Abstract Interpretation, cont.

Announcements

- HW3…
- HW4 is optional
- HW5 (proofs with Abstract Interpretation and Haskell) coming up on Friday
  - Please start early
  - I promise no more extensions!

Outline

- Paper presentations
  - Give 20-25 minute presentation
  - Email me a list of papers you would like to see
  - I will post a list papers to choose from

Towards Concretization and Abstraction Functions

- Abstraction relation is consistent with order!
  - Concrete order:
    - If $c_0 \subseteq c_1$ and $c_1$ is represented by $a$, then $c_0$ is represented by $a$
  - Abstract order:
    - If $a_s \leq a_t$ and $c$ is represented by $a_s$, then $c$ is represented by $a_t$
    - Advantage: reasoning in abstract space proceeds by going up the abstract lattice; property preserves the abstraction relation

Abstraction Relation is Consistent with Partial Orders!

Concrete lattice:

Abstract lattice:
Abstraction Relation is Consistent with Partial Orders!

Towards Concretization and Abstraction Functions

Concretization Function

- **Definition:**
  Concretization function $\gamma : A \rightarrow C$ (if it exists) maps $a \in A$ to the largest (most general) element in $c \in C$ such that $c \preceq a$

  Note: $\gamma(a)$ “covers” all concrete elements that are represented by $a$.

  $\gamma(a)$ returns the most general element $c$ such that $c$ is represented by $a$. This is called concretization

Gamma Examples

- Concretization of lattice of signs
  - $\gamma_S(T) \rightarrow \mathbb{Z}$
  - $\gamma_S(\pm) \rightarrow \{1, 2, 3, \ldots\}$
  - $\gamma_S(C) \rightarrow \{\ldots, -3, -2, -1\}$
  - $\gamma_S(0) \rightarrow \{0\}$
  - $\gamma_S(\perp) \rightarrow \{\}$

Concretization of lattice of intervals
  - $\gamma_I([a,b]) \rightarrow \{a, a+1, \ldots, b-1, b\}$
  - $\gamma_I(T)$ etc.

Abstraction Function

- **Definition:**
  Abstraction function $\alpha : C \rightarrow A$ (if it exists) maps $c \in C$ to the smallest (most precise) element in $a \in A$ such that $c \preceq a$

  $\alpha$ maps $c$ to the most precise $a$ such that $a$ represents $c$. This is called best abstraction
**Alpha Examples**

Concrete lattice:

\{-100, -2, -1\} \rightarrow \{0.1, 2\} \rightarrow \{0\} \rightarrow ∪

Abstract lattice:

\{-2, -1\} \rightarrow \{1.2\} \rightarrow \{0\} \rightarrow T

---

**Abstraction Function Examples**

- **Signs abstraction**
  - \(α_S(c) \rightarrow \bot\) if \(c = \{\}\\)
  - \(α_S(c) \rightarrow 0\) if \(c = \{0\}\)
  - \(α_S(c) \rightarrow +\) if for every \(n \in c, n > 0\)
  - \(α_S(c) \rightarrow -\) if for every \(n \in c, n < 0\)
  - \(α_S(c) \rightarrow T\) otherwise

- **Constants abstraction**
  - \(α_C(c) \rightarrow \{\}\) if \(c = \{\}\\)
  - \(α_C(c) \rightarrow n\) if \(c = \{n\}\)
  - \(α_C(c) \rightarrow T\) otherwise

---

**Outline**

- Abstract interpretation
- Semantics
- The abstraction relation
- Concretization and abstraction functions
- **Galois Connections**
- Applications of abstract interpretation
- Termination

---

**Galois Connection**

- A Galois Connection links \(α\) and \(γ\) so that they refer to the same abstraction relation \(\sqsubseteq\)

**Definition**

A **Galois connection** is defined by concrete lattice \((C, \sqsubseteq)\), abstract lattice \((A, \leq)\), an abstraction function \(α : C → A\) and concretization function \(γ : A → C\) such that

\[c \sqsubseteq γ(a)\] if and only if \(α(c) \leq a\)

For every \(a \in A\) and every \(c \in C\)

\[c \sqsubseteq γ(a)\] if and only if \(α(c) \leq a\)

If \(c = \{\}\), then for every \(a\), \(\{\} \subseteq γ(a)\) and \(α(\{\}) \leq a\)

If \(c = \{n\}\), then \(c \subseteq γ(n)\) or \(\{n\} \subseteq γ(a)\) and \(α(\{n\}) \leq a\)

If \(c = \{\}\), then \(c \subseteq γ(\bot)\) and \(α(c) \leq a\)

If \(c = \{n\}\), then \(c \subseteq γ(n)\) or \(\{n\} \subseteq γ(a)\) and \(α(c) \leq a\)

If \(c = \{\}\), then \(c \subseteq γ(T)\) and \(α(c) \leq a\)

---

**Galois Connection Example**

**Constants lattice**

\(α_C(c) \rightarrow \bot\) if \(c = \{\}\\)

\(γ_C(T) \rightarrow Z\)

\(α_C(c) \rightarrow n\) if \(c = \{n\}\)

\(γ_C(n) \rightarrow \{n\}\)

\(α_C(c) \rightarrow T\) otherwise

\(γ_C(\bot) \rightarrow \{\}\)

If \(c = \{\}\), then for every \(a\), \(\{\} \subseteq γ(a)\) and \(α(\{\}) \leq a\)

If \(c = \{n\}\), then \(c \subseteq γ(\{n\})\) or \(\{n\} \subseteq γ(a)\) and \(α(\{n\}) \leq a\)

In either case equivalence holds

If \(c = \{\}\), then \(c \subseteq γ(T)\) or \(\{\} \subseteq γ(a)\) only for \(a = T\)

(Since \(c\) has at least 2 elements.) Then \(c \subseteq γ(T)\) and \(α(c) \leq T\) hold
**Galois Connection Example**

- Signs lattice
  - \( a_C(c) \rightarrow + \) if for every \( n \) in \( c \), \( n > 0 \)
  - \( a_C(c) \rightarrow - \) if for every \( n \) in \( c \), \( n < 0 \)
  - \( a_C(c) \rightarrow 0 \) if \( c = \{0\} \)
  - \( a_C(c) \rightarrow T \) if \( c = \{\} \)

**Galois Connection Properties**

- for every \( a \in A \) and every \( c \in C \)
  - \( c \subseteq \gamma(a) \) if and only if \( \alpha(c) \leq a \)

  - **Contractive and expansive:**
    - \( \alpha \circ \gamma \) contracts: for every \( a \in A \): \( \alpha \circ \gamma(a) \leq a \)
    - \( \gamma \circ \alpha \) expands: for every \( c \in C \): \( \gamma \circ \alpha(c) \supseteq c \)

  - Proof: from definition of Galois Connection:
    - \( \gamma(a) \subseteq \gamma(a) \) implies \( \alpha(\gamma(a)) \leq a \)

- **Monotonicity:**
  - \( \alpha \) is monotone
  - \( \gamma \) is monotone

  Proof: exercise

**Other Properties**

- **Distributivity** (preservation of upper bounds)
  - For every \( c_0, c_1 \) in \( C \), \( \alpha(c_0 \lor c_1) = \alpha(c_0) \lor \alpha(c_1) \)

- **Uniqueness**
  - For a given \( \gamma : A \rightarrow C \), there exists at most one \( \alpha : C \rightarrow A \) such that \( \gamma \) and \( \alpha \) form a Galois connection
  - For a given \( \alpha : C \rightarrow A \), there exists at most one \( \gamma : A \rightarrow C \) such that \( \gamma \) and \( \alpha \) form a Galois connection

**Outline**

- Abstract interpretation
- Semantics
- The abstraction relation
- Concretization and abstraction functions
- Galois Connections
- Applications of abstract interpretation
- Termination
Applications of Abstract Interpretation
- Deriving and reasoning about static analysis!
- Deriving a static analysis
- The hard (ad-hoc) way...
  - Starting from some concrete space and semantics
  - Define abstract space and abstract semantics
  - Guess an invariant
  - Make an inductive argument that each transfer function preserves the invariant
  - Invariant must imply correctness (soundness) conclusion

An Example: The Semantics
- We will derive a static analysis that computes
- Signs abstraction: our running example
- Abstract state \( \hat{S} \) is a mapping from integer variables to values in \( \mathbb{Z} \)
- Concrete state \( S \) is a set of \( \sigma \)'s
- \[ \llbracket x = a \rrbracket (S) = \{ \sigma | (x \in \mathbb{Z}, \sigma) \in S \} \]
- \[ \llbracket x = y + z \rrbracket (S) = \{ \sigma | (x \in \mathbb{Z}, \sigma+y+z) \in S \} \]
- \[ \llbracket x = y \rrbracket (S) = \{ (x, y) | (x, y) \in S \} \]

The Abstraction
- \( \alpha \) is a composition of two abstractions
- Collecting abstraction: collects the values of a variable across all \( \sigma \)'s into one set
- Signs abstraction: our running example
- \[ \alpha : S \rightarrow (a_S (\llbracket n_x \rrbracket (n_{x\in\mathbb{Z}}) \in S)), \]
  \[ a_S (\llbracket n_y \rrbracket (n_{y\in\mathbb{Z}}) \in S)), \]
  \[ a_S (\llbracket n_z \rrbracket (n_{z\in\mathbb{Z}}) \in S)) \]
- E.g. \( S = \{ (1,2,0), (-1,3,0) \} \)
- \( \alpha(S) = (a_S ((1,1)), a_S ((2,3)), a_S ((0))) = (T,1,0) \)
- I.e., in the abstract, \( x \) is \( T \), \( y \) is \( 1 \), \( z \) is \( 0 \)

Deriving the Abstract Semantics
- We can derive an abstract semantics by "mechanical" transformation:
  - Applying \( \alpha \) on the result of a concrete transfer function \( [\text{Stmt}] \) (i.e., \( \alpha([\text{Stmt}])/[\hat{S}] \)) derives an abstract transfer function \( [\text{Stmt}] [\hat{S}] \)
  - Theorem: \( \alpha([\text{Stmt}])/[\hat{S}] = \alpha([\text{Stmt}])/[\hat{S}] \) implies static analysis is sound (i.e., computes an over-approximation)
  - The exact alpha-based transformation does not always work

\[ \hat{S} \text{-hat is standard notation for abstract state} \]
The Abstract Semantics

- Deriving the abstract semantics:
  - Abstracting result of concrete transfer function:
    \[ a^\ast \langle \text{Stmt} \rangle (S) = a \left( \{ n_x \mid n_x \in \langle \text{Stmt} \rangle (S) \} \right), \]
    \[ a \left( \{ n_y \mid n_y \in \langle \text{Stmt} \rangle (S) \} \right), \]
    \[ a \left( \{ \ldots \} \right) \in \langle \text{Stmt} \rangle (S) \}

Read: apply concrete transfer function \[ \langle \text{Stmt} \rangle (S) \] resulting in a new set of \( \sigma = (n_x, n_y, n_z) \)’s. Then

1. apply collecting abstraction: collect all values of \( x, y, \) and \( z \) into respective sets,
2. apply sign abstraction on each set

Guarantees soundness!

The Abstract Semantics

- For each concrete transfer function we derive the corresponding abstract transfer function:
  \[ a^\ast \langle x=n \rangle (S) = \]
  \[ a_s(\{ n_x \mid n_x, n_y, n_z \in \{ \sigma[x\leftarrow n] \mid \sigma \in S \} \}), \]
  \[ a_s(\{ n_y \mid n_x, n_y, n_z \in \{ \sigma[x\leftarrow n] \mid \sigma \in S \} \}), \]
  \[ a_s(\{ n_z \mid (n_x, n_y, n_z) \in \{ \sigma[x\leftarrow n] \mid \sigma \in S \} \}) = \]
  \[ a_s(\{ n_x \mid n_y, n_z \in \{ \sigma[x\leftarrow n] \mid \sigma \in S \} \}), a_s(\{ n_y \mid \{ \ldots \} \in S \}), a_s(\{ n_z \mid \{ \ldots \} \in S \}) \]
  \[ = a(S[x\leftarrow a_s(n)]) = \]
  \[ \langle x=n \rangle (S) = \hat{S}[x\leftarrow a_s(n)] \]

Abstract Interpretation and Dataflow Analysis

- Factorial
  \[ L_X: \]
  \[ T \]

\[ \begin{array}{cccccccc}
   0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
   1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
   0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
   1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
   1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
   1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
   1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
   1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\end{array} \]

Prove: applying abstract transfer function on abstract value \( a(S) \) yields same result as applying concrete transfer function on \( S \), then applying \( a \) on the result: \[ a^\ast a^\ast \langle \text{Stmt} \rangle \]

Derive abstract transfer functions, then apply fixpoint iteration.
Abstract Interpretation is Even More General!

- \( |\text{Stmt}| \circ \alpha \) s.t. \( |\text{Stmt}| = |\text{Stmt}| \circ \alpha \) may not exist for some \( |\text{Stmt}| \)
- For some abstract domains, \( \alpha \) does not exist!
- For some abstract domains, the lattice does not have finite height!
- Abstract Interpretation allows building analyses even in cases like these. Uses \( \gamma \):
  - \( |\text{Stmt}| \circ \gamma (\hat{S}) \subseteq |\text{Stmt}|(\hat{S}) \)
  - Sound (over-approximation) but less precise

Widening

- What if the abstract lattice does not have finite height?
- E.g., lattice of intervals, a popular abstract domain

<table>
<thead>
<tr>
<th>Interval</th>
<th>Interval</th>
<th>Interval</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>([-3,-2])</td>
<td>([-2,-1])</td>
<td>([-1,0])</td>
<td>([0,1])</td>
</tr>
<tr>
<td>([1,2])</td>
<td>([2,3])</td>
<td>([3,4])</td>
<td>([4,5])</td>
</tr>
</tbody>
</table>

Abstract Interpretation, Conclusion

- A general framework
- Building static analyses!
- Reasoning about correctness of static analysis
- Comparing static analyses

Next Time

- Types and Type Based Analysis (plan for next two weeks or so)
  - Pure Lambda calculus
  - Typed Lambda calculus
  - Type soundness:
    - Simple type inference
    - Hindley-Milner type inference and polymorphism

Abstract Interpretation, Conclusion

- Active area of research
- New applications of abstract interpretation
- Proof assistants
- New abstract domains
- Faster analysis techniques