Abstract Interpretation
Static Analysis

A general method for automatic and sound approximation of sw run-time behaviors before the execution

- “before”: statically, without running sw
- “automatic”: sw analyzes sw
- “sound”: all possibilities into account
- “approximation”: cannot be exact
- “general”: for any source language and property
  - C, C++, C#, F#, Java, JavaScript, ML, Scala, Python, JVM, Dalvik, x86, Excel, etc
  - “buffer-overflow?”, “memory leak?”, “type errors?”, “$x = y$ at line 2?”, “memory use $\leq 2K$?”, etc
Abstract Interpretation

• A powerful framework for designing correct static analysis
Abstraction?

- Without abstraction,
  - can’t capture all possible executions
  - can’t terminate

Examples
- Does the program have memory errors? (such as buffer overrun or memory leaks)
- Does the program always terminate?
Abstraction

MAX++

*0

*a++ =
Abstraction

Abstract Interpretation

The static analysis game

\[ \text{MAX}++ \]

\[ \ast\text{a}++ = \ast 0 \]
Abstraction

Abstract Interpretation

The static analysis game

\[ \text{MAX}++ = \]

\[ *a++ = \]

\[ *0 \]
Step 1

- Concrete interpretation with standard semantics
  - **states**: data values of program variables
  - **transitions**: elementary computation steps
  - **traces**: sequences of states corresponding to transitions

*from Patrick Cousot’s slides*
Concrete Interpretation
(Standard Semantics)

*from Patrick Cousot’s slides*
Concrete Interpretation
(Standard Semantics)

Execution Trace :
(Program Points x (Var → Z))+
Concrete Interpretation (Standard Semantics)

Execution Trace:
(Program Points × (Var → Z))^+

Concrete Interpretation (Collecting Semantics)

Partitioned Execution Traces:
Program Points → (Var → 2^Z)
Concrete Interpretation (Collecting Semantics)

*from Patrick Cousot’s slides*
Step 2

• Concrete interpretation with **collecting** semantics

• considering all traces simultaneously

• **set of states**: data values of program variables on all possible trajectories

• **set of transitions**: elementary computation steps on all possible trajectories

*from Patrick Cousot’s slides*
Concrete Interpretation
(Collecting Semantics)

Partitioned Execution Traces:
Program Points $\rightarrow$ (Var $\rightarrow$ $2^\mathbb{Z}$)

1: $x := 0;$
2: $y := 0;$
3: while ($x < 10$) {
4: $x := x + 1;$
5: $y := y + 1;$
6: }
7: print($x$)

The possible values of $x$ and $y$ are
{$1, \ldots, 9$} after executing line 3.
Concrete Interpretation (Collecting Semantics)

Partitioned Execution Traces:
Program Points $\rightarrow$ (Var $\rightarrow$ $2^Z$)

Abstract Interpretation (Abstract Semantics)

Partitioned Execution Traces:
Program Points $\rightarrow$ (Var $\rightarrow$ Interval)
Abstract Interpretation
(Abstract Semantics)
Abstract Interpretation
(Abstract Semantics)

Abstract State:
Program Points \rightarrow (\text{Var} \rightarrow \text{Interval})

1: \quad x := 0;
2: \quad y := 0;
3: \quad \textbf{while} \ (x < 10) \ { 
4: \quad \quad x := x + 1;
5: \quad \quad y := y + 1;
6: \quad }
7: \quad \text{print}(x)
Challenge

1: x := 0;
2: y := 0;
3: while (very complex) {
4:   x := x + 1;
5:   y := y + 1;
6: }
7: print(x)
Concrete Interpretation
(Collecting Semantics)

1: x := 0;
2: y := 0;
3: while (very complex) {
4:   x := x + 1;
5:   y := y + 1;
6: }
7: print(x)
Abstract Interpretation
(Abstract Semantics)

1:  x := 0;                    Abstract State
2:  y := 0;
3:  while (very complex) {
4:    x := x + 1;
5:    y := y + 1;
6:  }
7:  print(x)
Need for Theory

- How to ensure that we soundly approximate real executions?
- How to ensure the termination of analysis?
Abstract Interpretation Framework

real execution \[ [P] = \text{fix } F \in \mathcal{D} \]
abstract execution \[ [\hat{P}] = \text{fix } \hat{F} \in \hat{\mathcal{D}} \]
correctness \[ [P] \approx [\hat{P}] \]
implementation computation of \([\hat{P}]\)

• The framework requires:
  • a relation between \(\mathcal{D}\) and \(\hat{\mathcal{D}}\)
  • a relation between \(F \in \mathcal{D} \rightarrow \mathcal{D}\) and \(\hat{F} \in \hat{\mathcal{D}} \rightarrow \hat{\mathcal{D}}\)

• The framework guarantees:
  • correctness and implementation
  • freedom: any such \(\hat{\mathcal{D}}\) and \(\hat{F}\) are fine.
Abstract Interpretation Framework

real execution \([P] = \text{fix } F \in D\)

abstract execution \([\hat{P}] = \text{fix } \hat{F} \in \hat{D}\)

correctness \([P] \approx [\hat{P}]\)

implementation computation of \([\hat{P}]\)

- The framework requires:
  - a relation between \(D\) and
  - a relation between \(F \in D \rightarrow D\) and \(\hat{F} \in \hat{D} \rightarrow \hat{D}\)

- The framework guarantees:
  - correctness and implementation
  - freedom: any such \(\hat{D}\) and \(\hat{F}\) are fine.

A domain of concrete states (e.g., a set of memories)
\(\text{fix } F = X \text{ in } D \text{ s.t. } F(X) = X.\)

A function corresponding to one-step real execution
Abstract Interpretation Framework

real execution \[ [P] = \text{fix} \ F \in \mathcal{D} \]

abstract execution \[ [\hat{P}] = \text{fix} \ \hat{F} \in \hat{\mathcal{D}} \]

correctness \[ [P] \approx [\hat{P}] \]

implementation computation of \([\hat{P}]\)

• The framework requires:
  • a relation between \(\mathcal{D}\) and \(\hat{\mathcal{D}}\)
  • a relation between \(F \in \mathcal{D} \rightarrow \mathcal{D}\) and \(\hat{F} \in \hat{\mathcal{D}} \rightarrow \hat{\mathcal{D}}\)

• The framework guarantees:
  • correctness and implementation
  • freedom: any such \(\hat{\mathcal{D}}\) and \(\hat{F}\) are fine.
Steps

• Step 1: Define concrete semantics

• Step 2: Define abstract semantics

• Step 3: Compute abstract semantics guaranteeing termination
  • with a finite abstract domain
  • with a infinite abstract domain
Preliminaries for Step 1

• Domain theory

  • Every program’s meaning is an element of CPO,

  • which is a least fixed point of a continuous function : CPO → CPO.

  • Every continuous function has a unique least fixed point.

• See:

Preliminaries for Step 1

- What is
  - CPO?
  - Continuous Functions?
  - Least Fixed Point?
Partial Order

**Definition (Partial Order)**

We say a binary relation $\sqsubseteq$ is a partial order on a set $D$ iff $\sqsubseteq$ is

- reflexive: $\forall p \in D. \ p \sqsubseteq p$
- transitive: $\forall p, q, r \in D. \ p \sqsubseteq q \land q \sqsubseteq r \implies p \sqsubseteq r$
- anti-symmetric: $\forall p, q \in D. \ p \sqsubseteq q \land q \sqsubseteq p \implies p = q$

We call such a pair $(D, \sqsubseteq)$ partially ordered set, or poset.

**Lemma**

*If a partially ordered set $(D, \sqsubseteq)$ has a least element $d$, then $d$ is unique.*
Partial Order

- \{x,z\} \subseteq \{x,y,z\}

- \emptyset = \bot : unique
Definition (Least Upper Bound)

Let \((D, \sqsubseteq)\) be a partially ordered set and let \(Y\) be a subset of \(D\). An upper bound of \(Y\) is an element \(d\) of \(D\) such that

\[
\forall d' \in Y. \ d' \sqsubseteq d.
\]

An upper bound \(d\) of \(Y\) is a least upper bound if and only if \(d \sqsubseteq d'\) for every upper bound \(d'\) of \(Y\). The least upper bound of \(Y\) is denoted by \(\bigcup Y\).

Lemma

If \(Y\) has a least upper bound \(d\), then \(d\) is unique.
LUB

\[ \{x\} \cup \{y\} = \{x,y\} \]

\[ \{x,y\} \cup \{z\} = \{x,y,z\} \]
**Definition (Chain)**

Let \((D, \sqsubseteq)\) be a poset and \(Y\) a subset of \(D\). \(Y\) is called a chain if \(Y\) is totally ordered:

\[
\forall y_1, y_2 \in Y. y_1 \sqsubseteq y_2 \text{ or } y_2 \sqsubseteq y_1.
\]

**Example**

Consider the poset \(\varnothing(\{a, b, c\}, \subseteq)\).

- \(Y_1 = \{\emptyset, \{a\}, \{a, c\}\}\)
- \(Y_2 = \{\emptyset, \{a\}, \{c\}, \{a, c\}\}\)
Chain
# Complete Partial Order (CPO)

**Definition (CPO)**

A poset \((D, \sqsubseteq)\) is a CPO, if every chain \(Y \subseteq D\) has \(\bigcup Y \in D\).

**Definition (Complete Lattice)**

A poset \((D, \sqsubseteq)\) is a complete lattice, if every subset \(Y \subseteq D\) has \(\bigcup Y \in D\).

**Lemma**

If \((D, \sqsubseteq)\) is a CPO, then it has a least element \(\bot\) given by \(\bot = \bigcup \emptyset\).
Fixed Point

Definition (Fixed Point)

Let \((D, \sqsubseteq)\) be a poset. A fixed point of a function \(f : D \to D\) is an element \(d \in D\) such that \(f(d) = d\). We write \(\text{fix}(f)\) for the least fixed point of \(f\), if it exists, such that

- \(f(\text{fix}(f)) = \text{fix}(f)\)
- \(\forall d \in D. \ f(d) = d \implies \text{fix}(f) \sqsubseteq d\)
Continuous Function

Definition (Continuous Functions)
A function \( f : D_1 \to D_2 \) defined on posets \((D_1, \sqsubseteq_1)\) and \((D_2, \sqsubseteq_2)\) is continuous if it is monotone and it preserves least upper bounds of chains:

\[
\bigsqcup f(Y) = f(\bigsqcup Y)
\]

for all non-empty chains \( Y \) in \( D_1 \). If \( f(\bigsqcup Y) = \bigsqcup f(Y) \) holds for the empty chain (that is, \( \bot = f(\bot) \)), then we say that \( f \) is strict.
Step 1: Define Concrete Semantics

- Define a semantic domain $D$, which is a CPO
  - Any increasing chain $d_0 \subseteq d_1 \subseteq \ldots$ in $D$ has a least upper bound $\bigsqcup_{n \geq 0} d_n$ in $D$.
- Define a semantic function $F : D \to D$, which is continuous: for all chains $d_0 \subseteq d_1 \subseteq \ldots$
  \[ F\left(\bigsqcup_{n \geq 0} d_n\right) = \bigsqcup_{n \geq 0} F(d_n).\]
Step 1: Define Concrete Semantics

- Then, the concrete semantics is the least fixed point of semantic function $F : D \rightarrow D$

$$\text{fix} F = \bigsqcup_{i \in \mathbb{N}} F^i(\bot).$$

**Theorem (Kleene Fixed Point)**

Let $f : D \rightarrow D$ be a continuous function on a CPO $D$. Then $f$ has a least fixed point, $\text{fix}(f)$, and

$$\text{fix}(f) = \bigsqcup_{n \geq 0} f^n(\bot)$$

where $f^n(\bot) = \begin{cases} \bot & n = 0 \\ f(f^{n-1}(\bot)) & n > 0 \end{cases}$
Proof

We show the claims of the theorem by showing that $\bigsqcup_{n \geq 0} f^n(\bot)$ exists and it is indeed equivalent to $\text{fix}(f)$. First note that $\bigsqcup_{n \geq 0} f^n(\bot)$ exists because $f^0(\bot) \sqsubseteq f^1(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \ldots$ is a chain. We show by induction that $\forall n \in \mathbb{N}. f^n(\bot) \sqsubseteq f^{n+1}(\bot)$:

- $\bot \sqsubseteq f(\bot)$ ($\bot$ is the least element)
- $f^n(\bot) \sqsubseteq f^{n+1}(\bot) \implies f^{n+1}(\bot) \sqsubseteq f^{n+2}(\bot)$ (monotonicity of $f$)

Now, we show that $\text{fix}(f) = \bigsqcup_{n \geq 0} f^n(\bot)$ in two steps:

- We show that $\bigsqcup_{n \geq 0} f^n(\bot)$ is a fixed point of $f$:

$$f(\bigsqcup_{n \geq 0} f^n(\bot)) = \bigsqcup_{n \geq 0} f(f^n(\bot))$$

  $$= \bigsqcup_{n \geq 0} f^{n+1}(\bot)$$

  $$= \bigsqcup_{n \geq 0} f^n(\bot)$$

  (continuity of $f$)
Proof

- We show that $\bigsqcup_{n \geq 0} f^n(\bot)$ is smaller than all the other fixed points.

Suppose $d$ is a fixed point, i.e., $f(d) = d$. Then,

$$\bigsqcup_{n \geq 0} f^n(\bot) \subseteq d$$

since $\forall n \in \mathbb{N}. f^n(\bot) \subseteq d$:

$$f^0(\bot) = \bot \subseteq d, \quad f^n(\bot) \subseteq d \implies f^{n+1}(\bot) \subseteq f(d) = d.$$  

Therefore, we conclude

$$\text{fix}(f) = \bigsqcup_{n \geq 0} f^n(\bot).$$
Example: Concrete (Collecting) Semantics

• Program as a control flow graph (CFG)
  
  • \((\mathcal{C}, \rightarrow, c_0)\)

  • Each node \(c \in \mathcal{C}\) is with a command \(\text{cmd}(c)\)

  \[
  \text{cmd} \rightarrow \text{skip} \mid x := e
  \]

  \[
  e \rightarrow n \mid x \mid e + e \mid e - e
  \]
Example: Concrete (Collecting) Semantics

- Semantics of commands:

- Memory: $M = \operatorname{Var} \rightarrow \mathbb{Z}$

- Semantics:

$$
[e] : M \rightarrow M
$$

$$
\begin{align*}
\lbrack \text{skip} \rbrack (m) &= m \\
\lbrack x := e \rbrack (m) &= m[x \mapsto \lbrack e \rbrack (s)]
\end{align*}
$$

- Concrete memory states:

$$M = \text{Var} \rightarrow \mathbb{Z}$$

- Concrete semantics:

$$\begin{align*}
\lbrack \text{skip} \rbrack (m) &= m \\
\lbrack x := e \rbrack (m) &= m[x \mapsto \lbrack e \rbrack (s)] \\
\lbrack e \rbrack & : M \rightarrow \mathbb{Z} \\
\lbrack n \rbrack (m) &= n \\
\lbrack x \rbrack (m) &= m(x) \\
\lbrack e_1 + e_2 \rbrack (m) &= \lbrack e_1 \rbrack (m) + \lbrack e_2 \rbrack (m) \\
\lbrack e_1 - e_2 \rbrack (m) &= \lbrack e_1 \rbrack (m) + \lbrack e_2 \rbrack (m)
\end{align*}$$

e.g., $\{x \mapsto 1\} = \{x \mapsto 1\}$
$\{x \mapsto 1\}[x \mapsto 2] = \{x \mapsto 2\}$
Example: Concrete (Collecting) Semantics

- Program states: \( \text{State} = C \times M \)
- A trace \( \sigma \in \text{State}^+ \) is a (partial) execution sequence of the program:

\[
\sigma_0 \in I \land \forall k. \sigma_k \leadsto \sigma_{k+1}
\]

where \( I \subseteq \text{State} \) is the initial program states

\[
I = \{(c_0, m_0)\}
\]

and \( (\leadsto) \subseteq \text{State} \times \text{State} \) is the relation for the one-step execution:

\[
(c_i, s_i) \leadsto (c_j, s_j) \iff c_i \rightarrow c_j \land s_j = [\text{cmd}(c_j)](s_i)
\]
Example: Concrete (Collecting) Semantics

The collecting semantics of program $P$ is defined as the set of all finite traces of the program:

$$[P] = \{ \sigma \in \text{State}^+ \mid \sigma_0 \in I \land \forall k. \sigma_k \sim \sigma_{k+1} \}$$

The semantic domain:

$$D = \wp(\text{State}^+)$$

The semantic function:

$$F : \wp(\text{State}^+) \to \wp(\text{State}^+)$$

$$F(\Sigma) = I \cup \{ \sigma \cdot (c, m) \mid \sigma \in \Sigma \land \sigma \downarrow \sim (c, m) \}$$

Lemma

$$[P] = \text{fix} F.$$
Example: Concrete (Collecting) Semantics

0: ENTRY

1: x := 0

2: y := 0

3: skip

4: x := x + 1

5: y := y + 1

6: skip
Step 2: Define Abstract Semantics

Plan: define an abstraction that captures $\text{fix } F$ by using $\hat{F}$

- Define an abstract domain CPO $\hat{D}$
  - Intuition: $\hat{D}$ is an abstraction of $D$

- Define an abstract semantic function $\hat{F} : \hat{D} \rightarrow \hat{D}$
  - Intuition: $\hat{F}$ is an abstraction of $F$

- $\hat{F}$ must be monotone:
  \[
  \forall \hat{x}, \hat{y} \in \hat{D}. \hat{x} \sqsubseteq \hat{y} \implies \hat{F}(\hat{x}) \sqsubseteq \hat{F}(\hat{y})
  \]
  (or extensive: $\forall x \in \hat{D}. x \sqsubseteq \hat{F}(x)$)
Step 2: Define Abstract Semantics

- Then, static analysis is to compute an upper bound of:

\[ \bigcup_{i \in \mathbb{N}} \hat{F}^i(\bot) \]

- How can we ensure that the result soundly approximate the concrete semantics?
Requirement 1: Galois Connection

$D$ and $\hat{D}$ must be related with Galois-connection:

$$D \xleftarrow{\alpha} \gamma \xrightarrow{\alpha} \hat{D}$$

That is, we have

- **abstraction function**: $\alpha \in D \rightarrow \hat{D}$
  - represents elements in $D$ as elements of $\hat{D}$
- **concretization function**: $\gamma \in \hat{D} \rightarrow D$
  - gives the meaning of elements of $\hat{D}$ in terms of $D$
- $\forall x \in D, \hat{x} \in \hat{D}, \alpha(x) \sqsubseteq \hat{x} \iff x \sqsubseteq \gamma(\hat{x})$
  - $\alpha$ and $\gamma$ respect the orderings of $D$ and $\hat{D}$
Requirement 1: Galois Connection

\[ D \rightarrow \gamma \rightarrow \hat{D} \]

\[ D \rightarrow \alpha \rightarrow \hat{D} \]

\[ \gamma(\hat{x}) \rightarrow \gamma(x) \]

\[ x \rightarrow \alpha(x) \]

\[ \hat{x} \rightarrow \alpha(\hat{x}) \]
Example: Sign Abstraction

\[ D = 2^\mathbb{Z} \]

\[ \mathbb{Z}^+ = \{1, 2, 3, \ldots\} \]

\[ \emptyset \]

\[ \alpha \]

\[ \hat{D} \]

\[ + \]

\[ 0 \]

\[ - \]
Example: Sign Abstraction

Sign abstraction:

\[ \mathcal{O}(\mathbb{Z}) \xleftrightarrow{\gamma}_{\alpha} \{\bot, +, 0, -\top\} \]

where

\[ \alpha(Z) = \begin{cases} \bot & Z = \emptyset \\ + & \forall z \in Z. \ z > 0 \\ 0 & Z = \{0\} \\ - & \forall z \in Z. \ z < 0 \\ \top & \text{otherwise} \end{cases} \]

\[ \gamma(\bot) = \emptyset \]
\[ \gamma(\top) = \mathbb{Z} \]
\[ \gamma(+) = \{z \in \mathbb{Z} \mid z > 0\} \]
\[ \gamma(0) = \{0\} \]
\[ \gamma(-) = \{z \in \mathbb{Z} \mid z < 0\} \]
Example: Interval Abstraction

\[\varphi(\mathbb{Z}) \xleftarrow{\gamma} \{\bot\} \cup \{[a, b] \mid a \in \mathbb{Z} \cup \{-\infty\}, b \in \mathbb{Z} \cup \{+\infty\}\}\]

\[
\begin{align*}
\gamma(\bot) & = \emptyset \\
\gamma([a, b]) & = \{z \in \mathbb{Z} \mid a \leq z \leq b\} \\
\gamma([a, +\infty)) & = \{z \in \mathbb{Z} \mid z \geq a\} \\
\gamma([-\infty, b]) & = \{z \in \mathbb{Z} \mid z \leq b\} \\
\gamma([-\infty, +\infty]) & = \mathbb{Z}
\end{align*}
\]
Requirement 2: $\hat{F}$ and $F$

Plan: static analysis is computing an upper bound of $\bigcup_{i \in \mathbb{N}} \hat{F}^i(\bot)$

- For any $x \in D, \hat{x} \in \hat{D}$, $\hat{F}$ and $F$ must satisfy

$$\alpha(x) \sqsubseteq \hat{x} \implies \alpha(F(x)) \sqsubseteq \hat{F}(\hat{x})$$

- Intuition: the result of one-step abstract execution subsumes that of one-step real execution.

- or, alternatively,

$$\alpha \circ F \sqsubseteq \hat{F} \circ \alpha \quad (\text{i.e., } F \circ \gamma \sqsubseteq \gamma \circ \hat{F})$$
Then: a Correct Static Analysis

- static analysis = computing an upper bound of $\bigcup_{i \in \mathbb{N}} \hat{F}^i(\bot)$.

- Such an upper bound $\hat{A}$ is correct:
  
  \[
  \alpha(\text{fix } F) \subseteq \hat{A}, \quad \text{that is,} \\
  \text{fix } F \subseteq \gamma \hat{A}
  \]

- Theorem [fixpoint-transfer]

- Analysis result $\hat{A}$ subsumes the real execution $\text{fix } F$
Soundness Guarantee

Theorem (Fixpoint Transfer)

Let $D$ and $\hat{D}$ be related by Galois-connection $D \xleftrightarrow{\gamma} \hat{D}$. Let $F : D \rightarrow D$ be a continuous function and $\hat{F} : \hat{D} \rightarrow \hat{D}$ be a monotone function such that $\alpha \circ F \sqsubseteq \hat{F} \circ \alpha$. Then,

$$\alpha(\text{fix } F) \sqsubseteq \bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\bot).$$

Theorem (Fixpoint Transfer2)

Let $D$ and $\hat{D}$ be related by Galois-connection $D \xleftrightarrow{\gamma} \hat{D}$. Let $F : D \rightarrow D$ be a continuous function and $\hat{F} : \hat{D} \rightarrow \hat{D}$ be a monotone function such that $\alpha(x) \sqsubseteq \hat{x} \implies \alpha(F(x)) \sqsubseteq \hat{F}(\hat{x})$. Then,

$$\alpha(\text{fix } F) \sqsubseteq \bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\bot).$$
Example: Sign Analysis

- Plan

\[ \phi(\text{State}^+) \xleftarrow{\gamma_1} \mathcal{C} \xrightarrow{\gamma_2} \phi(\mathcal{M}) \xleftarrow{\alpha_2} \mathcal{C} \xrightarrow{\alpha_1} \hat{\mathcal{M}} \]

- \( \alpha_1 \) : partitioning abstraction

- \( \alpha_2 \) : memory state abstraction

Lemma

If \( D_1 \xleftarrow{\gamma_1} D_2 \) and \( D_2 \xleftarrow{\gamma_2} D_3 \), then

\[ D_1 \xleftarrow{\gamma_1 \circ \gamma_2} D_3. \]
Example: Sign Analysis
(Step 1: Partitioning Abstraction)

Galois-connection: \( \emptyset \left( \text{State}^+ \right) \xleftarrow{\gamma_1} \alpha_1 \xrightarrow{\alpha_1} C \rightarrow \emptyset (M) \)

\[ \alpha_1 (\Sigma) = \lambda c. \{ m \in M \mid \exists \sigma \in \Sigma \land \exists i. \sigma_i = (c, m) \} \]

Semantic function:

\[ \hat{F}_1 : (C \rightarrow \emptyset (M)) \rightarrow (C \rightarrow \emptyset (M)) \]

\[ \hat{F}_1 (X) = \alpha_1 (I) \sqcup \lambda c \in C. f_c ( \bigcup_{c' \rightarrow c} X (c')) \]

where \( f_c : \emptyset (M) \rightarrow \emptyset (M) \) is a transfer function at program point \( c \):

\[ f_c (M) = \{ m' \mid m \in M \land m' = \text{cmd}(c)(m) \} \]

Lemma (Soundness of Partitioning Abstraction)

\[ \alpha_1 (\text{fix } F') \subseteq \bigcup_{i \in \mathbb{N}} \hat{F}^i (\bot). \]
Example: Sign Analysis
(Step 2: Memory State Abstraction)

Galois-connection:

\[ C \rightarrow \phi(M) \xleftrightarrow{\gamma_2/\alpha_2} C \rightarrow \hat{M} \]

\[ \alpha_2(f) = \lambda c. \, \alpha_m(f(c)) \]

\[ \gamma_1(\hat{f}) = \lambda c. \, \gamma_m(\hat{f}(c)) \]

where we assume

\[ \phi(M) \xleftrightarrow{\gamma_m/\alpha_m} \hat{M} \]

Semantic function \( \hat{F} : (C \rightarrow \hat{M}) \rightarrow (C \rightarrow \hat{M}) \):

\[ \hat{F}(X) = (\alpha_2 \circ \alpha_1)(I) \cup \lambda c \in C. \, \hat{f}_c( \bigsqcup_{c' \rightarrow c} X(c')) \]

where abstract transfer function \( \hat{f}_c : \hat{M} \rightarrow \hat{M} \) is given such that

\[ \alpha_m \circ f_c \sqsubseteq \hat{f}_c \circ \alpha_m \]  \hspace{1cm} (1)

**Theorem (Soundness)**

\[ \alpha(\text{fix}F) \sqsubseteq \bigcup_{i \in \mathbb{N}} \hat{F}^i(\bot) \] where \( \alpha = \alpha_2 \circ \alpha_1 \).
Example: Sign Analysis
(Step 2: Memory State Abstraction)

Memory state abstraction:
\[ \varphi(M) \xleftrightarrow[\alpha_m]{\gamma_m} \hat{M} \]

\[ \alpha_m(M) = \lambda x \in \text{Var. } \alpha_s(\{m(x) \mid m \in M\}) \]
where \( \alpha_s \) is the sign abstraction:
\[ \varphi(Z) \xleftrightarrow[\alpha_s]{\gamma_s} \hat{Z} \]

The transfer function \( \hat{f}_c : \hat{M} \rightarrow \hat{M} \):
\[ \hat{f}_c(\hat{m}) = \hat{m} \quad c = \text{skip} \]
\[ \hat{f}_c(\hat{m}) = \hat{m}[x \mapsto \hat{V}(e)(\hat{m})] \quad c = x := e \]
\[ \hat{V}(n)(\hat{m}) = \alpha_s(\{n\}) \]
\[ \hat{V}(x)(\hat{m}) = \hat{m}(x) \]
\[ \hat{V}(e_1 + e_2) = \hat{V}(e_1)(\hat{m}) + \hat{V}(e_2)(\hat{m}) \]
\[ \hat{V}(e_1 - e_2) = \hat{V}(e_1)(\hat{m}) - \hat{V}(e_2)(\hat{m}) \]

Lemma
\[ \alpha_m \circ f_c \sqsubseteq f_c \circ \alpha_m \]
## Abstract Addition

<table>
<thead>
<tr>
<th></th>
<th>(\bot)</th>
<th>(+)</th>
<th>(0)</th>
<th>(-)</th>
<th>(\top)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
</tr>
<tr>
<td>(+)</td>
<td>(\bot)</td>
<td>(+)</td>
<td>(+)</td>
<td>(\top)</td>
<td>(\top)</td>
</tr>
<tr>
<td>(0)</td>
<td>(\bot)</td>
<td>(+)</td>
<td>(0)</td>
<td>(\bot)</td>
<td>(\top)</td>
</tr>
<tr>
<td>(-)</td>
<td>(\bot)</td>
<td>(\top)</td>
<td>(\top)</td>
<td>(\bot)</td>
<td>(\bot)</td>
</tr>
<tr>
<td>(\top)</td>
<td>(\top)</td>
<td>(\top)</td>
<td>(\top)</td>
<td>(\top)</td>
<td>(\top)</td>
</tr>
</tbody>
</table>
Abstract Subtraction

\[
\begin{array}{ccccccc}
\hat{\_} & \bot & + & 0 & - & \top \\
\bot & \bot & \bot & \bot & \bot & \bot & \bot \\
+ & \bot & \top & + & + & \top \\
0 & \bot & - & 0 & + & \top \\
- & \bot & - & - & - & \top & \top \\
\top & \top & \top & \top & \top & \top & \top \\
\end{array}
\]
Example: Abstract Semantics

0: ENTRY

1: x := 0

2: y := 0

3: skip

4: x := x + 1

5: y := y + 1

6: skip

Iter

<table>
<thead>
<tr>
<th>c1</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2</td>
</tr>
<tr>
<td>c3</td>
</tr>
<tr>
<td>c4</td>
</tr>
<tr>
<td>c5</td>
</tr>
<tr>
<td>c6</td>
</tr>
</tbody>
</table>

Fixpoint reached!
How to compute such an upper bound?

- If abstract domain \( \hat{D} \) is finite (i.e., all chains are finite), we can directly compute

\[
\bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\perp).
\]

The computation always terminate.

- Otherwise, we compute a finite chain \( \hat{X}_0 \subseteq \hat{X}_1 \subseteq \hat{X}_2 \subseteq \ldots \) such that

\[
\bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\perp) \subseteq \lim_{i \in \mathbb{N}} \hat{X}_i
\]
Abstract Domain

\[
\text{gfp } \hat{F} = \{ x \in \hat{D} | x \supseteq \hat{F}(x) \}
\]

\[
\text{fix}(\hat{F}) = \{ x \in \hat{D} | \hat{F}(x) = x \}
\]

\[
\text{postfp}(\hat{F}) = \{ x \in \hat{D} | x \supseteq \hat{F}(x) \}
\]

\[
\text{prefp}(\hat{F}) = \{ x \in \hat{D} | x \subseteq \hat{F}(x) \}
\]

The ideal result

\[
lfp \hat{F}
\]
Basic Upward/Downward Fixpoint Iteration

\[ \text{prefp}(\hat{F}) = \{ x \in \hat{D} \mid x \sqsubseteq \hat{F}(x) \} \]

\[ \text{fix}(\hat{F}) = \{ x \in \hat{D} \mid \hat{F}(x) = x \} \]

\[ \text{postfp}(\hat{F}) = \{ x \in \hat{D} \mid x \sqsupseteq \hat{F}(x) \} \]
Widening: Overshooting via Extrapolation

\[ \text{prefp}(\hat{F}) = \{ x \in \hat{D} \mid x \subseteq \hat{F}(x) \} \]

\[ \text{gfp} \ \hat{F} = \hat{F}^w(\perp) \]

\[ \hat{F}(\perp) \sqcup \hat{F}^2(\perp) \]

\[ \text{postfp}(\hat{F}) = \{ x \in \hat{D} \mid x \supseteq \hat{F}(x) \} \]
Refining the Widened Result

\[ \text{lfp } \hat{F} = \hat{F}^w(\bot) \]

\[ \text{gfp } \hat{F} = \hat{F}^w(\hat{x}) \]

\[ \text{prefp}(\hat{F}) = \{ x \in \hat{D} | x \sqsubseteq \hat{F}(x) \} \]

\[ \text{postfp}(\hat{F}) = \{ x \in \hat{D} | x \sqsupseteq \hat{F}(x) \} \]
Narrowing

\[ \text{prefp}(\hat{F}) = \{ x \in \hat{D} \mid x \sqsubseteq \hat{F}(x) \} \]

\[ \text{postfp}(\hat{F}) = \{ x \in \hat{D} \mid x \supseteq \hat{F}(x) \} \]
Widening

• We can define a finite chain with an widening operator $\nabla$

\[
\begin{align*}
\hat{X}_0 &= \bot \\
\hat{X}_{i+1} &= \begin{cases} 
\hat{X}_i & \text{if } \hat{F}(\hat{X}_i) \sqsubseteq \hat{X}_i \\
\hat{X}_i \nabla \hat{F}(\hat{X}_i) & \text{o.w.}
\end{cases}
\end{align*}
\]

• Conditions on $\nabla$:
  - $\forall a, b \in D. \ (a \sqsubseteq a \nabla b) \land (b \sqsubseteq a \nabla b)$
  - For all increasing chains $(x_i)_i$, the increasing chain $(y_i)_i$ defined as

\[
y_i = \begin{cases} 
x_0 & \text{if } i = 0 \\
y_{i-1} \nabla x_i & \text{if } i > 0
\end{cases}
\]

eventually stabilizes (i.e., the chain is finite).
Widening

- Then
  - \( \hat{X}_0 \subseteq \hat{X}_1 \subseteq \cdots \subseteq \hat{X}_n \) is a finite chain.
  - Its limit is correct:

**Theorem (Widening’s Safety)**

Let \( \hat{D} \) be a CPO, \( \hat{F} : \hat{D} \rightarrow \hat{D} \) a monotone function, \( \triangledown : \hat{D} \times \hat{D} \rightarrow \hat{D} \) a widening operator. Then, chain \( (\hat{X}_i)_i \) defined as (2) eventually stabilizes and

\[
\bigsqcup_{i \in \mathbb{N}} \hat{F}^i(\bot) \subseteq \lim_{i \in \mathbb{N}} \hat{X}_i.
\]
Widening for Interval

• A simple widening operator for the Interval domain:

\[
\begin{align*}
[a, b] \triangledown \bot &= [a, b] \\
\bot \triangledown [c, d] &= [c, d] \\
[a, b] \triangledown [c, d] &= [(c < a? - \infty : a), (b < d? + \infty : b)]
\end{align*}
\]

• Example: \([1, 2] \triangledown [1, 3] = [1, +\infty] \]
Narrowing

- We can refine the widened result \( \hat{A} \equiv \lim_{i \in \mathbb{N}} (\hat{X}_i) \) with a narrowing operator \( \triangle \).

- Compute chain \( \{\hat{Y}_i\}_i \)
  
  \[
  \hat{Y}_0 = \hat{A} \\
  \hat{Y}_{i+1} = \hat{Y}_i \triangle \hat{F}(\hat{Y}_i)
  \]

- Conditions:
  - \( \forall a, b \in \hat{D} : a \sqsupseteq b \Rightarrow a \sqsupseteq (a \triangle b) \sqsupseteq b \)
  - For all decreasing chain \( (x_i)_i \), the decreasing chain \( (y_i)_i \) defined as
    
    \[
    y_i = \begin{cases} 
      x_i & \text{if } i = 0 \\
      y_{i-1} \triangle x_i & \text{if } i > 0
    \end{cases}
    \]
    eventually stabilizes.
Narrowing

• Then

  • \( \{\hat{Y}_i\}_i \) is a finite chain.

  • Its limit is still correct:

\[
\lim_{i \to \infty} \hat{Y}_i
\]

Theorem (Narrowing’s Safety)

Let \( \hat{D} \) be a CPO, \( \hat{F} : \hat{D} \to \hat{D} \) a monotone function, \( \triangle : \hat{D} \times \hat{D} \to \hat{D} \) a narrowing operator. Then, chain \( (\hat{Y}_i)_i \) defined as (3) eventually stabilizes and

\[
\bigcup_{i \in \mathbb{N}} \hat{F}^i(\bot) \sqsubseteq \lim_{i \in \mathbb{N}} \hat{Y}_i.
\]
Narrowing for Interval

- A simple narrowing operator:

\[
\begin{align*}
[a, b] \triangle \bot &= \bot \\
\bot \triangle [c, d] &= \bot \\
[a, b] \triangle [c, d] &= [(a = -\infty?c : a), (b = +\infty?d : b)]
\end{align*}
\]

- Example: \([1, +\infty] \triangle [1, 3] = [1, 3]\)
Example

1: \( x := 0; \)
2: \( y := 0; \)
3: while \((x < 10)\) {
4: \( x := x + 1; \)
5: \( y := y + 1; \)
6: }
7: print(x)

Diagram:

0: ENTRY
1: \( x := 0 \)
2: \( y := 0 \)
3: \( x < 10 \)
4: \( x := x + 1 \)
5: \( y := y + 1 \)
6: skip

Decision point:

Y

N
Example

\[ \bigcup_{i \in \mathbb{N}} \hat{F}^i(\hat{\bot}): \]

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>...</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>(X_1)</td>
<td>(\bot)</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>...</td>
<td>[0,0]</td>
</tr>
<tr>
<td>(X_2)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>...</td>
<td>[0,0]</td>
</tr>
<tr>
<td>(X_3)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,1]</td>
<td>...</td>
<td>[0,9]</td>
</tr>
<tr>
<td>(X_4)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>...</td>
<td>[1,10]</td>
</tr>
<tr>
<td>(X_5)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>...</td>
<td>[1,10]</td>
</tr>
<tr>
<td>(X_6)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>(\bot)</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>...</td>
<td>[10,10]</td>
</tr>
</tbody>
</table>

0: ENTRY
1: \(x := 0\)
2: \(y := 0\)
3: \(x < 10\)
4: \(x := x + 1\)
5: \(y := y + 1\)
6: skip

\(Y\) \rightarrow N
### Widening Iterations

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>...</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$\perp$</td>
<td>[0,0]</td>
<td></td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td></td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td></td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[0,0]</td>
</tr>
<tr>
<td>$X_4$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td></td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>[1,0]</td>
<td>[1,0]</td>
</tr>
<tr>
<td>$X_5$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td></td>
<td>[1,1]</td>
<td>[1,1]</td>
<td>[1,0]</td>
<td>[1,0]</td>
</tr>
<tr>
<td>$X_6$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td></td>
<td>[0,0]</td>
<td>[0,0]</td>
<td>[10,0]</td>
<td>[10,0]</td>
</tr>
</tbody>
</table>

$X_3' = X_3 \lor ((X_2 \cup X_5) \cap [-\infty, 9])$

= $[0,0] \lor (([0,0] \cup [1,1]) \cap [-\infty, 9])$

= $[0,0] \lor [0,1] = [0, +\infty]$

```
1: x := 0
2: y := 0
3: x < 10
4: x := x + 1
5: y := y + 1
6: skip
```
# Narrowing Iterations

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>$[0,0]$</td>
<td>$[0,0]$</td>
<td>$[0,0]$</td>
<td>$[0,0]$</td>
</tr>
<tr>
<td>$X_2$</td>
<td>$[0,0]$</td>
<td>$[0,0]$</td>
<td>$[0,0]$</td>
<td>$[0,0]$</td>
</tr>
<tr>
<td>$X_3$</td>
<td>$[0,\infty]$</td>
<td>$[0,9]$</td>
<td>$[0,9]$</td>
<td>$[0,9]$</td>
</tr>
<tr>
<td>$X_4$</td>
<td>$[1,\infty]$</td>
<td>$[1,\infty]$</td>
<td>$[1,10]$</td>
<td>$[1,10]$</td>
</tr>
<tr>
<td>$X_5$</td>
<td>$[1,\infty]$</td>
<td>$[1,\infty]$</td>
<td>$[1,\infty]$</td>
<td>$[1,10]$</td>
</tr>
<tr>
<td>$X_6$</td>
<td>$[10,\infty]$</td>
<td>$[10,\infty]$</td>
<td>$[10,10]$</td>
<td>$[10,10]$</td>
</tr>
</tbody>
</table>

$X_3' = X_3 \triangle ((X_2 \cup X_5) \cap [-\infty, 9])$

$= [0,\infty] \triangle (([0,0] \cup [1, \infty]) \cap [-\infty, 9])$

$= [0,\infty] \triangle [0,9] = [0,9]$
Conclusion

• Abstract interpretation: a systematic method for sound approximations of concrete semantics

• Requirements: Galois connection / Soundness of abstract semantic function

• Termination: Widening / Narrowing