Static Program Analysis
Foundations of Abstract Interpretation

Sebastian Hack, Christian Hammer, Jan Reineke

Advanced Lecture, Winter 2014/15
Recap: Abstract Interpretation

- **Semantics-based** approach to program analysis
- Framework to develop **provably correct and terminating** analyses

**Ingredients:**
- **Concrete semantics:** Formalizes meaning of a program  ✓
- **Abstract semantics**  (✓)
- Both semantics defined as **fixpoints of monotone functions** over some domain ✓
- Relation between the two semantics establishing (✓) correctness
Abstract Semantics

Similar to concrete semantics:

- A complete lattice \((L^\#, \leq)\) as the domain for abstract elements
- A monotone function \(F^\#\) corresponding to the concrete function \(F\)
- Then the abstract semantics is the least fixed point of \(F^\#\), \(\text{lfp } F^\#\)

If \(F^\#\) “correctly approximates” \(F\), then \(\text{lfp } F^\# \text{“correctly approximates” } \text{lfp } F\).
Fixpoint Transfer Theorem

Let \((L, \leq)\) and \((L^\#, \leq^\#)\) be two complete lattices, \(\gamma : L^\# \to L\) a monotone function, and \(F : L \to L\) and \(F^\# : L^\# \to L^\#\) two monotone functions, with

\[
\forall l^\# \in L^\# : \gamma(F^#(l^#)) \geq F(\gamma(l^#)).
\]

Then:

\[
lfp F \leq \gamma(lfp F^#).
\]
An Example Abstract Domain for Values of Variables

(\mathcal{P}(\mathbb{Z}), \subseteq) \quad \{\ldots, -2, -1, 0, 1, 2, \ldots\}

(\mathbb{Z}_\perp, \leq)

How to relate the two?

- Concretization function, specifying “meaning” of abstract values.
  \[ \gamma : \mathbb{Z}_\perp \rightarrow \mathcal{P}(\mathbb{Z}) \]

- Abstraction function: determines best representation concrete values.
  \[ \alpha : \mathcal{P}(\mathbb{Z}) \rightarrow \mathbb{Z}_\perp \]
Relation between the Abstract and Concrete Domains

\[\begin{align*}
\gamma(\top) & := \mathbb{Z} \\
\gamma(\bot) & := \emptyset \\
\gamma(x) & := \{x\} \\
\alpha(A) & := \begin{cases} 
\top : |A| \geq 2 \\
x : A = \{x\} \\
\bot : A = \emptyset
\end{cases}
\end{align*}\]

1. Are these functions monotone?
2. Should they be?
3. What is the meaning of the partial order in the abstract domain?
4. What if we first abstract and the concretize?
\[\gamma(\alpha(x)) \geq x\]
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

Denotes abstract version of operator

<table>
<thead>
<tr>
<th></th>
<th>⊤</th>
<th>a</th>
<th>0</th>
<th>⊥</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊤</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊤</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>⊥</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to Compute in the Abstract Domain: Correctness Conditions

Correctness Condition:

Correct by construction (if concretization and abstraction have certain properties):
How to Compute in the Abstract Domain
Example: Multiplication on Flat Lattice

\[
\begin{array}{ccc}
\gamma & \gamma & \alpha \\
\{a\} & \{b\} & \{a \ast b\} \\
\end{array}
\]
How to Compute in the Abstract Domain

Example: Multiplication on Flat Lattice

\[
\begin{array}{c}
\mathbb{Z} \xrightarrow{\gamma} \{0\} \xrightarrow{\gamma} 0 \xrightarrow{\ast} 0 \\
\end{array}
\]
How to Compute in the Abstract Domain: Correct by Construction

**Correct by construction**
*(if concretization and abstraction have certain properties):*

```
    Abstract Domain
    \[ \gamma \quad \text{op}^# \quad \alpha \]
```

**“Certain properties”: Notion of Galois connection:**
Let \((L, \leq)\) and \((M, \sqsubseteq)\) be partially ordered sets and \(\alpha \in L \to M\), \(\gamma \in M \to L\). We call \((L, \leq) \leftrightarrow_{\alpha, \gamma} (M, \sqsubseteq)\) a Galois connection if \(\alpha\) and \(\gamma\) are monotone functions and

\[
\begin{align*}
    l & \leq \gamma(\alpha(l)) \\
    \alpha(\gamma(m)) & \sqsubseteq m
\end{align*}
\]

for all \(l \in L\) and \(m \in M\).
Galois connections

**Notion of Galois connections:**

Let \((L, \leq)\) and \((M, \sqsubseteq)\) be partially ordered sets and \(\alpha \in L \to M, \gamma \in M \to L\). We call \((L, \leq) \leftrightarrow_{\alpha, \gamma} (M, \sqsubseteq)\) a Galois connection if \(\alpha\) and \(\gamma\) are monotone functions and

\[
\begin{align*}
    l \leq \gamma(\alpha(l)) \\
    \alpha(\gamma(m)) \sqsubseteq m
\end{align*}
\]

for all \(l \in L\) and \(m \in M\).

**Graphically:**

\[
\begin{array}{c}
    \text{(L, \leq)} \\
    \gamma \\
    \alpha
\end{array}
\quad
\begin{array}{c}
    \text{(M, \sqsubseteq)} \\
    \theta
\end{array}
\]

**Why monotone?**

For soundness.

**For precision.**

For soundness.
Galois connections: Example

\[(\mathcal{P}(\mathbb{Z}), \subseteq) \leftrightarrow_{\alpha}^{\gamma} (\mathbb{Z}_{\bot}, \leq)\]

with:

\[\alpha : \mathcal{P}(\mathbb{Z}) \to \mathbb{Z}_{\bot} \quad \gamma : \mathbb{Z}_{\bot} \to \mathcal{P}(\mathbb{Z})\]

\[\begin{align*}
\alpha(A) := \begin{cases} 
\top & : |A| \geq 2 \\
\bot & : A = \emptyset \\
x & : A = \{x\}
\end{cases} & \gamma(\top) := \mathbb{Z} \\
\gamma(\bot) := \emptyset & \gamma(x) := \{x\}
\end{align*}\]
Galois connections: Properties

Properties:
1) Can be used to systematically construct correct (and in fact the most precise) abstract operations: \( \text{op}^\# = \alpha \circ \text{op} \circ \gamma \)
2) a) Abstraction function induces concretization function
b) Concretization function induces abstraction function

Why?
How?
How do abstraction and concretization induce each other?

\[ \gamma(a) := \bigcap \{ c \mid \alpha(c) \leq a \} \]

\[ \alpha(x) := \bigcap \{ a \mid x \leq \gamma(a) \} \]
Why is $\alpha \circ op \circ \gamma$ a correct abstract operation?
Why is $\alpha \circ \mathcal{O} \circ \gamma$ the best correct abstract transformer?

Could there not be multiple incomparable transformers?
Think of an abstraction that does not admit a Galois connection!

\[(\mathcal{P}(\mathbb{Q} \times \mathbb{Q}), \subseteq)\]

\[\mathbb{Q} \times \mathbb{Q} \times \mathbb{Q} \times \mathbb{R}\]
“Improves” abstract value without affecting meaning.

A Galois Connection is a Galois Insertion if $\alpha \circ \gamma = id$.

Where might this occur?
Abstracting Sets of Concrete States

Recap: Concrete States

Concrete states are not just sets of values...

Concrete states consist of variables and memory:

\[ s = (\rho, \mu) \in States \]

\[ \rho : \text{Vars} \to \text{int} \]

\[ \mu : \mathbb{N} \to \text{int} \]

\[ States = (\text{Vars} \to \mathbb{Z}) \times (\mathbb{N} \to \mathbb{Z}) \]
Reachability semantics is defined on sets of states:

\[
\llbracket \text{statement} \rrbracket \subseteq \text{States} \times \text{States}
\]

\[
\llbracket \text{statement} \rrbracket : \mathcal{P}(\text{States}) \rightarrow \mathcal{P}(\text{States})
\]

\[
\llbracket \text{statement} \rrbracket (S) := \{ s' \mid \exists s \in S : (s, s') \in \llbracket \text{statement} \rrbracket \}
\]

\[
\mathcal{P}(\text{States}) = \mathcal{P}(\text{Vars} \rightarrow \mathbb{Z} \times \mathbb{N} \rightarrow \mathbb{Z})
\]
Relation between Concrete Domain and Abstract Domain

Concrete domain!

\[ P(States) = P((Vars \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})) \]

Abstract domain?

\[ \widehat{States} = Vars \rightarrow \mathbb{Z}^{\uparrow} \]

Relation between the two?

→ For ease of understanding, introduce Intermediate domain:

\[ PowerSetStates = Vars \rightarrow P(\mathbb{Z}) \]
Relation between Concrete Domain and Intermediate Domain

**Concrete domain:**
\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \rightarrow \mathbb{Z}) \times (\mathbb{N} \rightarrow \mathbb{Z})) \]

**Intermediate domain:**
\[ \text{PowerSetStates} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \]

**Abstraction:**
\[ \alpha_{C,I} : \mathcal{P}(\text{States}) \rightarrow \text{PowerSetStates} \]
\[ \alpha_{C,I}(C) := \lambda x \in \text{Vars}. \{ v(x) \in \mathbb{Z} \mid (v, m) \in C \} \]

**Concretization:**
\[ \gamma_{I,C} : \text{PowerSetStates} \rightarrow \mathcal{P}(\text{States}) \]
\[ \gamma_{I,C}(\widehat{C}) := \{ (v, m) \in \text{States} \mid \forall x \in \text{Vars} : v(x) \in \widehat{c}(x) \} \]
Relation between Intermediate Domain and Abstract Domain

**Intermediate domain:**
\[ \overline{\text{PowerSetStates}} = \text{Vars} \rightarrow \mathcal{P}(\mathbb{Z}) \quad \overline{\text{States}} = \text{Vars} \rightarrow \mathbb{Z} \uparrow \]

**Abstract domain:**

**Abstraction:**
\[ \alpha_{I,A} : \overline{\text{PowerSetStates}} \rightarrow \overline{\text{States}} \]
\[ \alpha(\widehat{c}) := \lambda x \in \text{Vars}.\alpha(c(x)) \]

**Concretization:**
\[ \gamma_{A,I} : \overline{\text{States}} \rightarrow \overline{\text{PowerSetStates}} \]
\[ \gamma(\widehat{a}) := \lambda x \in \text{Vars}.\gamma(a(x)) \]

Abstraction and Concretization functions from before!

Could plug in other abstractions for sets of values…
Relation between Concrete Domain and Abstract Domain

**Concrete domain:**

\[ \mathcal{P}(\text{States}) = \mathcal{P}((\text{Vars} \to \mathbb{Z}) \times (\mathbb{N} \to \mathbb{Z})) \]

**Abstract domain:**

\[ \widehat{\text{States}} = \text{Vars} \to \mathbb{Z} \]

**Abstraction:**

\[ \alpha_{C,A} : \mathcal{P}(\text{States}) \to \widehat{\text{States}} \]

\[ \alpha_{C,A} := \alpha_{I,A} \circ \alpha_{C,I} \]

**Concretization:**

\[ \gamma_{A,C} : \widehat{\text{States}} \to \mathcal{P}(\text{States}) \]

\[ \gamma_{A,C} := \gamma_{I,C} \circ \gamma_{A,I} \]
Meaning of Statements in the Abstract Domain

\[
\begin{align*}
[R = e]^{\#} (\hat{a}) & := \hat{a}[R \mapsto [e]^{\#} (\hat{a})] \\
[R = M[e]]^{\#} (\hat{a}) & := \hat{a}[R \mapsto \top] \checkmark \\
[M[e_1] = e_2]^{\#} (\hat{a}) & := \hat{a} \checkmark \\
[Pos(e)]^{\#} (\hat{a}) & := \hat{a} \\
[Neg(e)]^{\#} (\hat{a}) & := \hat{a}
\end{align*}
\]

Can this be done better?

Again:

For Correctness:

For the best possible precision:
Meaning of Expressions

Evaluation of expressions is as expected:

\[
\begin{align*}
[x]^{\#}(\widehat{\alpha}) &= \widehat{\alpha}(x) & \text{if } x \in Vars \\
[e_1 \ op \ e_2]^{\#}(\widehat{\alpha}) &= [e_1]^{\#}(\widehat{\alpha}) \ op^{\#} [e_2]^{\#}(\widehat{\alpha})
\end{align*}
\]

As we have seen earlier!
Putting it all together:
The Abstract Reachability Semantics

Abstract Reachability Semantics captured as least fixed point of:

\[ \widehat{\text{Reach}} : V \to \text{States} \]

\[ \widehat{\text{Reach}}(\text{start}) = \top \]

\[ \forall v' \in V \setminus \{\text{start}\} : \widehat{\text{Reach}}(v') = \bigcup_{v \in V, (v,v') \in E} [\text{labeling}(v,v')] \# (\widehat{\text{Reach}}(v)) \]

\[ \widehat{\text{Reach}}(1) = \llbracket \text{labeling}(\text{start}, 1) \rrbracket \# (\widehat{\text{Reach}}(\text{start})) \cup \llbracket \text{labeling}(2, 1) \rrbracket (\widehat{\text{Reach}}(2)) \]

\[ \widehat{\text{Reach}}(2) = \llbracket \text{labeling}(1, 2) \rrbracket \# (\widehat{\text{Reach}}(1)) \]

\[ \widehat{\text{Reach}}(3) = \llbracket \text{labeling}(1, 3) \rrbracket \# (\widehat{\text{Reach}}(1)) \]

\[ \widehat{\text{Reach}}(1) = \llbracket x = 0 \rrbracket \# (\widehat{\text{Reach}}(\text{start})) \cup \llbracket x = x + 1 \rrbracket \# (\widehat{\text{Reach}}(2)) \]

\[ \widehat{\text{Reach}}(2) = \llbracket \text{Pos}(x < 100) \rrbracket \# (\widehat{\text{Reach}}(1)) \]

\[ \widehat{\text{Reach}}(3) = \llbracket \text{Neg}(x < 100) \rrbracket \# (\widehat{\text{Reach}}(1)) \]
Example: Kleene Iteration to Compute Abstract Reachability Semantics
Example: Kleene Iteration to Compute Abstract Reachability Semantics

\[
x = 0
\]

\[
\text{start}
\]

\[
\begin{align*}
&\text{Pos}(x < 100) \\
&\text{Neg}(x < 100)
\end{align*}
\]

\[
\begin{align*}
1 & \quad x = x + 1 \\
2 & \\
3 & \\
\end{align*}
\]

\[
x \rightarrow T \quad x \rightarrow \bot
\]

\[
x \rightarrow T \\
x \rightarrow 0 \\
x \rightarrow \bot
\]

\[
x \rightarrow T \\
x \rightarrow 0 \\
x \rightarrow \bot
\]

\[
x + 100
\]
Example II: Kleene Iteration to Compute Abstract Reachability Semantics

```plaintext
y = 0;
x = 1;
z = 3;
while (x > 0) {
    if (x == 1) {
        y = 7;
    }
    else {
        y = z+4;
    }
    x = 3;
    print y;
}
```
Next: Other Numerical Abstractions

- Signs
- Parity \( \varepsilon \ \text{EVEN or ODD?} \)
- Intervals
- Octagons / POLYHEDRA
- Congruence \( x \mod 2 \)