CIS 670: Program Analysis

Title: Abstract Interpretation.

Guest Lecturer: Sriram Sankaranarayanan.
NEC Labs America,
Princeton, NJ.
srirams@nec-labs.com

The story so far...

- Signs and Interval analyses: Lattice Inequalities.
- Iteration strategy for solving lattice inequalities.

\[ x_0 = f(\bot), \quad x_1 = f(x_0), \ldots \]

- The iteration converges if the lattice is finite.
- If the lattice is not finite, then iteration may diverge.
- We used widening to force convergence.
- Widening reaches a postfixed point
Ascending/Descending Iterations

\[ f(x_N) \subseteq x_N \quad -\quad \text{Widening post-fixed point} \]

\[ x_{i+1} = x_i \nabla f(x_i) \quad \ldots \quad f^i(x_N) = f^{i+1}(x_N) \]

\[ x_2 = x_1 \nabla f(x_1) \]

\[ x_1 = f(\bot) \nabla f^2(\bot) \quad \ldots \quad f^3(\bot) \]

\[ f^2(\bot) \quad \ldots \quad f(\bot) \quad \bot \]
**Descending Iteration: Convergence**

**Descending Chain Condition:** Dual to Ascending Chain condition.

Descending iteration need not necessarily converge in finitely many steps.

1. Stop the iteration after some fixed number of steps. This is not a good idea (for large programs).

2. Use a “narrowing” operator to force convergence.
Let $b \sqsubseteq a$, then $a \triangle b$ is intermediate to $a$, $b$.

$$b \sqsubseteq a \triangle b \sqsubseteq a.$$ 

Let $a_1 \sqsupseteq a_2 \sqsupseteq a_3 \sqsupseteq \ldots$ be an infinite decreasing iteration. **Narrowed iteration:** Define sequence $b_1, b_2, \ldots$,:

$$b_1 = a_1, \quad b_{i+1} = b_i \triangle (a_{i+1}).$$

(1) $b_1 \sqsupseteq b_2 \sqsupseteq \cdots \sqsupseteq b_N = b_{N+1}$ for $N > 0$.

(2) $\min_{\sqsubseteq} \{a_1, a_2, \ldots, \} \sqsubseteq b_N$. 

\begin{center}
Narrowing
\end{center}
Illustration:

\[ \begin{array}{c}
\hspace{1cm} x \\
\hspace{1cm} \triangle f(x) \\
\hspace{1cm} f(x)
\end{array} \]

Property:

If \( f(x) \sqsubseteq x \) then \( f(x \triangle f(x)) \sqsubseteq x \triangle f(x). \)

Therefore, result of narrowing is still part of the decreasing iteration.
Interval Narrowing

Let \([c, d] \subseteq [a, b]\). Then \([a, b] \bigtriangleup [c, d] = [\ell, u]\).

\[
\ell = \begin{cases} 
  c & a = -\infty \\
  a & \text{otherwise}
\end{cases}
\]

\[
u = \begin{cases} 
  d & b = \infty \\
  b & \text{otherwise}
\end{cases}
\]

Special case: \(x \bigtriangleup \bot = \bot\).
Interval Narrowing: Examples

\[ [1, 1] \triangle \perp = \perp \]
\[ [-1, \infty) \triangle [1, 10] = [-1, 10] \]
\[ [-1, \infty) \triangle [5, \infty) = [-1, \infty) \]
\[ [-\infty, \infty] \triangle [0, 10] = [0, 10] \]
Updated picture with Widening/Narrowing sequence
In order to improve precision:

- First apply $k > 0$ regular iterations,

$$x^0 = \bot, \; x^{i+1} = f(x^i), \text{ if } i < k.$$ 

- Then apply widening iteration until post fixed point.

$$x^{i+1} = x^i \nabla f(x^i).$$

- Similarly narrowing iteration can be delayed.
Example: Delayed Widening

\[ n_0 : (i, j) := (0, 0) \]

\[ n_1 \]

\[ n_2 : i < 100 \]

\[ n_3 : i == 0 \]

\[ n_4 : i++, j++ \]

\[ n_5 : i++ \]

\[ n_6 \]
With no delay in widening, we compute the fixed point at \( n_1 \):

\[
i \in [0, 100] \text{ and } j \in [0, \infty).
\]

With delay in widening (\(~5\) step delay), we can compute:

\[
i \in [0, 100] \text{ and } j \in [0, 1].
\]
Where to widen?

Our current approach says widen everywhere.

$n_0: x < 100$

$n_1: x := 0$

$n_2: x := 1$

$n_3: x := x + 1$

$n_4: x := x + 1$

$n_5$

**Question:** With delayless widening, what is the solution computed at $n_5$?
Widening needs to be applied when there are loops in the code.

Widening needs to be applied only at the loop heads:

\[ \chi_{j}^{i+1} = \begin{cases} 
  f(x_{j}^{i}) & \text{if } n_{j} \text{ not a loop head} \\
  \chi_{j}^{i} \nabla f(x_{j}^{i}) & \text{if } n_{j} \text{ is the head of a loop}
\end{cases} \]

Similarly, we need to narrow only at the heads of loops.
Widening Upto Operator

- Current widening goes from finite to infinity in one step:
  \[ [0, 0] \nabla [0, 1] = [0, \infty), \quad [0, 1] \nabla [-1, 1] = (-\infty, 1]. \]

- Upto set: A set of integer points. Eg.,
  \[ U = \{-1, 0, 1, 100, 200, 1000}\].

- Widening upto operator \( \nabla_U \): choose the smallest bound from the upto set to replace (if no bound exists, use \( \pm \infty \)).

- Eg., \( [-1, 5] \nabla_U [-1, 6] = [-1, 100], \quad [1, 10] \nabla_U [0, 10] = [-1, 10], \quad \ldots \).
The Big Picture

- Signs Analysis: Compute a sign for every variable.
- Interval Analysis: Compute an interval for every variable.
- Are these analyses [sound]? What does [soundness] mean?
Collecting Semantics

a.k.a “Concrete Interpretation”.

State: A program state is an assignment of integer values to variables.

\[ s : \langle x_1 : v_1, x_2 : v_2, \ldots, x_n : v_n \rangle. \]

Let \( \Sigma : \mathbb{Z} \times \mathbb{Z} \times \cdots \times \mathbb{Z} \) be the set of all program states.

Reachable states: Let \( \text{Reach}(n) \subseteq \Sigma \) be the set of all states reaching a location \( n \).
$n_0 : y := \cdots$

\[
\begin{align*}
\text{post}(n_0, \text{Reach}(n_0)) & \subseteq \text{Reach}(n_1) \\
\text{Reach}(n_5) & \subseteq \text{Reach}(n_1) \\
\text{Reach}(n_1) & \subseteq \text{Reach}(n_2) \\
\text{Reach}(n_2) \cap [y > 1] & \subseteq \text{Reach}(n_3) \\
\text{Reach}(n_3) \cap [\varphi_1] & \subseteq \text{Reach}(n_4) \\
\text{post}(n_4, \text{Reach}(n_4)) & \subseteq \text{Reach}(n_5) \\
\text{post}(n_6, \text{Reach}(n_6)) & \subseteq \text{Reach}(n_5)
\end{align*}
\]
• The concrete lattice is $\mathcal{C} : 2^\Sigma$ ordered by $\subseteq$.

• Reachable states can be expressed as a fix point of a monotonic function over sets of states.

\[
\text{Reach}(\cdot) : \{F(\emptyset) \cup F^2(\emptyset) \cup \cdots \cup F^n(\emptyset)\}.
\]

• This is however, a purely theoretical exercise.
  
  – The lattice of state sets $2^\Sigma$ has infinite height.
  
  – Arbitrary infinite sets cannot be represented inside a computer.
Galois Connections
Consider two lattices $\langle C, \subseteq \rangle$ and $\langle A, \subseteq \rangle$.

A **Galois Connection** between $C$ and $A$ is a pair of functions $\alpha : C \mapsto A$ and $\gamma : A \mapsto C$, such that

$$\text{for all } S \in C \text{ and } a \in A, \alpha(S) \subseteq a \text{ iff } S \subseteq \gamma(a).$$

$\alpha$ is called the “Abstraction Map” and $\gamma$ is called the “Concretization Map”.
Example #1: Signs Lattice

\[ \subseteq \quad \mathbb{Z} \]
\[ \{ x | x > 0 \} \quad \{ 0 \} \quad \{ x | x < 0 \} \]

\[ (I) = \begin{cases} 
\text{"⊥", if } I = \emptyset \\
\text{"+", if } I \subseteq \text{Pos} \\
\text{"-", if } I \subseteq \text{Neg} \\
\text{"0", if } I \equiv \{ 0 \} \\
\text{"⊤", o.w.} 
\end{cases} \]

\[ \gamma(c) = [c]. \]
Example #2: Interval Lattice

Let $C : 2^Z$ and $A : \text{Intervals}$.

$$\alpha(X) = [\min(X), \max(X)]$$

$$\gamma([\ell, u]) = [\ell, u] = \{z \mid \ell \leq z \leq u\}.$$ 

Verify the Galois connection.

$$(\forall I \subseteq Z, [\ell, u] \in \text{Intervals}) \ \alpha(I) \subseteq [\ell, u] \iff I \subseteq \gamma([\ell, u]).$$
Galois Connection: Intuition

\[ \alpha : \text{Sets of States} \rightarrow \text{Abstraction (signs/intervals/\ldots)} . \]

and

\[ \gamma : \text{Abstraction} \rightarrow \text{Sets of states it represents} . \]

**Question:** What does a Galois connection mean?

If \( \alpha \) abstracts a set \( S \) iff the concretization of \( \alpha \) overapproximates \( S \).
Galois Connection: "Best" abstraction & concretization

Property # 0: Derive $\alpha$ given $\gamma$ (and vice versa).

Idea:
"Best" abstraction of $S$ should be the smallest abstract element that contains $S$.

$$\alpha_b(S) = \min\{a \mid S \subseteq \gamma(a)\}.$$

Similarly, "best" concretization given $\alpha$ is

$$\gamma_b(a) = \max\{S \mid \alpha(S) \subseteq a\}.$$

Let us try to apply this to the two domains we have seen.
Galois Connection: Closure

**Property # 1:** \((\forall S \in C) \ S \subseteq \gamma(\alpha(S))\)

**Proof:**

\[
\alpha(S) \subseteq \alpha(S). \text{Therefore, } S \subseteq \gamma(\alpha(S)).
\]

**Property # 2:** \((\forall a \in A) \ \alpha(\gamma(a)) \subseteq a\)

**Proof:**

\[
\gamma(a) \subseteq \gamma(a). \text{Therefore, } \alpha(\gamma(a)) \subseteq a.
\]
Galois Connection: Monotonicity

Property # 3: $\alpha$ and $\gamma$ are monotonic. I.e.,

If $S_1 \subseteq S_2$ then $\alpha(S_1) \subseteq \alpha(S_2)$.

Similarly,

If $a_1 \subseteq a_2$ then $\gamma(a_1) \subseteq \gamma(a_2)$.

Proof: Let $S_1 \subseteq S_2$. We know from Property #1 that $S_2 \subseteq \gamma(\alpha(S_2))$. Therefore, $S_1 \subseteq \gamma(\alpha(S_2))$. Applying Galois connection definition, $\alpha(S_1) \subseteq \alpha(S_2)$.

Similarly, we can prove the other part too.
Property # 4: For all $S_1, S_2 \in C$,

$$\alpha(S_1 \cup S_2) = \alpha(S_1) \sqcup \alpha(S_2).$$

Proof: We rely on a sub-fact about lattices.

Fact: If for $a, b \in L$, for all $c \in L$,

$a \sqsubseteq c \iff b \sqsubseteq c$ then $a = b$. 

\[ \alpha(S_1 \cup S_2) \subseteq c \quad \text{iff} \quad S_1 \cup S_2 \subseteq \gamma(c) \]
\[ \text{iff} \quad S_1 \subseteq \gamma(c), \ S_2 \subseteq \gamma(c) \]
\[ \text{iff} \quad \alpha(S_1) \subseteq c, \ \alpha(S_2) \subseteq c \]
\[ \text{iff} \quad \alpha(S_1) \sqcup \alpha(S_2) \subseteq c \]

Now applying fact, we get

\[ \alpha(S_1 \cup S_2) = \alpha(S_1) \sqcup \alpha(S_2). \]
Meet Preservation

For all $S_1, S_2 \in C$,

$$\alpha(S_1 \cap S_2) = \alpha(S_1) \cap \alpha(S_2).$$

**Proof:** Use dual fact.
Monotone Function Theorem

Let $f : C \mapsto C$ and $g : A \mapsto A$ be monotone functions on $C, A$ respectively.

g is a sound abstraction of $f$ iff

$$\forall \, S \in C, \, \alpha(f(S)) \sqsubseteq g(\alpha(S)).$$

Claim: $\alpha(\text{LFP}_C(f)) \sqsubseteq \text{LFP}_A(g)$.

1. $\alpha(\emptyset) = \bot$

2. $\forall \, n \geq 0, \, \forall S \in C, \, \alpha(f^n(S)) \sqsubseteq g^n(\alpha(S))$

3. $\alpha(\text{LFP}(f)) \sqsubseteq \text{LFP}(g)$. 
Proving Soundness of Abstract Interpretation
We have a “concrete domain” $C : 2^\Sigma$ and abstract domain $\langle L, \sqsubseteq \rangle$.

Fixed point inside lattice $C : \text{Reach}(n)$.

Dataflow analysis inside lattice $L : \text{fp}(n)$ (eg., $	ext{sign}(n, x), \text{Rng}(n, y)$).

**Goal:** Relate concrete fixed point $\text{Reach}(n)$ with abstract fixed point $\text{fp}_L(n)$.

Let $\langle \alpha, \gamma \rangle$ be a galois connection between $C$ and $L$. 
We will establish $\alpha \circ f \subseteq g \circ \alpha$.

- For any sets $S_1, S_2$,
  \[
  \alpha(S_1 \cup S_2) \subseteq \alpha(S_1) \cup \alpha(S_2).
  \]
  This is the join preservation result.

- For sets $S_1, S_2$,
  \[
  \alpha(S_1 \cap S_2) \subseteq \alpha(S_1) \cap \alpha(S_2).
  \]
  The meet preservation result.
• For any set $S_1$,

$$\alpha(\text{post}_C(n, S)) \subseteq \text{post}_L(n, \alpha(S)).$$

This is a requirement.

• We can now lift the result to dataflow inequalities.
For a given program $P$,
Let $F(X) \subseteq X$ be the flow inequalities in the **concrete** domain.
Let $g(x) \subseteq x$ be the flow inequalities in the **abstract** domain.

**Obs. 1:** $F$ and $g$ are structurally identical.
For example,

$$F : \text{post}(n_0, X_0) \cup (X_1 \cap [I]) \cup \text{post}(n_1, X_1) \cup X_2.$$

and

$$g : \text{post}_L(n_0, x_0) \cup (x_1 \cap \alpha(I)) \cup \text{post}(n_1, x_1) \cup x_2.$$
**Reason:** The generation of dataflow inequalities is “syntax-directed”.

**Obs. 2:** $\alpha(F(X)) \subseteq g(\alpha(x))$.

**Proof:** Build this up from proof for basic operations.
Soundness

Let \( L \) be a dataflow lattice such that

1. There exists a **Galois connection** between \( L \) and concrete domain \( C \).

2. Post condition on \( L \) is sound, w.r.t post condition on \( C \),

\[
\alpha(\text{post}(n, S)) \subseteq \text{post}_L(n, \alpha(S)).
\]

given program we get the dataflow inequalities:
\( F(X) \subseteq X \) on \( C \) and \( g(x) \subseteq x \) on \( L \),
then, the least fixed point of \( g \) on \( L \) abstracts the LFP of \( F \) on \( C \).

\[
\text{LFP}_C(F) \subseteq \text{LFP}_L(g).
\]