Iterative Program Analysis
Abstract Interpretation

Mooly Sagiv
Tel Aviv University
640-6706

Textbook: Principles of Program Analysis
Chapter 4
CC79, CC92
Specialized Chaotic Iterations
System of Equations

\[ S = \begin{cases} 
    df_{\text{entry}}[s] = t \\
    df_{\text{entry}}[v] = \bigsqcup \{ f(u, v) (df_{\text{entry}}[u]) \mid (u, v) \in E \} 
\end{cases} \]

\[ F_S : L^n \to L^n \]

\[ F_S(X)[s] = t \]

\[ F_S(X)[v] = \bigsqcup \{ f(u, v)(X[u]) \mid (u, v) \in E \} \]

\[ \text{lfp}(S) = \text{lfp}(F_S) \]
Specialized Chaotic Iterations

Chaotic(G(V, E): Graph, s: Node, L: Lattice, \( \iota: L \rightarrow (L \rightarrow L) \)) {

    for each \( v \) in \( V \) to \( n \) do  \( \text{df}_{\text{entry}}[v] := \bot \)
    \( \text{df}[s] = \iota \)
    \( WL = \{ s \} \)

    while (WL \neq \emptyset) do

        select and remove an element \( u \in WL \)

        for each \( v \), such that. \( (u, v) \in E \) do

            \( \text{temp} = f(e)(\text{df}_{\text{entry}}[u]) \)

            \( \text{new} := \text{df}_{\text{entry}}(v) \sqcup \text{temp} \)

            if (\( \text{new} \neq \text{df}_{\text{entry}}[v] \)) then

                \( \text{df}_{\text{entry}}[v] := \text{new}; \)

                \( WL := WL \cup \{ v \} \)
\( z = 3 \)

\( x = 1 \)

\( y = 7 \)

\( y = z + 4 \)

\( x = 3 \)

\( \text{print } y \)

\[
\begin{array}{|c|c|}
\hline
\text{WL} & \text{df}_{\text{entry}[v]} \\
\hline
\{1\} & \text{ } \\
\hline
\{2\} & \text{df}[2]:=[x\mapsto0, y\mapsto0, z\mapsto3] \\
\hline
\{3\} & \text{df}[3]:=[x\mapsto1, y\mapsto0, z\mapsto3] \\
\hline
\{4\} & \text{df}[4]:=[x\mapsto1, y\mapsto0, z\mapsto3] \\
\hline
\{5\} & \text{df}[5]:=[x\mapsto1, y\mapsto0, z\mapsto3] \\
\hline
\{7\} & \text{df}[7]:=[x\mapsto1, y\mapsto7, z\mapsto3] \\
\hline
\{8\} & \text{df}[8]:=[x\mapsto3, y\mapsto7, z\mapsto3] \\
\hline
\{3\} & \text{df}[3]:=[x\mapsto\top, y\mapsto\top, z\mapsto3] \\
\hline
\{4\} & \text{df}[4]:=[x\mapsto\top, y\mapsto\top, z\mapsto3] \\
\hline
\{5,6\} & \text{df}[5]:=[x\mapsto1, y\mapsto\top, z\mapsto3] \\
\hline
\{6,7\} & \text{df}[6]:=[x\mapsto\top, y\mapsto\top, z\mapsto3] \\
\hline
\{7\} & \text{df}[7]:=[x\mapsto\top, y\mapsto7, z\mapsto3] \\
\hline
\end{array}
\]
The Abstract Interpretation Technique (Cousot & Cousot)

- The foundation of program analysis
- Defines the meaning of the information computed by static tools
- A mathematical framework
- Allows proving that an analysis is sound in a local way
- Identify design bugs
- Understand where precision is lost
- New analysis from old
- Not limited to certain programming style
Abstract (Conservative) interpretation

- Set of states
  - Abstract representation
  - Abstract semantics

- Set of states
  - Abstract representation
  - Abstract semantics

Operational semantics

statement $s$
Abstract (Conservative) interpretation

Set of states \( \subseteq \) Set of states

abstract representation \( s \) \( s \)

concretization

abstract semantics

Operational semantics
Abstract Interpretation

Concrete
Sets of stores

Abstract
Descriptors of sets of stores

$\alpha$

$\gamma$
Galois Connections

- Lattices $C$ and $A$ and functions $\alpha: C \rightarrow A$ and $\gamma: A \rightarrow C$

- The pair of functions $(\alpha, \gamma)$ form a Galois connection if
  - $\alpha$ and $\gamma$ are monotone
  - $\forall a \in A$ 
    $\Rightarrow \alpha(\gamma(a)) \subseteq a$
  - $\forall c \in C$
    $\Rightarrow c \subseteq \gamma(\alpha(C))$

- Alternatively if:
  $\forall c \in C$
  $\forall a \in A$
  $\alpha(c) \subseteq a$ iff $c \subseteq \gamma(a)$

- $\alpha$ and $\gamma$ uniquely determine each other
The Abstraction Function (CP)

- Map collecting states into constants
- The abstraction of an individual state
  \[ \beta_{CP} : [\text{Var}_* \rightarrow \mathbb{Z}] \rightarrow [\text{Var}_* \rightarrow \mathbb{Z} \cup \{ \bot, \top \}] \]
  \[ \beta_{CP}(\sigma) = \sigma \]
- The abstraction of set of states
  \[ \alpha_{CP} : \mathcal{P}([\text{Var}_* \rightarrow \mathbb{Z}]) \rightarrow [\text{Var}_* \rightarrow \mathbb{Z} \cup \{ \bot, \top \}] \]
  \[ \alpha_{CP}(CS) = \bigcup \{ \beta_{CP}(\sigma) | \sigma \in CS \} = \bigcup \{ \sigma | \sigma \in CS \} \]
- Soundness
  \[ \alpha_{CP}(\text{Reach}(v)) \subseteq \text{df}(v) \]
- Completeness
The Concretization Function

- Map constants into collecting states
- The formal meaning of constants
- The concretization

\[ \gamma_{CP}: [Var_\ast \rightarrow Z \cup \{\bot, \top\}] \rightarrow P([Var_\ast \rightarrow Z]) \]

\[ \gamma_{CP}(df) = \{\sigma | \beta_{CP}(\sigma) \subseteq df\} = \{\sigma | \sigma \subseteq df\} \]

- Soundness
  Reach (v) \subseteq \gamma_{CP}(df(v))

- Completeness
Galois Connection Constant
Propagation

◆ $\alpha_{CP}$ is monotone
◆ $\gamma_{CP}$ is monotone
◆ $\forall \; df \in [\text{Var}_* \rightarrow \mathbb{Z} \cup \{\bot, \top\}]$
  - $\alpha_{CP}(\gamma_{CP}(df)) \subseteq df$
◆ $\forall \; c \in P([\text{Var}_* \rightarrow \mathbb{Z}])$
  - $c_{CP} \equiv \gamma_{CP}(\alpha_{CP}(C))$
Upper Closures

- Define abstractions on sets of concrete states
- \( \uparrow: P(\Sigma) \rightarrow P(\Sigma) \) such that
  - \( \uparrow \) is monotone, i.e., \( X \subseteq Y \rightarrow \uparrow X \subseteq \uparrow Y \)
  - \( \uparrow \) is extensive, i.e., \( \uparrow X \supseteq X \)
  - \( \uparrow \) is closure, i.e., \( \uparrow(\uparrow X) = \uparrow X \)
- Every Galois connection defines an upper closure
Proof of Soundness

- Define an “appropriate” operational semantics
- Define “collecting” operational semantics by pointwise extension
- Establish a Galois connection between collecting states and abstract states
- (Local correctness) Show that the abstract interpretation of every atomic statement is sound w.r.t. the collecting semantics
- (Global correctness) Conclude that the analysis is sound
Collecting Semantics

- The input state is not known at compile-time
- "Collect" all the states for all possible inputs to the program
- No lost of precision
A Simple Example Program

\[
\begin{align*}
&z = 3 \quad \{[x \mapsto 0, y \mapsto 0, z \mapsto 3]\} \\
x = 1 \quad \{[x \mapsto 1, y \mapsto 0, z \mapsto 3]\} \\
&\text{while (x > 0)} \quad \{[x \mapsto 1, y \mapsto 0, z \mapsto 3], [x \mapsto 3, y \mapsto 0, z \mapsto 3]\}, \\
&\quad \text{if (x = 1) then } y = 7 \quad \{[x \mapsto 1, y \mapsto 7, z \mapsto 3], [x \mapsto 3, y \mapsto 7, z \mapsto 3]\} \\
&\quad \quad \text{else } y = z + 4 \\
x = 3 \quad \{[x \mapsto 1, y \mapsto 7, z \mapsto 3], [x \mapsto 3, y \mapsto 7, z \mapsto 3]\} \\
\text{print } y \quad \{[x \mapsto 3, y \mapsto 7, z \mapsto 3]\} \\
) \quad \{[x \mapsto 3, y \mapsto 7, z \mapsto 3]\}
\end{align*}
\]
Another Example

x = 0

while (true) do

  x = x + 1
An “Iterative” Definition

- Generate a system of monotone equations
- The least solution is well-defined
- The least solution is the collecting interpretation
- But may not be computable
Equations Generated for Collecting Interpretation

◆ Equations for elementary statements
  - [skip]
    \[ CS_{exit}(1) = CS_{entry}(l) \]
  - [b]
    \[ CS_{exit}(1) = \{ \sigma : \sigma \in CS_{entry}(l), \llbracket b \rrbracket \sigma = tt \} \]
  - [x := a]
    \[ CS_{exit}(1) = \{ (s[x \mapsto A[a]s]) \mid s \in CS_{entry}(l) \} \]

◆ Equations for control flow constructs
  \[ CS_{entry}(l) = \bigcup CS_{exit}(l') \]
  \(l'\) immediately precedes \(l\) in the control flow graph

◆ An equation for the entry
  \[ CS_{entry}(1) = \{ \sigma \mid \sigma \in Var_* \rightarrow Z \} \]
Specialized Chaotic Iterations
System of Equations
(Collecting Semantics)

\[ S = \left\{ \begin{array}{l}
CS_{\text{entry}}[s] = \{ \sigma_0 \} \\
CS_{\text{entry}}[v] = \bigcup \{ f(e)(CS_{\text{entry}}[u]) \mid (u, v) \in E \} \\
\text{where } f(e) = \lambda X. \{ [\text{st(e)}] \sigma \mid \sigma \in X \} \text{ for atomic statements}
\end{array} \right. \]

\[ F_S: L^n \to L^n \]

\[ F_S(X)[v] = \bigcup \{ f(e)[u] \mid (u, v) \in E \} \]

\[ \text{lfp}(S) = \text{lfp}(F_S) \]
The Least Solution

- 2n sets of equations
  \[ \text{CS}_{\text{entry}}(1), \ldots, \text{CS}_{\text{entry}}(n), \text{CS}_{\text{exit}}(1), \ldots, \text{CS}_{\text{exit}}(n) \]
- Can be written in vectorial form
  \[ \vec{\text{CS}} = F_{cs}(\vec{\text{CS}}) \]
- The least solution \( \text{lfp}(F_{cs}) \) is well-defined
- Every component is minimal
- Since \( F_{cs} \) is monotone such a solution always exists
- \( \text{CS}_{\text{entry}}(v) = \{ s | \exists s_0 \text{ s.t. } <P, s_0> \Rightarrow^* (S', s), \text{init}(S') = v \} \)
- Simplify the soundness criteria
∀a: f(γ(a)) ⊆ γ(f#(a))
Finite Height Case

\[ \text{Lfp}(f^\#) \]

\[ f^\# \]

\[ f \]

\[ \gamma \]

\[ \text{Lfp}(f) \]
Soundness Theorem (1)

1. Let \((\alpha, \gamma)\) form Galois connection from \(C\) to \(A\)
2. \(f: C \to C\) be a monotone function
3. \(f^\#: A \to A\) be a monotone function
4. \(\forall a \in A: f(\gamma(a)) \sqsubseteq \gamma(f^\#(a))\)

\[
\text{lfp}(f) \sqsubseteq \gamma(\text{lfp}(f^\#))
\]

\[
\alpha(\text{lfp}(f)) \sqsubseteq \text{lfp}(f^\#)
\]
Soundness Theorem (2)

1. Let \((\alpha, \gamma)\) form Galois connection from \(C\) to \(A\)
2. \(f: C \rightarrow C\) be a monotone function
3. \(f^\#: A \rightarrow A\) be a monotone function
4. \(\forall c \in C: \alpha(f(c)) \subseteq f^\#(\alpha(c))\)

\[\alpha(lfp(f)) \subseteq lfp(f^\#)\]
\[lfp(f) \subseteq \gamma(lfp(f^\#))\]
Soundness Theorem (3)

1. Let \((\alpha, \gamma)\) form Galois connection from \(C\) to \(A\)
2. \(f: C \to C\) be a monotone function
3. \(f^\# : A \to A\) be a monotone function
4. \(\forall a \in A: \alpha(f(\gamma(a))) \subseteq f^\#(a)\)

\[
\alpha(\text{lfp}(f)) \subseteq \text{lfp}(f^\#)
\]

\[
\text{lfp}(f) \subseteq \gamma(\text{lfp}(f^\#))
\]
Proof of Soundness (Summary)

- Define an “appropriate” structural operational semantics
- Define “collecting” structural operational semantics
- Establish a Galois connection between collecting states and reaching definitions
- (Local correctness) Show that the abstract interpretation of every atomic statement is sound w.r.t. the collecting semantics
- (Global correctness) Conclude that the analysis is sound
Completeness

\[ \alpha(\operatorname{lfp}(f)) = \operatorname{lfp}(f^\#) \]

\[ \operatorname{lfp}(f) = \gamma(\operatorname{lfp}(f^\#)) \]
Constant Propagation

- $\beta: [\text{Var} \rightarrow \mathbb{Z}] \rightarrow [\text{Var} \rightarrow \mathbb{Z} \cup \{\top, \bot\}]$
  - $\beta(\sigma) = (\sigma)$

- $\alpha: \mathcal{P}([\text{Var} \rightarrow \mathbb{Z}]) \rightarrow [\text{Var} \rightarrow \mathbb{Z} \cup \{\top, \bot\}]$
  - $\alpha(X) = \bigcup \{\beta(\sigma) \mid \sigma \in X\} = \bigcup \{\sigma \mid \sigma \in X\}$

- $\gamma: [\text{Var} \rightarrow \mathbb{Z} \cup \{\top, \bot\}] \rightarrow \mathcal{P}([\text{Var} \rightarrow \mathbb{Z}])$
  - $\gamma(\sigma^\#) = \{\sigma \mid \beta(\sigma) \subseteq \sigma^\#\} = \{\sigma \mid \sigma \subseteq \sigma^\#\}$

- **Local Soundness**
  - $\llbracket \text{st} \rrbracket^\#(\sigma^\#) \supseteq \alpha(\{\llbracket \text{st} \rrbracket \sigma \mid \sigma \in \gamma(\sigma^\#)\}) = \bigcup \{\llbracket \text{st} \rrbracket \sigma \mid \sigma \subseteq \sigma^\#\}$

- **Optimality (Induced)**
  - $\llbracket \text{st} \rrbracket^\#(\sigma^\#) = \alpha(\{\llbracket \text{st} \rrbracket \sigma \mid \sigma \in \gamma(\sigma^\#)\}) = \bigcup \{\llbracket \text{st} \rrbracket \sigma \mid \sigma \subseteq \sigma^\#\}$

- **Soundness**

- **Completeness**
Proof of Soundness (Summary)

- Define an “appropriate” structural operational semantics
- Define “collecting” structural operational semantics
- Establish a Galois connection between collecting states and reaching definitions
- (Local correctness) Show that the abstract interpretation of every atomic statement is sound w.r.t. the collecting semantics
- (Global correctness) Conclude that the analysis is sound
Best (Conservative) interpretation

Set of states

abstract representation

statement $s$

Operational semantics

concretization

abstract semantics

Abstract representation

Set of states $\subseteq$

Set of states

abstraction

concretization
Induced Analysis (Relatively Optimal)

- It is sometimes possible to show that a given analysis is not only sound but optimal w.r.t. the chosen abstraction
  - but not necessarily optimal!
- Define
  
  \[\llbracket S \rrbracket^#(df) = \alpha(\{\llbracket S \rrbracket\sigma | \sigma \in \gamma(df)\})\]
- But this \(\llbracket S \rrbracket^#\) may not be computable
- Derive (at compiler-generation time) an alternative form for \(\llbracket S \rrbracket^#\)
- A useful measure to decide if the abstraction must lead to overly imprecise results
Example Dataflow Problem

- Formal available expression analysis
- Find out which expressions are available at a given program point
- Example program

```plaintext
x = y + t  
z = y + r  
while (...) {
    t = t + (y + r)
}
```

- Lattice
- Galois connection
- Basic statements
- Soundness
Example: May-Be-Garbage

- A variable $x$ may-be-garbage at a program point $v$ if there exists an execution path leading to $v$ in which $x$’s value is unpredictable:
  - Was not assigned
  - Was assigned using an unpredictable expression

- Lattice
- Galois connection
- Basic statements
- Soundness
Points-To Analysis

- Determine if a pointer variable p may point to q on some path leading to a program point
- “Adapt” other optimizations
  - Constant propagation
    \[
    \begin{align*}
    x &= 5; \\
    *p &= 7; \\
    \ldots x \ldots
    \end{align*}
    \]
- Pointer aliases
  - Variables p and q are may-aliases at v if the points-to set at v contains entries (p, x) and (q, x)
- Side-effect analysis
  \[
  *p = *q + * * t
  \]
The **PWhile** Programming Language

Abstract Syntax

\[ a := x | *x | \&x | n | a_1 \text{ op } a_2 \]

\[ b := \text{true} | \text{false} | \text{not } b | b_1 \text{ op } b_2 | a_1 \text{ op } r \ a_2 \]

\[ S := x := a | *x := a | \text{skip} | S_1 ; S_2 | \text{if } b \text{ then } S_1 \text{ else } S_2 | \text{while } b \text{ do } S \]
Concrete Semantics 1 for PWhile

State1 = [Loc → Loc ∪ Z]

For every atomic statement S

\[ [S]: States1 \rightarrow States1 \]

\[ [x := a](\sigma) = \sigma[ loc(x) \rightarrow A[a] \sigma] \]

\[ [x := &y](\sigma) \]

\[ [x := *y](\sigma) \]

\[ [x := y](\sigma) \]

\[ *[x := y](\sigma) \]
Points-To Analysis

- Lattice $L_{pt} =$
- Galois connection
Abstract Semantics for PWhile

• For every atomic statement $S$

\[
\left[ S \right] \# : P(\text{Var}^* \times \text{Var}^*) \rightarrow P(\text{Var}^* \times \text{Var}^*)
\]

\[
\left[ x := &y \right] \#
\]

\[
\left[ x := *y \right] \#
\]

\[
\left[ x := y \right] \#
\]

\[
\left[ *x := y \right] \#
\]
t := &a;
y := &b;
z := &c;

if x > 0;
    then p := &y;
    else p := &z;

*p := t;
/* Ø */ t := &a; /* {(t, a)} */
/* {(t, a)} */ y := &b; /* {(t, a), (y, b)} */
/* {(t, a), (y, b)} */ z := &c; /* {(t, a), (y, b), (z, c)} */
if x > 0;
    then p := &y; /* {(t, a), (y, b), (z, c), (p, y)} */
    else p := &z; /* {(t, a), (y, b), (z, c), (p, z)} */
/* {(t, a), (y, b), (z, c), (p, y), (p, z)} */
*p := t;
/* {(t, a), (y, b), (y, c), (p, y), (p, z), (y, a), (z, a)} */
Flow insensitive points-to-analysis
Steengard 1996

- Ignore control flow
- One set of points-to per program
- Can be represented as a directed graph
- Conservative approximation
  - Accumulate pointers
- Can be computed in almost linear time
t := &a;
y := &b;

z := &c;

if x > 0;
    then p := &y;
    else p := &z;

*p := t;
Precision

◆ We cannot usually have
  – $\alpha(CS) = DF$
    on all programs

◆ But can we say something about precision in all programs?
Summary

- Abstract interpretation Connects Abstract and Concrete Semantics
- Galois Connection
- Local Correctness
- Global Correctness