  {n0=n,j=0,i>=1,n0>=i}
    while (j <> i) do
      {n0=n,j>=0,i>=j+1,n0>=i}
        j := j + 1
      {n0=n,j>=1,i>=j,n0>=i}
    od;
  {n0=n,i=j,i>=1,n0>=i}
  i := i - 1
  {i+1=j,n0=n,i>=0,n0>=i+1}
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;

**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.
Potential impact of runtime errors

- 50% of the security attacks on computer systems are through buffer overruns\(^1\)!
- Embedded computer system crashes easily result from overflows of various kinds.

\(^1\) See for example the Microsoft Security Bulletins MS02-065, MS04-011, etc.
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

Possible trajectories
Undecidability

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

Example: Kurt Gödel argument on termination
- Assume $\text{termination}(P)$ would always terminates and returns true iff $P$ always terminates on all input data;
- The following program yields a contradiction

$$P \equiv \text{while } \text{termination}(P) \text{ do skip od}.$$
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\(^2\).

\(^2\) 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

Forbidden zone

Test of a few trajectories

Error !!!

Possible trajectories
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

\[ x(t) \]

- Forbidden zone
- Abstraction of the trajectories
- Possible trajectories
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 predefined 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: Erroneous abstraction — I
Graphic example: Erroneous abstraction — II

Forbidden zone

Error!!!

Possible trajectories

Erroneous trajectory abstraction

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

Imprecise trajectory abstraction

Forbidden zone

False alarm

Possible trajectories
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

Imprecise trajectory abstraction by intervals

False alarms

Possible trajectories
Graphic example: A more refined abstraction

Forbidden zone

Refinement of intervals

Possible trajectories

$x(t)$
A very informal introduction to static analysis algorithms
Trace semantics
Trace semantics

- Consider (possibly infinite) traces that is series of states corresponding to executions described by discrete transitions;
- The collection of all such traces, starting from the initial states, is the trace semantics.
Graphic example: Small-steps transition semantics
Trace semantics, intuition

Initial states

Intermediate states

Final states of the finite traces

Infinite traces

Discrete time
Prefix trace semantics
Prefixes of a finite trace

A finite trace

Its finite prefixes

discrete time
Prefixes of an infinite trace

Initial states

Intermediate states

An infinite trace

Its finite prefixes

Discrete time
Prefix trace semantics

Trace semantics: maximal finite and infinite behaviors

Prefix trace semantics: finite prefixes of the maximal behaviors
Abstraction

This is an abstraction. For example:

Trace semantics: \( \{ a^n b \mid n \geq 0 \} \)
Prefix trace semantics: \( \{ a^n \mid n \geq 0 \} \cup \{ a^n b \mid n \geq 0 \} \)

Is there of possible behavior with infinitely many successive \( a \)?

- Trace semantics: no
- Prefix trace semantics: I don’t know
Prefix trace semantics in fixpoint form
Least **Fixpoint** Prefix Trace Semantics

Prefixes = \{ \bullet \mid \bullet \text{ is an initial state} \} \\
\bigcup \{ \cdots \bullet \bullet \bullet \mid \bullet \cdots \bullet \in \text{Prefixes} \} \\
& \text{& } \bullet \bullet \bullet \text{ is a transition step} \}

- In general, the equation \( \text{Prefixes} = F(\text{Prefixes}) \) may have multiple solutions;
- Choose the least one for subset inclusion \( \subseteq \).
- **Abstractions** of this equation lead to effective iterative analysis algorithms.
Collecting semantics
Collecting semantics

- Collect all states that can appear on some trace at any given discrete time:

![Diagram showing collecting semantics]

Trace semantics

Collecting semantics

discrete time
Collecting abstraction

– This is an abstraction. Does the red trace exists?
  Trace semantics: no, collecting semantics: I don’t know.
Graphic example: collecting semantics

$x(t)$

Possible discrete trajectories
Collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form

$x(t)$

Possible discrete trajectories
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form

$\frac{dx}{dt}$
Graphic example: collecting semantics in fixpoint form

$\mathbf{x}(t)$

Possible discrete trajectories
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form

Possible discrete trajectories
Graphic example: collecting semantics in fixpoint form

$x(t)$
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form

$x(t)$

Possible discrete trajectories
Graphic example: collecting semantics in fixpoint form
Graphic example: collecting semantics in fixpoint form

\[ x(t) \]

Possible discrete trajectories
Interval Abstraction
(in iterative 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

\[ x(t) \]
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

\[ x(t) \]
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
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

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

\[ X \subseteq Y \Rightarrow \alpha(X) \subseteq \alpha(Y) \]
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

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

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

\[
\{ x : [1, 99], y : [2, 77] \} = /\subseteq Y
\]
Correspondance between concrete and abstract properties

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

$$\langle \wp(S), \subseteq \rangle \leftrightarrow_{\alpha} \gamma \leftrightarrow \langle D, \sqsubseteq \rangle$$

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

\[ \langle \mathcal{D}, \subseteq \rangle \iff \frac{\gamma}{\alpha} \iff \langle \overline{\mathcal{D}}, \sqsubseteq \rangle \]

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

\[ \text{iff} \]
\[ \forall x \in \mathcal{D}, \overline{y} \in \overline{\mathcal{D}} : \alpha(x) \sqsubseteq \overline{y} \iff x \subseteq \gamma(y) \]
Example: Set of traces to trace of intervals abstraction

Set of traces:

\[ \mathbf{\alpha}_1 \downarrow \]

Trace of sets:

\[ \mathbf{\alpha}_2 \downarrow \]

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:

\[ \langle L, \leq \rangle \xleftarrow{\gamma_1} \xrightarrow{\alpha_1} \langle M, \sqsubseteq \rangle \]

and:

\[ \langle M, \sqsubseteq \rangle \xleftarrow{\gamma_2} \xrightarrow{\alpha_2} \langle N, \leq \rangle \]

is a Galois connection:

\[ \langle L, \leq \rangle \xleftarrow{\gamma_1 \circ \gamma_2} \xrightarrow{\alpha_2 \circ \alpha_1} \langle N, \leq \rangle \]
Convergence acceleration by widening/narrowing
Graphic example: upward iteration with widening

Initial states

Possible discrete trajectories

$x(t)$

$t$
Graphic example: upward iteration with widening

Interval transition

Possible discrete trajectories
Graphic example: upward iteration with widening

Interval transition with widening

Possible discrete trajectories
Graphic example: upward iteration with widening
Graphic example: stability of the upward iteration
Interval widening

- $\bar{L} = \{ \bot \} \cup \{ [\ell, u] \mid \ell, u \in \mathbb{Z} \cup \{-\infty\} \land u \in \mathbb{Z} \cup \{\} \land \ell \leq u \}$
- The widening extrapolates unstable bounds to infinity:
  $\bot \nabla X = X$
  $X \nabla \bot = X$
  $[\ell_0, u_0] \nabla [\ell_1, u_1] = \begin{cases} -\infty & \text{if } \ell_1 < \ell_0 \text{ then} \\ \ell_0, & \text{else} \\ +\infty & \text{if } u_1 > u_0 \text{ then} \\ u_0, & \text{else} \end{cases}$

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

Program to be analyzed:

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

Equations (abstract interpretation of the semantics):

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

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

Resolution by chaotic increasing iteration:

\[ x := 1; \]
1:

\[ \text{while } x < 10000 \text{ do} \]

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

2:

\[ x := x + 1 \]

3:

\[ \text{od;} \]

4:

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

Increasing chaotic iteration:

\[
x := 1;
\]

\[
1:
\]

\[
\text{while } x < 10000 \text{ do}
\]

\[
2:
x := x + 1
\]

\[
3:
\]

\[
\text{od;}
\]

\[
4:
\]

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

Increasing chaotic iteration:

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

\[ X_1 = [1, 1] \]
\[ X_2 = [1, 1] \]
\[ X_3 = \emptyset \]
\[ X_4 = \emptyset \]
Example: Interval analysis (1975)

Increasing chaotic iteration:

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

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

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

Increasing chaotic iteration:

\[
X_1 = [1, 1] \\
X_2 = (X_1 \cup X_3) \cap [\infty, 9999] \\
X_3 = X_2 \oplus [1, 1] \\
X_4 = (X_1 \cup X_3) \cap [10000, \infty]
\]

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

Increasing chaotic iteration: convergence!

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

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, 2] \\
X_3 &= [2, 3] \\
X_4 &= \emptyset
\end{align*}
\]

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

Increasing chaotic iteration: convergence !!

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

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

Increasing chaotic iteration: convergence !!!

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

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, 3] \\
X_3 &= [2, 4] \\
X_4 &= \emptyset
\end{align*}
\]

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

Increasing chaotic iteration: convergence !!!!

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

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

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

Increasing chaotic iteration: convergence !!!!!

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

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

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

Increasing chaotic iteration: convergence !!!!!!

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

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

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

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

```plaintext
1:
   x := 1;
2:
   while x < 10000 do
5:
      x := x + 1
6:
   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)

Convergence speed-up by widening:

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

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

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

Decreasing chaotic iteration:

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

Decreasing chaotic iteration:

```plaintext
Example: Interval analysis (1975)

Decreasing chaotic iteration:

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

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

Decreasing chaotic iteration:

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

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, 9999] \\
X_3 &= [2, +10000] \\
X_4 &= \emptyset
\end{align*}
\]

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

Final solution:

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

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

Result of the interval analysis:

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

\[
\begin{align*}
X_1 &= [1, 1] \\
X_2 &= [1, 9999] \\
X_3 &= [2, +10000] \\
X_4 &= [+10000, +10000]
\end{align*}
\]

\[
\begin{align*}
x := 1; \\
1: \{x = 1\} \\
\text{while } x < 10000 \text{ do} \\
2: \{x \in [1, 9999]\} \\
x := x + 1 \\
3: \{x \in [2, +10000]\} \\
\text{od;} \\
4: \{x = 10000\}
\end{align*}
\]
Example: Interval analysis (1975)

Checking absence of runtime errors with interval analysis:

\[
\begin{align*}
x & := 1; \\
1: \ & \{x = 1\} \\
\quad \text{while } x < 10000 \text{ do} \\
2: \ & \{x \in [1, 9999]\} \\
\quad x & := x + 1 \\
3: \ & \{x \in [2, +10000]\} \\
\quad \text{od;} \\
4: \ & \{x = 10000\}
\end{align*}
\]

← no overflow
Refinement of abstractions
Approximations of an [in]finite set of points:

\{\ldots, (19, 77), \ldots, (20, 03), \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: dual$^3$ + combinations.

---

$^3$ Trivial for finite states (liveness model-checking), more difficult for infinite states (variant functions).
Effective computable approximations of an [in]finite 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

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

\[ \text{A. Miné. A New Numerical Abstract Domain Based on Difference-Bound Matrices. PADO '2001.}
\]
\[ \text{LNCS 2053, pp. 155–172. Springer 2001. See the The Octagon Abstract Domain Library on}
\]
\[ \text{http://www.di.ens.fr/~mine/oct/} \]
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

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

---

Effective computable approximations of an infinite set of points; Linear congruences

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

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

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

\(^{10}\) F. Masdupuy. Array Operations Abstraction Using Semantic Analysis of Trapezoid Congruences. ACM ICS ’92.
Refinement of iterates
Graphic example: Refinement required by false alarms
Graphic example: Partitionning

\[ x(t) \]

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

Possible discrete trajectories
Graphic example: partitionned upward iteration with widening

Possible discrete trajectories
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

$x(t)$

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

Forbidden zone

Possible discrete trajectories

\[ x(t) \]

\[ t \]
Interval widening with threshold set

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

- 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

[Diagram of a graph with nodes and edges]

---

91
Iterated forward/backward analysis
Example of iterated forward/backward analysis

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

\{
\begin{align*}
\{x &\geq y\} \\
\text{while } (x &\not\equiv y) \text{ do} \\
&\{x = y + 2\} \\
&x := x - 1; \\
&\{x = y + 1\} \\
&y := y + 1 \\
&\{x = y\}
\end{align*}
\}

{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:

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

$$\{x=y, k=0\}$$
assume (k = 0)
$$\{x=y, k=0\}$$

Moreover the difference of $x$ and $y$ must be even for proper termination.
Applications of abstract interpretation
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]
Industrial applications of abstract interpretation

- **Program analysis and manipulation**: a small rate of false alarms is acceptable
  - AiT: worst case execution time
  - StackAnalyzer: stack usage analysis

- **Program verification**: no false alarms is acceptable
  - TVLA: A system for generating abstract interpreters
  - Astrée: verification of absence of run-time errors

---

11 applied to the primary flight control software of the Airbus A340/600 and A380 fly-by-wire systems
Seminal papers


Recent surveys


Anticipated Content of Course 16.399: Abstract Interpretation

– Today: an informal overview of abstract interpretation;

– The software verification problem (undecidability, complexity, test, simulation, specification, formal methods (deductive methods, model-checking, static analysis) and their limitations, intuitive notion of approximation, false alarms);

– Mathematical foundations (naive set theory, first order classical logic, lattice theory, fixpoints);
– **Semantics of programming languages** (abstract syntax, operational semantics, inductive definitions, example of a simple imperative language, grammar and interpreter of the language, trace semantics);

– **Program specification and manual proofs** (safety properties, Hoare logic, predicate transformers, liveness properties, linear-time temporal logic (LTL));

– **Order-theoretic approximation** (abstraction, closures, Galois connections, fixpoint abstraction, widening, narrowing, reduced product, absence of best approximation, refinement);
- **Principle of static analysis by abstract interpretation** (reachability analysis of a transition system, finite approximation, model-checking, infinite approximation, static analysis, program-based versus language-based analysis, limitations of finite approximations);

- **Design of a generic structural abstract interpreter** (collecting semantics, non-relational and relational analysis, convergence acceleration by wideing/narrowing);

- **Static analysis** (forward reachability analysis, backward analysis, iterated forward/backward analysis, inevitability analysis, termination)