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 ≤ 2K?”, etc
Abstract Interpretation

- A powerful framework for designing correct static analysis
  - “framework”: correct static analysis comes out, reusable
  - “powerful”: all static analyses are understood in this framework
  - “simple”: prescription is simple
  - “eye-opening”: any static analysis is an abstract interpretation
Why Abstraction?

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

• Abstraction is not omission
  • reality: \{2, 4, 6, 8, \ldots \}
  • “even number” (abstraction) vs “multiple of 4” (omission)
Abstraction

- MAX++
- *a++ =
- *0
Abstraction

Abstract Interpretation

\[ \text{MAX}++ \]

\[ *\text{a}++ = \]

\[ *0 \]
Abstraction

Abstract Interpretation

The static analysis game

MAX++

*a++ =

*0
Concrete Interpretation  
(Standard Semantics)

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

Execution Trace:
(Program Points \( x \ (\text{Var} \to \mathbb{Z})^+ \))

(1, \( \{x \mapsto 0\} \))
(2, \( \{x \mapsto 0, y \mapsto 0\} \))
(3, \( \{x \mapsto 0, y \mapsto 0\} \))
(4, \( \{x \mapsto 1, y \mapsto 0\} \))
\ldots
(3, \( \{x \mapsto 10, y \mapsto 10\} \))
(7, \( \{x \mapsto 10, y \mapsto 10\} \))
Concrete Interpretation
(Collecting Semantics)

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 $x$ are
\{1, ..., 10\} after executing line 4.

(1, $\{x \mapsto \{0\}\}$)
(2, $\{x \mapsto \{0\}, y \mapsto \{0\}\}$)
(3, $\{x \mapsto \{0,1,\ldots,9\}$,
    $y \mapsto \{0,1,\ldots,9\}$)
(4, $\{x \mapsto \{1,2,\ldots,10\}$,
    $y \mapsto \{0,1,\ldots,9\}$)

...
Abstract Interpretation
(Abstract Semantics)

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

Abstract State:
: Program Points → (Var → Interval)
(1, {x ↦ [0, 0]})
(2, {x ↦ [0, 0], y ↦ [0, 0]})
(3, {x ↦ [0, 9], y ↦ [0, 9]})
(4, {x ↦ [1, 10], y ↦ [0, 9]})

The possible value of x ranges from 1 to 10 after executing line 4.

(7, {x ↦ [10, 10], y ↦ [10, 10]})
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)

Partitioned Execution Traces

(1, \{x \mapsto \{0\}\})
(2, \{x \mapsto \{0\}, y \mapsto \{0\}\})
(3, may be undecidable)
(4, may be undecidable)

... 
(7, may be undecidable)
Abstract Interpretation  
(Abstract Semantics)

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

**Abstract State**

(1, \{x \mapsto [0, 0]\})

(2, \{x \mapsto [0, 0], y \mapsto [0, 0]\})

(3, \{x \mapsto [0,\infty], y \mapsto [0,\infty]\})

(4, \{x \mapsto [1,\infty], y \mapsto [1,\infty]\})

(7, \{x \mapsto [0,\infty], y \mapsto [0,\infty]\})

The possible value of \( x \) is greater than or equal to 1 after executing line 4.
Need for Theory

• How to ensure that we soundly approximate real executions?

• How to ensure the termination of analysis?
Abstract Interpretation Framework

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

- The framework requires:
  - a relation between \( D \) and \( \hat{D} \)
  - 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.
Abstract Interpretation Framework

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

- 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.
Plan

• Step 1: Define concrete semantics

• Step 2: Define abstract semantics

• Step 3: Compute abstract semantics guaranteeing termination
  • with a finite abstract domain
  • with an 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.

- \( \{1,2\} \sqsubseteq \{1,2,3\} \)
- \( \emptyset = \perp \)
Least Upper Bound

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 \(\sqcup Y\).

Lemma

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

- \(\{1,2\} \sqcup \{3\} = \{1,2,3\}\)
Chain

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 $(\varnothing(\{a, b, c\}), \subseteq)$.

- $Y_1 = \{\emptyset, \{a\}, \{a, c\}\}$
- $Y_2 = \{\emptyset, \{a\}, \{c\}, \{a, c\}\}$
CPO

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

**Definition (Complete Lattice)**
A poset \((D, \sqsubseteq)\) is a complete lattice, if every subset \(Y \subseteq D\) has \(\bigsqcup Y \in D\).

**Lemma**
If \((D, \sqsubseteq)\) is a CPO, then it has a least element \(\bot\) given by \(\bot = \bigsqcup \emptyset\).
Definition (Continuous Functions)

A function \( f : D_1 \rightarrow D_2 \) defined on posets \((D_1, \sqsubseteq_1)\) and \((D_2, \sqsubseteq_2)\) is continuous if

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

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.
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

- \(f(\text{fix}(f)) = \text{fix}(f)\)
- \(\forall d \in D. \ f(d) = d \implies \text{fix}(f) \sqsubseteq d\)
Step 1: Define Concrete Semantics

- Define a semantic domain $D$, which is a CPO

- Any increasing chain $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ 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 $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$

  $$F(\bigsqcup_{n \geq 0} d_n) = \bigsqcup_{n \geq 0} F(d_n).$$
Step 1: Define Concrete Semantics

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

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

**Theorem (Kleene Fixed Point)**

Let $f : D \to D$ 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 $f$:

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

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

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

  (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,

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

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

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

Therefore, we conclude

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

• Program as a control flow graph (CFG)
  • \((C, \rightarrow, c_0)\)
  • Each node \(c \in C\) is with a command \(\text{cmd}(c)\)

\[
\begin{align*}
\text{cmd} & \rightarrow \text{skip} | x := e \\
& \rightarrow n | x | e + e | e - e
\end{align*}
\]
Example: Concrete (Collecting) Semantics

- Semantics of commands:

- Memory: \( M = \text{Var} \rightarrow \mathbb{Z} \)

- Semantics:

\[
\begin{align*}
[e] : M &\rightarrow M \\
[skip](m) &= m \\
[x := e](m) &= m[x \mapsto [e](s)]
\end{align*}
\]

\[
\begin{align*}
[e] : M &\rightarrow \mathbb{Z} \\
[n](m) &= n \\
[x](m) &= m(x) \\
[e_1 + e_2](m) &= [e_1](m) + [e_2](m) \\
[e_1 - e_2](m) &= [e_1](m) + [e_2](m)
\end{align*}
\]

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

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

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

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

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

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

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

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

$$\llbracket P \rrbracket = \{ \sigma \in \text{State}^+ | \sigma_0 \in I \land \forall k.\sigma_k \leadsto \sigma_{k+1} \}$$

The semantic domain:

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

The semantic function:

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

$$F(\Sigma) = I \cup \{ \sigma \cdot (c, m) | \sigma \in \Sigma \land \sigma \vdash \leadsto (c, m) \}$$

Lemma

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

\[
F(\emptyset) = I \cup \{\sigma \cdot (c, m) \mid \sigma \in \emptyset, \sigma \rightarrow (c, m)\} \\
= \{(c_0, m_0)\}
\]

\[
F^2(\emptyset) = I \cup \{\sigma \cdot (c, m) \mid \sigma \in I, \sigma \rightarrow (c, m)\} \\
= \{(c_0, m_0), (c_0, m_0) \cdot (c_1, m_0[x \mapsto 0])\}
\]

\[
F^3(\emptyset) = I \cup \{\sigma \cdot (c, m) \mid \sigma \in F^2(\emptyset), \sigma \rightarrow (c, m)\} \\
= \{(c_0, m_0), (c_0, m_0) \cdot (c_1, m_0[x \mapsto 0]), (c_0, m_0) \cdot (c_1, m_0[x \mapsto 0]) \cdot (c_2, m_0[x \mapsto 0][y \mapsto 0])\}
\]

\ldots
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:

\[
D \xleftarrow{\alpha} \xrightarrow{\gamma} \hat{D}
\]

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 \( D \)
- \( \forall x \in D, \hat{x} \in \hat{D}. \alpha(x) \subseteq \hat{x} \iff x \subseteq \gamma(\hat{x}) \)
  - \( \alpha \) and \( \gamma \) respect the orderings of \( D \) and \( \hat{D} \)
Requirement 1: Galois Connection
Example: Sign Abstraction

$D = 2^\mathbb{Z}$

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

$\emptyset$

$\gamma$

$\alpha$

$\hat{D}$
Example: Sign Abstraction

Sign abstraction:

$$\varphi(\mathbb{Z}) \xleftrightarrow[\gamma]{\alpha} \{\perp, +, 0, -\top\}$$

where

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

$$\gamma(\perp) = \emptyset$$

$$\gamma(\top) = \mathbb{Z}$$

$$\gamma(+) = \{z \in \mathbb{Z} \mid z > 0\}$$

$$\gamma(0) = \{0\}$$

$$\gamma(-) = \{z \in \mathbb{Z} \mid z < 0\}$$
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 $\bigcup_{i \in \mathbb{N}} \hat{F}^i(\perp)$

- 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 \( \bigcup_{i \in \mathbb{N}} \hat{F}^i(\perp) \).

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

- 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 $D \xleftarrow{\gamma} \alpha \xrightarrow{\alpha} \hat{D}$. Let $F : D \to D$ be a continuous function and $\hat{F} : \hat{D} \to \hat{D}$ be a monotone function such that $\alpha \circ F \subseteq \hat{F} \circ \alpha$. Then,

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

**Theorem (Fixpoint Transfer2)**

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

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

• Plan

\[ \varphi(\text{State}^+) \xrightarrow{\gamma_1} C \rightarrow \varphi(\hat{M}) \xrightarrow{\gamma_2} C \rightarrow \hat{M} \]

\[ \hat{M} = \text{Var} \rightarrow \hat{D} \]

• \( \alpha_1 \) : partitioning abstraction

• \( \alpha_2 \) : memory state abstraction

Lemma

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

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

Galois-connection: \( \varphi(\text{State}^+) \xleftrightarrow[\gamma_1]{\alpha_1} \mathbb{C} \rightarrow \varphi(\mathbb{M}) \)

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

Semantic function:

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

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

where \( f_c : \varphi(\mathbb{M}) \rightarrow \varphi(\mathbb{M}) \) is a transfer function at program point \( c \):

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

Lemma (Soundness of Partitioning Abstraction)

\[
\alpha_1(\text{fix}F') \subseteq \bigcup_{i \in \mathbb{N}} \hat{F}_1^i(\bot).
\]
Example: Sign Analysis
(Step 2: Memory State Abstraction)

Galois-connection:
\[
\begin{align*}
& \mathbb{C} \rightarrow \mathfrak{F}(\hat{\mathbb{M}}) \overset{\gamma_2}{\underset{\alpha_2}{\leftrightarrow}} \mathbb{C} \rightarrow \hat{\mathbb{M}} \\
& \alpha_2(f) = \lambda c. \alpha_m(f(c)) \\
& \gamma_1(\hat{f}) = \lambda c. \gamma_m(\hat{f}(c))
\end{align*}
\]

where we assume
\[
\mathfrak{F}(\hat{\mathbb{M}}) \overset{\gamma_m}{\underset{\alpha_m}{\leftrightarrow}} \hat{\mathbb{M}}
\]

Semantic function \( \hat{\mathcal{F}} : (\mathbb{C} \rightarrow \hat{\mathbb{M}}) \rightarrow (\mathbb{C} \rightarrow \hat{\mathbb{M}}) : \)
\[
\hat{\mathcal{F}}(X) = (\alpha_2 \circ \alpha_1)(I) \sqcup \lambda c \in \mathbb{C}. \hat{f}_c(\bigsqcup_{c' \rightarrow c} X(c'))
\]

where abstract transfer function \( \hat{f}_c : \hat{\mathbb{M}} \rightarrow \hat{\mathbb{M}} \) is given such that
\[
\alpha_m \circ f_c \sqsubseteq \hat{f}_c \circ \alpha_m \tag{1}
\]

**Theorem (Soundness)**
\[
\alpha(\text{fix}\mathcal{F}) \sqsubseteq \bigcup_{i \in \mathbb{N}} \hat{\mathcal{F}}^i(\perp) \text{ where } \alpha = \alpha_2 \circ \alpha_1.
\]
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:

\[ \varphi(\mathbb{Z}) \xleftrightarrow[\alpha_s]{\gamma_s} \hat{\mathbb{Z}} \]

The transfer function \( \hat{f}_c : \hat{M} \rightarrow \hat{M} \):

\[
\begin{align*}
\hat{f}_c(\hat{m}) &= \hat{m} & c &= \text{skip} \\
\hat{f}_c(\hat{m}) &= \hat{m}[x \mapsto \hat{V}(e)(\hat{m})] & c &= x := e \\
\hat{V}(n)(\hat{m}) &= \alpha_s(\{n\}) \\
\hat{V}(x)(\hat{m}) &= \hat{m}(x) \\
\hat{V}(e_1 + e_2) &= \hat{V}(e_1)(\hat{m}) + \hat{V}(e_2)(\hat{m}) \\
\hat{V}(e_1 - e_2) &= \hat{V}(e_1)(\hat{m}) - \hat{V}(e_2)(\hat{m})
\end{align*}
\]

Lemma

\[ \alpha_m \circ f_c \subseteq \hat{f}_c \circ \alpha_m \]
Abstract Addition

<table>
<thead>
<tr>
<th>(\hat{+})</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>(\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>(\bot)</td>
</tr>
<tr>
<td>(-)</td>
<td>(\bot)</td>
<td>(\top)</td>
<td>(-)</td>
<td>(-)</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

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

<table>
<thead>
<tr>
<th>Iter</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
<td>{x \mapsto 0}</td>
</tr>
<tr>
<td>c2</td>
<td>{y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
</tr>
<tr>
<td>c3</td>
<td>{}</td>
<td>{y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto +, y \mapsto +}</td>
<td>{x \mapsto +, y \mapsto +}</td>
<td>{x \mapsto +, y \mapsto +}</td>
<td>{x \mapsto +, y \mapsto +}</td>
</tr>
<tr>
<td>c4</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{x \mapsto +, y \mapsto 0}</td>
<td>{x \mapsto +, y \mapsto 0}</td>
<td>{x \mapsto +, y \mapsto 0}</td>
<td>{x \mapsto +, y \mapsto 0}</td>
<td>{x \mapsto +, y \mapsto 0}</td>
</tr>
<tr>
<td>c5</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{}</td>
<td>{x \mapsto +, y \mapsto +}</td>
<td>{x \mapsto +, y \mapsto +}</td>
<td>{x \mapsto +, y \mapsto +}</td>
<td>{x \mapsto +, y \mapsto +}</td>
</tr>
<tr>
<td>c6</td>
<td>{}</td>
<td>{}</td>
<td>{y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
<td>{x \mapsto 0, y \mapsto 0}</td>
</tr>
</tbody>
</table>

Fixpoint reached!
Next

- What if the abstract domain is infinite?
- Widening / Narrowing
- Interval analysis
- Worklist algorithm for more efficient fixpoint computation