Static Program Analysis
Part 10 – abstract interpretation

http://cs.au.dk/~amoeller/spa/

Anders Møller & Michael I. Schwartzbach
Computer Science, Aarhus University
Abstract interpretation

Abstract interpretation provides a solid mathematical foundation for reasoning about static program analyses

• Is my analysis sound? (Does it safely approximate the actual program behavior?)
• Is it as precise as possible for the currently used analysis lattice? If not, where can precision losses arise? Which precision losses can be avoided (without sacrificing soundness)?

Answering such questions requires a precise definition of the semantics of the programming language, and precise definitions of the analysis abstractions in terms of the semantics
Agenda

- Collecting semantics
- Abstraction and concretization
- Soundness
- Optimality
- Completeness
Sign analysis, recap

\[
\begin{align*}
\text{Sign} & = + \cup - \cup 0 \\
\text{States} & = \text{Vars} \rightarrow \text{Sign} \\
\text{States}^n \\
af([v_1], \ldots, [v_n]) & = (af_{v_1}([v_1], \ldots, [v_n]), \ldots, af_{v_n}([v_1], \ldots, [v_n])) \\
af: \text{States}^n & \rightarrow \text{States}^n \quad \text{is the analysis representation of the given program} \\
[P] & = lfp(af)
\end{align*}
\]
Program semantics as constraint systems

\[ \text{ConcreteStates} = \text{Vars} \rightarrow \mathbb{Z} \]

\[ \{v\} \subseteq \text{ConcreteStates} \]
The semantics of expressions

\[ ceval: \text{ConcreteStates} \times \text{Exp} \to \mathcal{P}(\mathbb{Z}) \]

\[ ceval(\rho, X) = \{\rho(X)\} \]
\[ ceval(\rho, I) = \{I\} \]
\[ ceval(\rho, \text{input}) = \mathbb{Z} \]
\[ ceval(\rho, E_1 \text{ op } E_2) = \{v_1 \text{ op } v_2 \mid v_1 \in ceval(\rho, E_1) \land v_2 \in ceval(\rho, E_2)\} \]

\[ ceval(R, E) = \bigcup_{\rho \in R} ceval(\rho, E) \]
Successors and joins

\[ csucc : \text{ConcreteStates} \times \text{Nodes} \rightarrow \mathcal{P}(\text{Nodes}) \]

\[ csucc(R, v) = \bigcup_{\rho \in R} csucc(\rho, v) \]

\[ \text{CJOIN}(v) = \{ \rho \in \text{ConcreteStates} \mid \exists w \in \text{Nodes} : \rho \in \{w\} \land v \in csucc(\rho, w) \} \]
Semantics of statements

\[
[X=E] = \{ \rho[X \mapsto z] \mid \rho \in CJOIN(v) \land z \in ceval(\rho, E) \}
\]

\[
[var \, X_1, \ldots, X_n] = \\
\{ \rho[X_1 \mapsto z_1, \ldots, X_n \mapsto z_n] \mid \rho \in CJOIN(v) \land z_1 \in \mathbb{Z} \land \cdots \land z_n \in \mathbb{Z} \}
\]

\[
[ entry ] = \{ [] \}
\]

\[
[v] = CJOIN(v)
\]
The resulting constraint system

\[
\{v_1\} = cf_{v_1}(\{v_1\}, \ldots, \{v_n\})
\]
\[
\{v_2\} = cf_{v_2}(\{v_1\}, \ldots, \{v_n\})
\]
\[
\vdots
\]
\[
\{v_n\} = cf_{v_n}(\{v_1\}, \ldots, \{v_n\})
\]

\[
cf: (\mathcal{P}(\text{ConcreteStates}))^n \rightarrow (\mathcal{P}(\text{ConcreteStates}))^n
\]
\[
cf(x_1, \ldots, x_n) = (cf_{v_1}(x_1, \ldots, x_n), \ldots, cf_{v_n}(x_1, \ldots, x_n))
\]

is the semantic representation of the given program

\[
x = cf(x)
\]
\[
\{P\} = \text{lfp}(cf)
\]
Example

```javascript
var x;
x = 0;
while (input) {
    x = x + 2;
}
```

<table>
<thead>
<tr>
<th></th>
<th>solution 1</th>
<th>solution 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{entry}</code></td>
<td><code>{}</code></td>
<td><code>{}</code></td>
</tr>
<tr>
<td><code>{var x}</code></td>
<td>`{{x → z}</td>
<td>z ∈ \mathbb{Z}}`</td>
</tr>
<tr>
<td><code>{x = 0}</code></td>
<td><code>{{x → 0}}</code></td>
<td><code>{{x → 0}}</code></td>
</tr>
<tr>
<td><code>{input}</code></td>
<td>`{{x → z}</td>
<td>z ∈ {0, 2, 4, ...}}`</td>
</tr>
<tr>
<td><code>{x = x + 2}</code></td>
<td>`{{x → z}</td>
<td>z ∈ {2, 4, ...}}`</td>
</tr>
<tr>
<td><code>{exit}</code></td>
<td>`{{x → z}</td>
<td>z ∈ {0, 2, 4, ...}}`</td>
</tr>
</tbody>
</table>

The least solution
Kleene’s fixed point theorem for complete join morphisms

If \( f : L \to L \) is a complete join morphism if

\[
f(\bigsqcup A) = \bigsqcup_{a \in A} f(a)
\]

for every \( A \subseteq L \).

If \( f \) is a complete join morphism:

\[
\text{lfp}(f) = \bigsqcup_{i \geq 0} f^i(\bot)
\]

(even when \( L \) has infinite height!)

cf is a complete join morphism
Tarski’s fixed-point theorem

In a complete lattice $L$, every monotone function $f : L \to L$ has a unique least fixed point given by $\bigcap \{ x \in L \mid f(x) \subseteq x \}$.

$cf$ is monotone
Semantics vs. analysis

\[
\begin{align*}
\{b = 87\} &= \{[a \mapsto 42, b \mapsto 87, c \mapsto z] \mid z \in \mathbb{Z}\} \\
\{c = a - b\} &= \{[a \mapsto 42, b \mapsto 87, c \mapsto -45]\} \\
\{\text{exit}\} &= \{[a \mapsto 42, b \mapsto 87, c \mapsto 129], [a \mapsto 42, b \mapsto 87, c \mapsto -45]\}
\end{align*}
\]

\[
\begin{align*}
[b = 87] &= [a \mapsto +, b \mapsto +, c \mapsto T] \\
[c = a - b] &= [a \mapsto +, b \mapsto +, c \mapsto T] \\
\{\text{exit}\} &= [a \mapsto +, b \mapsto +, c \mapsto T]
\end{align*}
\]

```javascript
var a, b, c;
a = 42;
b = 87;
if (input) {
c = a + b;
} else {
c = a - b;
}
```
Agenda

- Collecting semantics
- Abstraction and concretization
- Soundness
- Optimality
- Completeness
Abstraction functions for sign analysis

\[\alpha_a : \mathcal{P}(\mathbb{Z}) \rightarrow \text{Sign}\]
\[\alpha_b : \mathcal{P}(\text{ConcreteStates}) \rightarrow \text{States}\]
\[\alpha_c : (\mathcal{P}(\text{ConcreteStates}))^n \rightarrow \text{States}^n\]

\[\alpha_a(D) = \begin{cases} 
\bot & \text{if } D \text{ is empty} \\
+ & \text{if } D \text{ is nonempty and contains only positive integers} \\
- & \text{if } D \text{ is nonempty and contains only negative integers} \\
0 & \text{if } D \text{ is nonempty and contains only the integer } 0 \\
\top & \text{otherwise} 
\end{cases}\]

for any \( D \in \mathcal{P}(\mathbb{Z}) \)

\[\alpha_b(R) = \sigma \quad \text{where } \sigma(X) = \alpha_a(\{\rho(X) \mid \rho \in R\})\]
for any \( R \subseteq \text{ConcreteStates} \) and \( X \in \text{Vars} \)

\[\alpha_c(R_1, \ldots, R_n) = (\alpha_b(R_1), \ldots, \alpha_b(R_n))\]
for any \( R_1, \ldots, R_n \subseteq \text{ConcreteStates} \)
Concretization functions for sign analysis

\[ \gamma_a : \text{Sign} \rightarrow \mathcal{P}(\mathbb{Z}) \]
\[ \gamma_b : \text{States} \rightarrow \mathcal{P}(\text{ConcreteStates}) \]
\[ \gamma_c : \text{States}^n \rightarrow (\mathcal{P}(\text{ConcreteStates}))^n \]

\[ \gamma_a(s) = \begin{cases} 
\emptyset & \text{if } s = \perp \\
\{1, 2, 3, \ldots \} & \text{if } s = + \\
\{-1, -2, -3, \ldots \} & \text{if } s = - \\
\{0\} & \text{if } s = \emptyset \\
\mathbb{Z} & \text{if } s = \top 
\end{cases} \]
for any \( s \in \text{Sign} \)

\[ \gamma_b(\sigma) = \{ \rho \in \text{ConcreteStates} \mid \rho(X) \in \gamma_a(\sigma(X)) \text{ for all } X \in \text{Vars} \} \]
for any \( \sigma \in \text{States} \)

\[ \gamma_c(\sigma_1, \ldots, \sigma_n) = (\gamma_b(\sigma_1), \ldots, \gamma_b(\sigma_n)) \]
for any \( (\sigma_1, \ldots, \sigma_n) \in \text{States}^n \)
Monotonicity of abstraction and concretization functions

Concretization functions are, like abstraction functions, naturally monotone.

(A larger set of concrete values should correspond to a larger abstract state, and conversely)
Galois connections

The pair of monotone functions, $\alpha$ and $\gamma$, is called a Galois connection if

$\gamma \circ \alpha$ is extensive

$\alpha \circ \gamma$ is reductive

all three pairs of abstraction and concretization functions $(\alpha_a, \gamma_a)$, $(\alpha_b, \gamma_b)$, and $(\alpha_c, \gamma_c)$ from the sign analysis example are Galois connections
Galois connections

For Galois connections, the concretization function uniquely determines the abstraction function and vice versa:

\[
\gamma(y) = \bigcup \{x \in L_1 \mid \alpha(x) \subseteq y\}
\]

\[
\alpha(x) = \bigcap \{y \in L_2 \mid x \subseteq \gamma(y)\}
\]
Galois connections

For this lattice, given the “obvious” concretization function, is there an abstraction function such that the concretization function and the abstraction function form a Galois connection?

how should we define $\alpha_a(\{0\})$?
Representation functions

\[ \beta : \mathbb{Z} \rightarrow \text{Sign} \]

\[ \beta(d) = \begin{cases} 
+ & \text{if } d > 0 \\
- & \text{if } d < 0 \\
0 & \text{if } d = 0 
\end{cases} \]

\[ \alpha_a(D) = \{ \beta(d) \mid d \in D \} \]
Agenda

- Collecting semantics
- Abstraction and concretization
- Soundness
- Optimality
- Completeness
Soundness

\[ \alpha(\{P\}) \subseteq [P] \]

\[
\begin{align*}
\mathcal{P}(\text{ConcreteStates})^n & \quad \text{States}^n \\
\{P\} & \quad [P]
\end{align*}
\]
Soundness

\[ \{P\} \subseteq \gamma([P]) \]

Diagram:

- \(\{P\}\) and \([P]\) are connected by \(\gamma_c\).
- \((\mathcal{P}(\text{Concrete States}))^n\) and \(\text{States}^n\) are connected.
Sound abstractions

\[ \alpha_a(ceval(R, E)) \subseteq eval(\alpha_b(R), E) \]

\[ csucc(R, v) \subseteq succ(v) \text{ for any } R \subseteq \text{ConcreteStates} \]

\[ \alpha_b(CJOIN(v)) \subseteq JOIN(v) \]

if \( \alpha_b([w]) \subseteq [w] \) for all \( w \in \text{Nodes} \).
Sound abstractions

if \( v \) represents an assignment statement \( X = E \) : 

\[
cf_v(\{v_1\}, \ldots, \{v_n\}) = \{ \rho[X \mapsto z] \mid \rho \in CJOIN(v) \land z \in ceval(\rho, E) \}
\]

\[
af_v(\langle v_1 \rangle, \ldots, \langle v_n \rangle) = \sigma[X \mapsto eval(\sigma, E)] \text{ where } \sigma = JOIN(v)
\]

\[
\alpha_b(cf_v(R_1, \ldots, R_n)) \subseteq af_v(\alpha_b(R_1), \ldots, \alpha_b(R_n))
\]
The two constraint systems

\[ cf(\{v_1\}, \ldots, \{v_n\}) = (cf_{v_1}(\{v_1\}, \ldots, \{v_n\}), \ldots, cf_{v_n}(\{v_1\}, \ldots, \{v_n\})) \]

\[ af(\{v_1\}, \ldots, \{v_n\}) = (af_{v_1}(\{v_1\}, \ldots, \{v_n\}), \ldots, af_{v_n}(\{v_1\}, \ldots, \{v_n\})) \]
Sound abstractions

\[ \alpha_c(cf(R_1, \ldots, R_n)) \subseteq af(\alpha_c(R_1, \ldots, R_n)) \]
Sound abstractions

\alpha \circ cf \subseteq af \circ \alpha

cf \circ \gamma \subseteq \gamma \circ af

Equivalent, if \alpha and \gamma form a Galois connection
If $L_1$ and $L_2$ are complete lattices with a concretization function $\gamma: L_2 \to L_1$, $cf: L_1 \to L_1$ and $af: L_2 \to L_2$ are monotone, and $af$ is a sound abstraction of $cf$ with respect to $\gamma$, i.e., $cf \circ \gamma \sqsubseteq \gamma \circ af$, then $\text{lfp}(cf) \sqsubseteq \gamma(\text{lfp}(af))$.\[\]
Agenda

• Collecting semantics
• Abstraction and concretization
• Soundness
• Optimality
• Completeness
Optimal approximations

\( af \) is an optimal approximation of \( cf \) if

\[
af = \alpha \circ cf \circ \gamma
\]

\((\mathcal{P}(\text{ConcreteStates}))^n\) \hspace{1cm} States^n
Optimal approximations in sign analysis?

\[ s_1 \hat{\star} s_2 = \alpha_a \left( \gamma_a(s_1) \cdot \gamma_a(s_2) \right) \]

* eval is not optimal: 
\[ \sigma(x) = \top \]
\[ \text{eval}(\sigma, x-x) = \top \]
\[ \alpha_b \left( \text{ceval}(\gamma_b(\sigma), x-x) \right) = 0 \]

Even if we could make \textit{eval} optimal, the analysis result is not always optimal:

\[
\begin{align*}
x &= \text{input}; \\
y &= x; \\
z &= x - y;
\end{align*}
\]
Agenda

• Collecting semantics
• Abstraction and concretization
• Soundness
• Optimality
• **Completeness**
Completeness

\[ [P] \subseteq \alpha([P]) \]

Sound and complete: \( \alpha([P]) = [P] \)

(Intuitively, the analysis result is the most precise possible for the currently used lattice)

Not the same as \( \{P\} = \gamma([P]) \) (called “exact”)

(Intuitively, the analysis result exactly captures the semantics of the program)
Completeness in sign analysis?

\( \hat{\star} \) is complete:

\[ \alpha_a(D_1) \hat{\star} \alpha_a(D_2) \subseteq \alpha_a(D_1 \cdot D_2) \]

\( \hat{\dagger} \) is not complete

\[ \alpha_a(D_1) \hat{\dagger} \alpha_a(D_2) \nsubseteq \alpha_a(D_1 + D_2) \]

Sign analysis is sound and complete for some programs, but not for all programs.
Conclusions

Abstract interpretation provides a solid mathematical foundation for reasoning about soundness and precision of static program analyses.

We need:

- the static analysis (the analysis lattices and constraint rules)
- the language semantics (a suitable collecting semantics)
- abstraction/concretization functions that specify the meaning of the elements in the analysis lattice in terms of the semantic lattice

... and then:

- if each constituent of the analysis is a sound abstraction of its semantic counterpart, then the analysis is sound (according to the soundness theorem)
- if an abstraction is optimal, then it is as precise as possible (yet sound), relative to the choice of analysis lattice
- if the analysis is sound and complete, then the analysis result is as precise as possible (yet sound), relative to the choice of analysis lattice