Announcements

- HW3 due on Thursday. Any questions?
- HW4 out today (Optional)
  - 3 larger benchmarks
  - Run RTA and XTA including library classes
  - Compare RTA vs. XTA call graphs
  - Write a 1-2 page summary of your results
- HW4 is a team assignment

Abstract Interpretation

- Patrick Cousot and Radhia Cousot, POPL’77
- A general framework
  - Building static analyses
  - Reasoning about correctness of static analysis
  - Comparing static analyses
- Combines ideas from dataflow analysis (monotone frameworks and fixpoint iteration) and formal verification (axiomatic semantics)

Lecture Notes Based On

- “Principles of Program Analysis” by Nielsen, Nielsen and Hankin, Chapter 3
  - Alex Salcianu’s friendlier account of Chapter 3: https://web.eecs.umich.edu/~bchandra/courses/papers/Salcianu_AbstractInterpretation.pdf
- Lecture notes by Xavier Rival, ENS

AI Overview

Program Execution:

x \to h_i; A

x = y.n(z)
passes value of z to parameter p

Points-to Analysis (PTA):

x \to o_i; A

x = y.n(z)
pts(z) \subseteq pts(p)

Points-to analysis is an abstraction. Abstracts infinitely many heap objects h_i created at site i into a single o_i.
Small-step Operational Semantics

- Also called trace semantics, or concrete semantics, models a trace of execution
- Memory state maps variables (V) to values (Z): \( \sigma : V \rightarrow Z \)
- Control state describes where we are
  - label \( \ell \) (note: we used the term program point)
  - Describes transition \( \langle \ell_1, \sigma_1 \rangle \rightarrow \langle \ell_2, \sigma_2 \rangle \)
  - (read: program executes statement at label \( \ell_1 \) on current state \( \sigma_1 \), transitioning to label \( \ell_2 \) in state \( \sigma_2 \))
- Also called, \( \sigma \) denotes state, also called value semantics
- Describes transition \( \langle \ell, \sigma \rangle \rightarrow \langle \ell', \sigma' \rangle \)
- Models a trace of execution

A Simple Imperative Language: Syntax (We’ve Seen This Before!)

- Simple expression: \( E ::= x | n \)
- Assignment: \( \langle \ell, \sigma \rangle \rightarrow \langle \ell', \sigma' \rangle \)
- Loop: \( \ell : \text{while} \; (b) \; \text{Seq} \)
- Conditional: \( \ell : \text{if} \; (b) \; \{ \ell_1 \} \; \ell_2 \)
- Sequence: \( \ell : \{ \ell_1; \; \ell_2 \} \)
- \( V \) is the set of program variables, \( x \in V \)
- \( Z \) is the set of values variables take, \( n \in Z \)

A Simple Imperative Language: Operational Semantics

- Operational semantics of expressions:
  - \( [n](\sigma) = n \) // constant \( n \) evaluates to \( n \)
  - \( [x](\sigma) = \sigma(x) \) // variable \( x \) evaluates to the value \( n \) that \( x \) maps to in \( \sigma \)
- Assignment: \( \ell : x = E; \quad \ell : \ldots \)
  - \( (\ell, \sigma) \rightarrow (\ell', \sigma[x \leftarrow [E](\sigma)]) \)
- Assignment: \( \ell : x = E_1 \; \text{Op} \; E_2 ; \quad \ell : \ldots \)
  - \( (\ell, \sigma) \rightarrow (\ell', \sigma[x \leftarrow [[E_1](\sigma) \; \text{Op} \; [[E_2](\sigma)])] \)

Collecting Semantics

- Collects all states (i.e., \( \sigma \)'s) a program can have at a given label (i.e., program point)
  - E.g., variable \( x \) cannot ever be null at \( \ell \)
  - Variable \( y \) is always greater than \( 100 \) at \( \ell \)
- Given a label, \( \ell \), we are interested in a function
  - \( \ell : \text{Labels} \rightarrow 2^Z \)
  - The set of all states a program can have at \( \ell \)
Collecting Semantics

- “Ground truth”
  - We base reasoning about correctness (soundness) of static analysis off of it
- Undecidable
- Relation to MOP solution?
- Define abstraction of state and semantics
- Goal: show that abstraction “properly represents” all values computed by the collecting semantics

Outline

- Semantics
  - Notion of abstraction
    - Concretization and abstraction functions
    - Galois Connections
  - Applications of abstract interpretation

Abstraction Example 1: signs

- Concrete values: sets of integers
- Abstract values: signs

Lattice of signs:

\[
\begin{align*}
\mathbb{T} & \quad \text{represents the empty set} \\
+ & \quad \text{represents any set of positive integers} \\
0 & \quad \text{represents set \{0\}} \\
- & \quad \text{represents any set of negative integers} \\
\mathbb{T} & \quad \text{represents any set of integers}
\end{align*}
\]

Abstraction relation relates concrete elements to abstract ones: \( c \overset{S}{\rightarrow} a \) (i.e., \( a \) represents \( c \), or conversely \( c \) is represented by \( a \))

\[
\begin{align*}
\{1,2,3\} & \overset{S}{\rightarrow} + \\
\{1,2,3\} & \overset{S}{\rightarrow} \mathbb{T}
\end{align*}
\]
Abstraction Example 1: signs

- We use the abstraction relation to define 
  abstract semantics, i.e., the execution of 
  program statements over abstract elements

- If \( x \) is a positive integer and \( y \) is a positive integer, then \( x + y \) is a positive integer.
- Therefore, the concrete value of \( x + y \) is a positive integer too, thus represented by +.

Abstraction Example 1: signs

- If \( x \) is + and \( y \) is + then \( x + y \) is +.
- Analysis computes over abstract elements.
- Correctness conclusion, informally: if analysis (works on abstract elements) determines that \( x \) at label \( l \) is a, then a represents the set of concrete values \( c \) collected by the collecting semantics for \( x \) at \( l \).

Abstraction Example 2: constants

- Concrete elements: elements of concrete lattice, \( c \in \mathbb{Z}^2 \)
- Abstract elements: \( \perp, T, n \), where \( n \in \mathbb{Z} \)
  - Flat lattice:
  - Abstraction relation:
    - \( c \vdash n \) if and only if \( c = \{ n \} \)
    - empty set is represented by \( \perp \)
    - any set of integers is represented by T

Abstraction Example 2: constants

- Abstract semantics, works on abstract elements (the elements of the flat lattice)
- If \( x \) is \( n_1 \) and \( y \) is \( n_2 \), then \( x + y \) is \( n_1 + n_2 \)
  - \( n_1 \) represents exactly integer \( n_1 \),
  - \( n_2 \) represents \( n_2 \),
  - then \( x + y \) is \( n_1 + n_2 \), represented by \( n_1 + n_2 \).
- If \( x \) is \( n_1 \) and \( y \) is T, then \( x + y \) is T.
  - Since T represents any set, we cannot do better in the abstract but T.
Abstraction Example 3: intervals

- Concrete elements: elements of $2^\mathbb{Z}$
- Abstract elements: $\bot$, $\top$, intervals $[a,b]$ where $a \in \mathbb{Z} \cup \{-\infty\}$ and $b \in \mathbb{Z} \cup \{\infty\}$ and $a \leq b$
  - Is it a lattice?
    - Yes!
- Abstraction relation:
  - $\emptyset \vdash \bot$
  - $S \vdash \top$
  - $S \vdash [a,b]$ iff for every $n \in S$, $a \leq n \leq b$

Abstraction Example 4: heap

- We need to expand our syntax and operational semantics to model the heap
  - Realistic analyses of imperative languages do need to model heap
- “Collection of objects” view (Java):
  - $s \in S : \text{Id} \rightarrow \text{Addr}$ // Stack
  - $h \in H : \text{Addr} \times \text{Id} \rightarrow \text{Addr}$ // Heap

Abstract semantics

- If $x_1$ is $[a_1,b_1]$ and $x_2$ is $[a_2,b_2]$ then $x_1 + x_2$ is?
- If $x_1$ is $[a_1,b_1]$ and $x_2$ is $[a_2,b_2]$ then $x_1 \cup x_2$ is?
- If $x_1$ is $[a_1,b_1]$ and $x_2$ is $[a_2,b_2]$ then $x_1 \cap x_2$ is?

Outline

- Semantics
- Notion of abstraction
- Concretization and abstraction functions
- Galois Connections
- Applications of abstract interpretation
Towards Concretization and Abstraction Functions

- Abstraction relation is consistent with order!

- Concrete order:
  - If $c_0 \leq c_1$ and $c_1$ is represented by $a$, then $c_0$ is represented by $a$

- Abstract order:
  - If $a_0 \leq a_1$ and $c$ is represented by $a_0$, then $c$ is represented by $a_1$

Abstraction Relation is Consistent with Partial Orders!

Concrete lattice:

Abstract lattice:

Towards Concretization and Abstraction Functions

- Previous slides, more formally
- Concrete lattice $C$, and abstract lattice $A$, $\leq$
- Abstraction relation is consistent with ordering:
  - For every $c_0, c_1 \in C$ and every $a \in A$,
  - $c_0 \subseteq c_1$ and $c_1 \leftarrow a \Rightarrow c_0 \leftarrow a$
  - For every $a_0, a_1 \in A$ and every $c \in C$,
  - $a_0 \leq a_1$ and $c \leftarrow a_0 \Rightarrow c \leftarrow a_1$
- The abstraction relation makes sense but is inconvenient. We need functions!
  - Concretization function: $A \rightarrow C$
  - Abstraction function: $C \rightarrow A$

Concretization Function

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

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

- 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 $c \subseteq C$ to the smallest (most precise) element $a \subseteq A$ such that $c \models a$

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

Alpha Examples

Concrete lattice:

Abstract lattice:

Abstraction Function Examples

- Signs abstraction
  - $a_2(c) \rightarrow \perp$ if $c = \{\}$
  - $a_2(c) \rightarrow 0$ if $c = \{0\}$
  - $a_2(c) \rightarrow +$ if for every $n \subseteq c$, $n > 0$
  - $a_2(c) \rightarrow -$ if for every $n \subseteq c$, $n < 0$
  - $a_2(c) \rightarrow T$ otherwise

- Constants abstraction
  - $a_C(c) \rightarrow \perp$ if $c = \{\}$
  - $a_C(c) \rightarrow n$ if $c = \{n\}$
  - $a_C(c) \rightarrow T$ otherwise

Outline

- Abstract interpretation
- Semantics
  - Notion of abstraction
  - Concretization and abstraction functions
  - Galois Connections
- Applications of abstract interpretation