Lecture 6. Abstract Interpretation

Wei Le

2014.10
Outline

- Motivation
- History
- What it is: an intuitive understanding
- An example
- Steps of abstract interpretation
- Galois connection
- Narrowing and Widening
- Fixed point
Motivation

- Static program analysis: automatically discovering properties of a program that hold for all possible execution paths of the program
- Discovering a sufficient set of properties for checking every operation of a program is an undecidable problem
- False positives
- Specialization:
  - Tailoring the program analyzer algorithms for a specific class of programs
  - Precision and scalability is guaranteed for this class of programs only
  - Requires a lot of try-and-test to fine-tune the algorithms
History: Patrick Cousot, Radhia Cousot 1977

- Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints, 1977
- Methods and Logics for Proving Programs, 1990
- Completeness in Abstract Interpretation, 1995
- Directions for Research in Approximate System Analysis, 1999
- Probabilistic Abstract Interpretation, 2012
- An abstract interpretation framework for termination, 2012
- Abstract interpretation: past, present and future, 2014
What it is [1]

- A theoretical framework to formalize *approximation*
- An applications of abstraction to the semantics of programming languages as well as to static program analysis
- Provides approximate methods for computing abstract semantics
An Example

See Prof. Alex Aiken’s slide
Steps of Abstract Interpretation

1. Concrete Semantics
2. Collecting Semantics
3. Partitioning
4. Abstract Semantics
5. Tuners
6. Iterative Resolution Algorithms

Abstract Domain
Abstract Domain
Concrete Semantics

Small-step operational semantics: \((\Sigma, \rightarrow)\)

\[
s = \langle \text{program point}, \text{env} \rangle \quad s \rightarrow s'
\]

Example:

1: \(n = 0;\)
2: \(\text{while } n < 1000 \text{ do}\)
3: \(n = n + 1;\)
4: \(\text{end}\)
5: \(\text{exit}\)

\[
\langle 1, n \Rightarrow \Omega \rangle \rightarrow \langle 2, n \Rightarrow 0 \rangle \rightarrow \langle 3, n \Rightarrow 0 \rangle \rightarrow \langle 4, n \Rightarrow 1 \rangle \\
\rightarrow \langle 2, n \Rightarrow 1 \rangle \rightarrow \cdots \rightarrow \langle 5, n \Rightarrow 1000 \rangle
\]

Undefined value
Collecting Semantics

- (A set of) Partial program states
- Precise, strongest static properties
- Examples: computation traces, forward/backward reachable states, predicate transformers
We formulate collecting semantics in terms of sets because they describe properties, e.g.,

- the set 1, 3, 5, ... describes the property odd
- the set 2, 4, 6, ... describes the property even
- the singleton set 42 describes a constant property
- the set 4, 5, 6, 7, 8, 9, 10 describes an interval property [4; 10]
Collecting Semantics: Examples

- The set of all descendants of the initial state
- The set of all descendants of the initial state that can reach a final state
- The set of all finite traces from the initial state
- The set of all finite and infinite traces from the initial state

- Buffer overrun, division by zero, arithmetic overflows: state properties
- Deadlocks, un-initialized variables: finite trace properties
- Loop termination: finite and infinite trace properties
Collecting Semantics: Example

*Trace (or path) semantics* model program computations by a set of finite or infinite sequences of states.
Collecting Semantics: Example

*Relational* and *Natural* semantics
Collecting Semantics: Example

Transition semantics
Collecting Semantics: Example

Reachable states
Partitioning and Abstract Domain

Partition: abstract sets of environments
Abstract domain:

Environment: \( x \Rightarrow v, \ y \Rightarrow w, \ldots \)

Various kinds of approximations:

- Intervals (nonrelational):
  \( x \Rightarrow [a, b], \ y \Rightarrow [a', b'], \ldots \)

- Polyhedra (relational):
  \( x + y - 2z \leq 10, \ldots \)

- Difference-bound matrices (weakly relational):
  \( y - x \leq 5, \ z - y \leq 10, \ldots \)
Partitioning and Abstract Domain

(a) [In]finite Set of Points

(b) Sign Abstraction

(c) Interval Abstraction

(d) Simple Congruence Abstraction

(b) Polyhedral Abstraction
Abstract Domain: Interval

1: \ n = 0;
2: \ \textbf{while} \ n < 1000 \ \textbf{do}
3: \ \ n = n + 1;
4: \ \textbf{end}
5: \ \textbf{exit}

- Iteration 1: \ E_2^* = [0, 0]
- Iteration 2: \ E_2^* = [0, 1]
- Iteration 3: \ E_2^* = [0, 2]
- Iteration 4: \ E_2^* = [0, 3]
- \ldots
Abstract Semantics: the Example

- Abstract values: +, -, 0
- Abstract operations:

<table>
<thead>
<tr>
<th>×</th>
<th>+</th>
<th>0</th>
<th>−</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>0</td>
<td>−</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>−</td>
<td>−</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

- Sound: the signedness properties preserve in the abstract domain

See the extension of the Prof. Alex Aiken’s example
Problem: Compute a sound approximation $S^\#$ of $S$

Solution: Galois connections
Abstract Interpretation

- An abstract interpretation consists of:
  - An abstract domain $A$ (+,-,0) and concrete domain $D$ (Int)
  - Concretization $\gamma$ and abstraction functions $\sigma$, forming a *Galois insertion*
  - A (sound) abstract semantic function
- Finite domains (lattice) + monotonic functions
  - Large domains = slow analysis
  - In practice, domains are forced to be small
  - Chain height is the critical measure
  - The focus in abstract interpretation is on correctness; not much insight into efficient algorithms
Galois Connection [2]
Concrete and Abstract Lattice

Concrete lattice

\((L, \sqsubseteq)\)

Abstract lattice

\((\overline{L}, \sqsubseteq)\)

Example (Sets of Values)

For a variable ranging over a domain \(\mathbb{D}\):

\((\mathcal{P}(\mathbb{D}), \subseteq)\)

Example (Sign Lattice)

\[
\begin{array}{c}
\top \\
\downarrow \\
0 \\
\downarrow \\
- \\
\downarrow \\
-0 \\
\downarrow \\
-0+ \\
\downarrow \\
0+ \\
\downarrow \\
+ \\
\downarrow \\
\bot
\end{array}
\]
Galois Connection: Definition

**Definition**

A Galois connection between a lattice \((L, \sqsubseteq)\) and a lattice \((\overline{L}, \sqsubseteq)\) is a pair of functions \((\alpha, \gamma)\), with \(\alpha : L \rightarrow \overline{L}\) and \(\gamma : \overline{L} \rightarrow L\), satisfying:

\[
\alpha(x) \sqsubseteq \overline{y} \quad \text{iff} \quad x \sqsubseteq \gamma(\overline{y}) \quad \text{(for all} \ x \in L, \overline{y} \in \overline{L})
\]

Notation for Galois connections: \((L, \sqsubseteq) \leftrightarrow \overline{L}, \sqsubseteq \)
Galois Connection: Intuition

The "order" (property) is preserved: Soundness

\[ (L, \sqsubseteq) \xrightarrow{\gamma} (\overline{L}, \sqsubseteq) \]

Concretization
\( \gamma \) is the concretization function.
\( \gamma(\overline{y}) \) is the concrete value in \( L \) that is represented by \( \overline{y} \).

Abstraction
\( \alpha \) is the abstraction function.
\( \alpha(x) \) is the most precise abstract value in \( \overline{L} \) whose concretization approximates \( x \).
Galois Connection: Characterization

Consider two lattices \((L, \sqsubseteq)\) and \((\bar{L}, \sqsubseteq)\).

For any two functions \(\alpha : L \rightarrow \bar{L}\) et \(\gamma : \bar{L} \rightarrow L\), we have

\[
(L, \sqsubseteq) \overset{\gamma}{\iff} \overset{\alpha}{\sqsubseteq} (\bar{L}, \sqsubseteq)
\]

iff

\[
\begin{cases}
  x \sqsubseteq \gamma \circ \alpha(x) & \text{(for all } x \in L) \\
  \alpha \circ \gamma(\bar{y}) \sqsubseteq \bar{y} & \text{(for all } \bar{y} \in \bar{L})
\end{cases}
\]

\(\alpha\) is monotonic
\(\gamma\) is monotonic
Galois Connection: Characterization

\[ (L, \sqsubseteq) \leftrightarrow \gamma \rightarrow (\overline{L}, \sqsupseteq) \]

\[ x \sqsubseteq \gamma \circ \alpha(x) \quad (\gamma \circ \alpha \text{ extensive}) \]
Galois Connection: Characterization

\[(L, \sqsubseteq) \leftrightarrow (\bar{L}, \sqsupseteq)\]

\[\alpha \circ \gamma(\bar{y}) \sqsubseteq \bar{y}\]

\[(\alpha \circ \gamma \text{ reductive})\]
Galois Connection: Characterization

$$\gamma$$

$$\alpha$$

$$\alpha$$ is monotonic
Galois Connection: Characterization

\((L, \sqsubseteq) \leftrightarrow (\overline{L}, \sqsupseteq)\)

\(\gamma\) is monotonic
Iterative Resolution Algorithm

- Widening: pair-widening, set-widening
- Narrowing: pair-narrowing, set-narrowing
Widening

Lattice \((L, \leq)\): \(\nabla : L \times L \rightarrow L\)

- **Abstract union operator:**
  \[\forall x \forall y : x \leq x \nabla y \ \& \ y \leq x \nabla y\]
- **Enforces convergence:** \((x_n)_{n \geq 0}\)

\[
\begin{cases}
  y_0 = x_0 \\
  y_{n+1} = y_n \nabla x_{n+1}
\end{cases}
\]
Widening: Example

Interval domain: \( L \)
\[
L = \bot \cup \{ [l, u] | [l, u] \in (\mathbb{Z} \cup \{-\infty, \infty\}) \land l \leq u \}
\]

Lattice: \( L_{[l,u]} = \{ \bot, [l,u], [l,\infty], [-\infty,u], [-\infty,\infty] \} \)

Use the following widening \( \triangledown \):
\[
[l,u] \triangledown \bot = \bot \triangledown [l,u] = [l,u]
\]
\[
[l,u] \triangledown [l',u'] = [\text{if } l' < l \text{ then } -\infty \text{ else } l, \text{ if } u' > u \text{ then } \infty \text{ else } u]
\]
Widening and Fixpoint
Narrowing

Lattice \((L, \leq)\): \(\Delta : L \times L \rightarrow L\)

- Abstract intersection operator:
  \[
  \forall x \forall y : x \cap y \leq x \Delta y
  \]
- Enforces convergence: \((x_n)_{n \geq 0}\)
  \[
  \begin{cases}
    y_0 = x_0 \\
    y_{n+1} = y_n \Delta x_{n+1}
  \end{cases}
  \]
Narrowing and Fixpoint
Patrick Cousot.
Abstract interpretation based formal methods and future challenges.
In Informatics - 10 Years Back. 10 Years Ahead., pages 138–156,

Grgoire Sutre.
Software verification, September 2008.