Chaotic Iterations, Termination, Optimality, Correctness, JOP vs. LFP

cs264
Ras Bodik

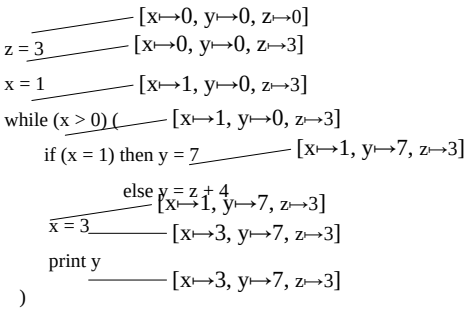
spring 2005

slides by Mooly Sagiv

Outline

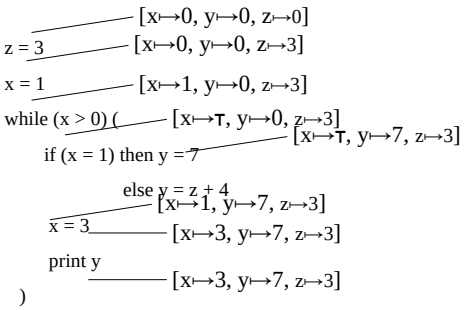
- familiar constant propagation
- mathematical background: posets and lattices
- chaotic iterations
- termination
- precision

A Simple Example Program



A Simple Example Program

A Simple Example Program



A Simple Example Program

A Simple Example Program

$[x\mapsto 0, y\mapsto 0, z\mapsto 0]$

z = 3 $[x\mapsto 0, y\mapsto 0, z\mapsto 3]$

x = 1 $[x\mapsto 1, y\mapsto 0, z\mapsto 3]$

while (x > 0) ($[x\mapsto \top, y\mapsto 0, z\mapsto 3]$

if (x = 1) then y = 7 $[x\mapsto 1, y\mapsto 7, z\mapsto 3]$

else y =z + 4 $[x\mapsto \top, y\mapsto 7, z\mapsto 3]$ $[x\mapsto \top, y\mapsto 7, z\mapsto 3]$

x = 3 $[x\mapsto 1, y\mapsto 7, z\mapsto 3]$

$[x\mapsto 3, y\mapsto 7, z\mapsto 3]$

print y

$[x\mapsto 3, y\mapsto 7, z\mapsto 3]$

)

A Simple Example Program

$[x\mapsto 0, y\mapsto 0, z\mapsto 0]$

z = 3 $[x\mapsto 0, y\mapsto 0, z\mapsto 3]$

x = 1 $[x\mapsto 1, y\mapsto 0, z\mapsto 3]$

while (x > 0) ($[x\mapsto \top, y\mapsto 0, z\mapsto 3]$

if (x = 1) then y = 7 $[x\mapsto 1, y\mapsto 7, z\mapsto 3]$

else y =z + 4 $[x\mapsto \top, y\mapsto 7, z\mapsto 3]$ $[x\mapsto \top, y\mapsto 7, z\mapsto 3]$

x = 3 $[x\mapsto \top, y\mapsto 7, z\mapsto 3]$

$[x\mapsto 3, y\mapsto 7, z\mapsto 3]$

print y

$[x\mapsto 3, y\mapsto 7, z\mapsto 3]$

)

Computing Constants

- Construct a control flow graph (CFG)
- Associate transfer functions with control flow graph edges
- Iterate until a solution is found
- The solution is unique
  - But order of evaluation may affect the number of iterations

Constructing CFG

z = 3

x = 1

while (x > 0) (

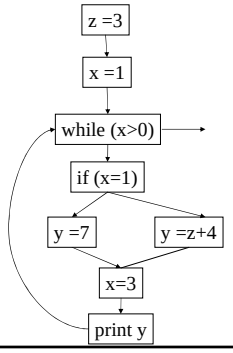
if (x = 1) then y = 7

else y =z + 4

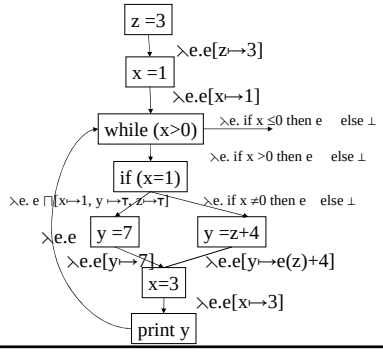
x = 3

print y

)

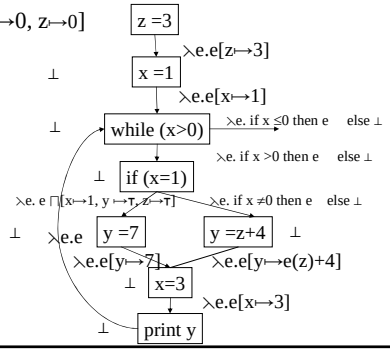


Associating Transfer Functions

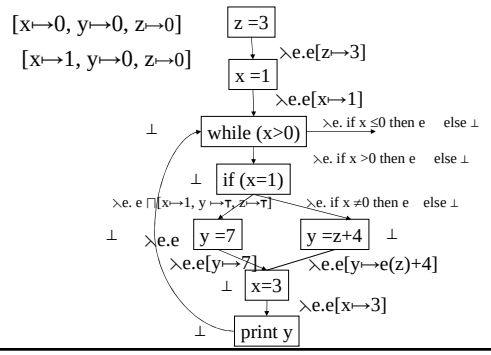


Iterative Computation

$[x\mapsto 0, y\mapsto 0, z\mapsto 0]$

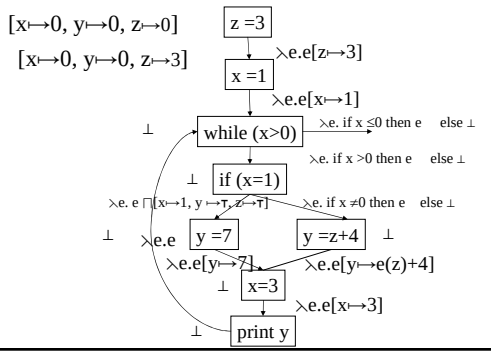


Iterative Computation



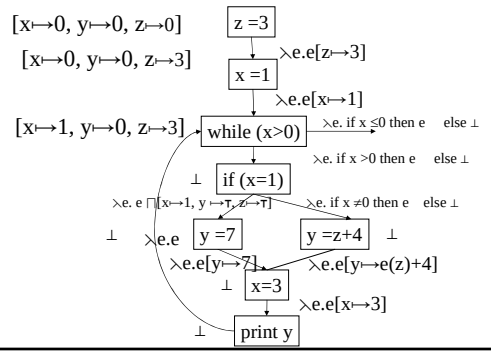
13

Iterative Computation



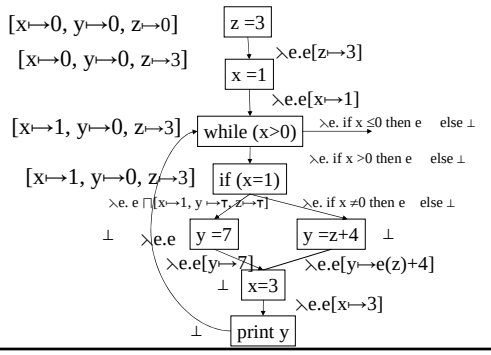
14

Iterative Computation



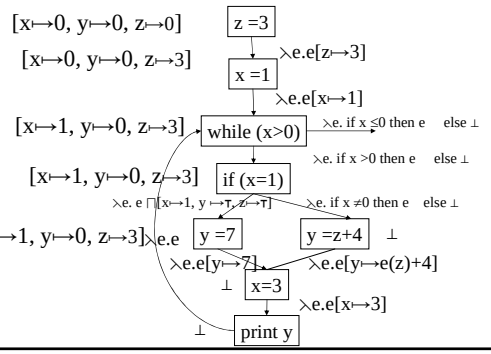
15

Iterative Computation



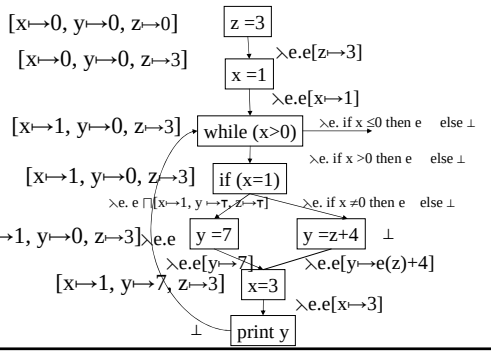
16

Iterative Computation



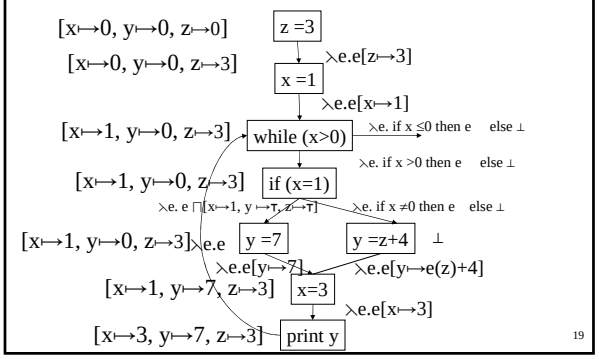
17

Iterative Computation



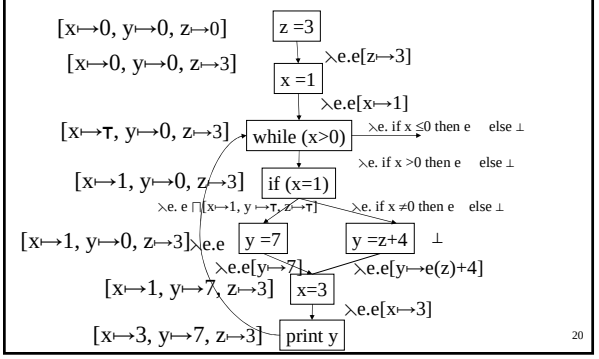
18

Iterative Computation



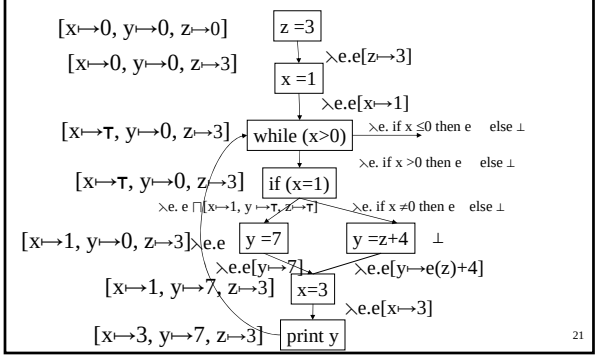
19

Iterative Computation



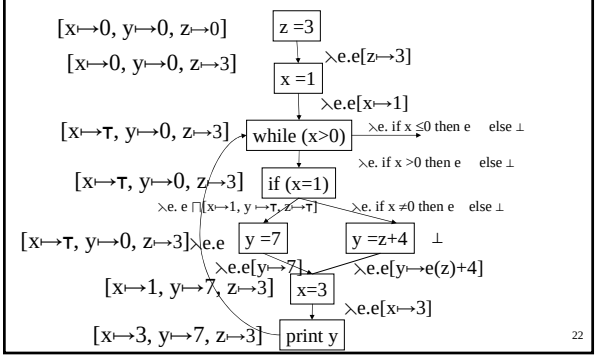
20

Iterative Computation



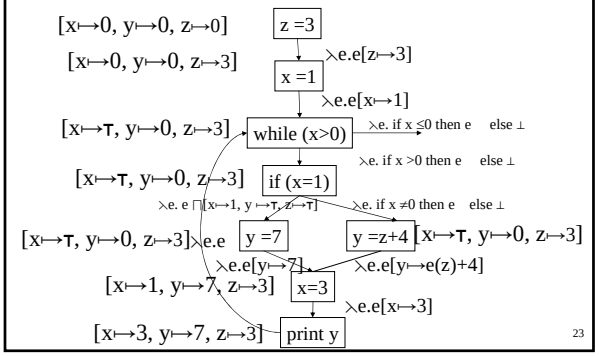
21

Iterative Computation



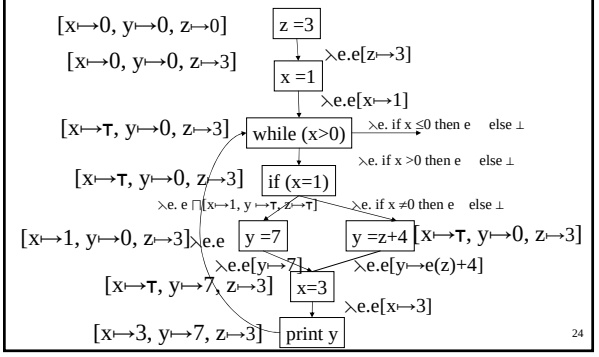
22

Iterative Computation



23

Iterative Computation



24

Mathematical Background

- Declaratively define
  - The result of the analysis
  - The exact solution
  - Allow comparison

25

Posets

- A partial ordering is a binary relation
  $\sqsubseteq : L \times L \rightarrow \{false, true\}$
  - For all $l \in L : l \sqsubseteq l$ (Reflexive)
  - For all $l_1, l_2, l_3 \in L : l_1 \sqsubseteq l_2, l_2 \sqsubseteq l_3 \Rightarrow l_1 \sqsubseteq l_3$ (Transitive)
  - For all $l_1, l_2 \in L : l_1 \sqsubseteq l_2, l_2 \sqsubseteq l_1 \Rightarrow l_1 = l_2$ (Anti-Symmetric)

26

Posets

- Denoted by $(L, \sqsubseteq)$
- In program analysis:
  $l_1 \sqsubseteq l_2 \Leftrightarrow$
  l_1 is more precise than $l_2 \Leftrightarrow$
  l_1 represents fewer concrete states than l_2
- Examples
  - Total orders $(N, \leq)$
  - Powersets $(P(S), \subseteq)$
  - Powersets $(P(S), \supseteq)$
  - Constant propagation

27

Upper and Lower Bounds

- Consider a poset $(L, \sqsubseteq)$
- A subset $L' \subseteq L$ has a lower bound $l \in L$ if for all $l' \in L' : l \sqsubseteq l'$
- A subset $L' \subseteq L$ has an upper bound $u \in L$ if for all $l' \in L' : l' \sqsubseteq u$

28

Bounds

- A greatest lower bound of a subset $L' \subseteq L$ is a lower bound $l_0 \in L$ such that $l \sqsubseteq l_0$ for any lower bound l of L'
- A lowest upper bound of a subset $L' \subseteq L$ is an upper bound $u_0 \in L$ such that $u_0 \sqsubseteq u$ for any upper bound u of L'
- For every subset $L' \subseteq L$:
  - The greatest lower bound of L' is unique if at all exists
    - $\sqcap L'$ (meet) $a \sqcap b$
  - The lowest upper bound of L' is unique if at all exists
    - $\sqcup L'$ (join) $a \sqcup b$

29

Complete Lattices

- A poset $(L, \sqsubseteq)$ is a complete lattice if every subset has least and upper bounds
- $L = (L, \sqsubseteq) = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
  - $\bot = \sqcup \emptyset = \sqcap L$
  - $\top = \sqcup L = \sqcap \emptyset$
- Examples
  - Total orders $(N, \leq)$
  - Powersets $(P(S), \subseteq)$
  - Powersets $(P(S), \supseteq)$
  - Constant propagation

30

Complete Lattices

- Lemma For every poset $(L, \sqsubseteq)$ the following conditions are equivalent
  - L is a complete lattice
  - Every subset of L has a least upper bound
  - Every subset of L has a greatest lower bound

Cartesian Products

- A complete lattice $(L_1, \sqsubseteq_1) = (L_1, \sqsubseteq, \sqcup_1, \sqcap_1, \bot_1, \top_1)$
- A complete lattice $(L_2, \sqsubseteq_2) = (, \sqsubseteq, \sqcup_2, \sqcap_2, \bot_2, \top_2)$
- Define a Poset $L = (L_1 \times L_2, \sqsubseteq)$ where
  - $(x_1, x_2) \sqsubseteq (y_1, y_2)$ if
    - $x_1 \sqsubseteq y_1$ and
    - $x_2 \sqsubseteq y_2$
- L is a complete lattice

Finite Maps

- A complete lattice $(L_1, \sqsubseteq_1) = (L_1, \sqsubseteq, \sqcup_1, \sqcap_1, \bot_1, \top_1)$
- A finite set V
- Define a Poset $L = (V \to L_1, \sqsubseteq)$ where
  - $e_1 \sqsubseteq e_2$ if for all $v \in V$
    - $e_1 v \sqsubseteq e_2 v$
- L is a complete lattice

Chains

- A subset $Y \subseteq L$ in a poset $(L, \sqsubseteq)$ is a chain if every two elements in Y are ordered
  - For all $l_1, l_2 \in Y$: $l_1 \sqsubseteq l_2$ or $l_2 \sqsubseteq l_1$
- An ascending chain is a sequence of values
  - $l_1 \sqsubseteq l_2 \sqsubseteq l_3 \sqsubseteq \ldots$
- A strictly ascending chain is a sequence of values
  - $l_1 \sqsubset l_2 \sqsubset l_3 \sqsubset \ldots$
- A descending chain is a sequence of values
  - $l_1 \sqsupseteq l_2 \sqsupseteq l_3 \sqsupseteq \ldots$
- A strictly descending chain is a sequence of values
  - $l_1 \sqsupset l_2 \sqsupset l_3 \sqsupset \ldots$
- L has a finite height if every chain in L is finite
- Lemma A poset $(L, \sqsubseteq)$ has finite height if and only if every strictly decreasing and strictly increasing chains are finite

Monotone Functions

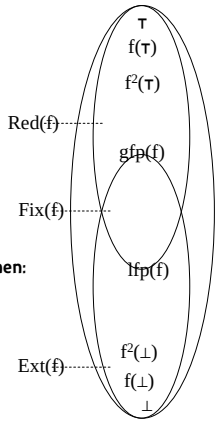
- A poset $(L, \sqsubseteq)$
- A function $f: L \to L$ is monotone if for every $l_1, l_2 \in L$:
  - $l_1 \sqsubseteq l_2 \Rightarrow f(l_1) \sqsubseteq f(l_2)$

Galois Connection

- Lattices C and A and functions $\alpha: C \to A$ and $\gamma: A \to C$
- The pair of functions (α, γ) form Galois connection if
  - α and γ are monotone
  - $\forall a \in A: \alpha(\gamma(a)) \sqsubseteq a$
  - $\forall c \in C: c \sqsubseteq \gamma(\alpha(C))$
- Alternatively if:
  $\forall c \in C . \forall a \in A$
  $\alpha(c) \sqsubseteq a$ iff $c \sqsubseteq \gamma(a)$
- α and γ uniquely determine each other

Fixed Points

- A monotone function f: L → L where (L, ⊑, ⊔, ⊓, ⊥, ⊤) is a complete lattice
- Fix(f) = { l: l ∈ L, f(l) = l}
- Red(f) = {l: l ∈ L, f(l) ⊑ l}
- Ext(f) = {l: l ∈ L, l ⊑ f(l)}
  - $l_1 \sqsubseteq l_2 \Rightarrow f(l_1) \sqsubseteq f(l_2)$
- Tarski's Theorem 1955: if f is monotone then:
  - lfp(f) = ⊓ Fix(f) = ⊓ Red(f) ∈ Fix(f)
  - gfp(f) = ⊔ Fix(f) = ⊔ Ext(f) ∈ Fix(f)



Chaotic Iterations

- A lattice L = (L, ⊑, ⊔, ⊓, ⊥, ⊤) with finite strictly increasing chains
- $L^n = L \times L \times \ldots \times L$
- A monotone function $\underline{f}: L^n \rightarrow L^n$
- Compute lfp($\underline{f}$)
- The simultaneous least fixed of the system $\{x[i] = \underline{f}_i(x) : 1 \leq i \leq n\}$

$\underline{x} := (\bot, \bot, \ldots, \bot)$

while ($\underline{f}(x) \neq \underline{x}$) do

$\underline{x} := \underline{f}(\underline{x})$

Chaotic Iterations

```
for i :=1  to n do
    x[i] = ⊥
WL = {1, 2, …, n}
while (WL ≠  Ø ) do
   select and remove an element i ∈ WL
   new := f_i(x)
    if (new ≠ x[i]) then
        x[i] := new;
        Add all the indexes that directly depends on i to WL
```

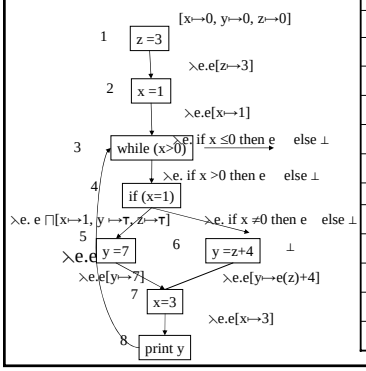

Precision

Least fixed point gives us

- Minimum number of non-constant
- Maximum number of ⊥

Specialized Chaotic Iterations

```
Chaotic(G(V, E): Graph, s: Node, L: Lattice, ι: L, f: E →(L →L) ){
   for each v in V to n do df_entry[v] := ⊥
   df[v] = ι
   WL = {s}
   while (WL ≠  Ø ) do
      select and remove an element u ∈ WL
      for each v, such that. (u, v) ∈E do
            temp = f(e)(df_entry[u])
            new := df_entry(v)⊔ temp
            if (new ≠ df_entry[v]) then
                  df_entry[v] := new;
                  WL := WL ∪{v}
```

WL	$df_{entry}[v]$
{1}	
{2}	df[2]:=[x↦0, y↦0, z↦3]
{3}	df[3]:=[x↦1, y↦0, z↦3]
{4}	df[4]:=[x↦1, y↦0, z↦3]
{5}	df[5]:=[x↦1, y↦0, z↦3]
{7}	df[7]:=[x↦1, y↦7, z↦3]
{8}	df[8]:=[x↦3, y↦7, z↦3]
{3}	df[3]:=[x↦⊤, y↦⊤, z↦3]
{4}	df[4]:=[x↦⊤, y↦⊤, z↦3]
{5,6}	df[5]:=[x↦1, y↦⊤, z↦3]
{6,7}	df[6]:=[x↦⊤, y↦⊤, z↦3]
{7}	df[7]:=[x↦⊤, y↦7, z↦3]

Specialized Chaotic Iterations: System of Equations

$$S = \left\{ \begin{array}{l} df_{entry}[s] = \iota \\ df_{entry}[v] = \sqcup\{f(u, v)\,(df_{entry}[u]) \mid (u, v) \in E\} \end{array} \right\}$$

$F_S: L^n \rightarrow L^n$

$F_S(X)[s] = \iota$

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

$$lfp(S) = lfp(F_S)$$

43

Complexity of Chaotic Iterations

- Parameters:
  - n the number of CFG nodes
  - k is the maximum outdegree of edges
  - A lattice of height h
  - c is the maximum cost of
    - applying $f_{(e)}$
    - $\sqcup$
    - L comparisons
- Complexity: O(n * h * c * k)

44

Soundness, Completeness

- Soundness:
  - Every detected constant is indeed such
  - Every error will be detected
  - The least fixed point contains all occurring runtime states
- Completeness:
  - Every constant is indeed detected as such
  - Every detected error is real
  - Every state contained in the least fixed point is reachable for some input

45

The Abstract Interpretation Technique

- The foundation of program analysis
- Goals
  - Establish soundness of (find faults in) a given program analysis algorithm
  - Design new program analysis algorithms
- The main ideas:
  - Relate each step in the algorithm to a step in a structural operational semantics
  - Establish global correctness using a general theorem
- Not limited to a particular form of analysis

46

Soundness in Constant Propagation

- Every detected constant is indeed such
- May include fewer constants
- May miss $\perp$
- At every CFG node *l*
  All constants in $df_{entry}(l)$ are indeed constants
- At every CFG node l $df_{entry}(l)$ "represents" all the possible concrete states arising when the structural operational semantics reaches *l*

47

Proof of Soundness

- 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 CC1976

$$\gamma(lfp(S)[v, entry]) \supseteq REACH[v, entry]$$

48

Structural Semantics for While

axioms

$[ass_{sos}]$ $\langle x := a, s\rangle \Rightarrow s[x \mapsto \mathbf{A}[\![a]\!]s]$

$[skip_{sos}]$ $\langle \mathbf{skip}, s\rangle \Rightarrow s$

rules

$$[comp^1_{sos}]\ \frac{\langle S_1, s\rangle \Rightarrow \langle S'_1, s'\rangle}{\langle S_1; S_2, s\rangle \Rightarrow \langle S'_1; S_2, s'\rangle}$$

$$[comp^2_{sos}]\ \frac{\langle S_1, s\rangle \Rightarrow s'}{\langle S_1; S_2, s\rangle \Rightarrow \langle S_2, s'\rangle}$$

Structural Semantics for While if construct

$[if^{tt}_{sos}]$ $\langle$if b then S_1 else S_2, s$\rangle \Rightarrow \langle S_1, s\rangle$ if $\mathbf{B}[\![b]\!]s=tt$

$[if^{ff}_{os}]$ $\langle$if b then S_1 else S_2, s$\rangle \Rightarrow \langle S_2, s\rangle$ if $\mathbf{B}[\![b]\!]s=ff$

Structural Semantics for While while construct

$[while_{sos}]$ $\langle$while b do S, s$\rangle \Rightarrow$
$\langle$if b then (S; while b do S) else skip, s$\rangle$

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

$\{[x\mapsto 0, y\mapsto 0, z\mapsto 0]\}$

z = 3 $\{[x\mapsto 0, y\mapsto 0, z\mapsto 3]\}$

x = 1 $\{[x\mapsto 1, y\mapsto 0, z\mapsto 3]\}$

while (x > 0) ($\{[x\mapsto 1, y\mapsto 0, z\mapsto 3], [x\mapsto 3, y\mapsto 0, z\mapsto 3],\}$

if (x = 1) then y = 7 $\{[x\mapsto 1, y\mapsto 7, z\mapsto 3], [x\mapsto 3, y\mapsto 7, z\mapsto 3]\}$

else y = z + 4

x = 3 $\{[x\mapsto 1, y\mapsto 7, z\mapsto 3], [x\mapsto 3, y\mapsto 7, z\mapsto 3]\}$

print y $\{[x\mapsto 3, y\mapsto 7, z\mapsto 3]\}$

) $\{[x\mapsto 3, y\mapsto 7, z\mapsto 3]\}$

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

55

Equations Generated for Collecting Interpretation

- Equations for elementary statements
  - [skip]
    $CS_{exit}(1) = CS_{entry}(l)$
  - [b]
    $CS_{exit}(1) = \{\sigma: \sigma \in CS_{entry}(l), [\![b]\!]\sigma = tt\}$
  - [x := a]
    $CS_{exit}(1) = \{(s[x \mapsto \mathbf{A}[\![a]\!]s]) \mid s \in CS_{entry}(l)\}$
- Equations for control flow constructs
  $CS_{entry}(l) = \cup CS_{exit}(l')$ *l'* immediately precedes *l* in the control flow graph
- An equation for the entry
  $CS_{entry}(1) = \{\sigma \mid \sigma \in Var_* \to Z\}$

56

Specialized Chaotic Iterations System of Equations (Collecting Semantics)

S =

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

$F_S: L^n \to L^n$

$F_s(X)[v] = \cup\{f(e)[u] \mid (u, v) \in E\}$

$lfp(S) = lfp(F_S)$

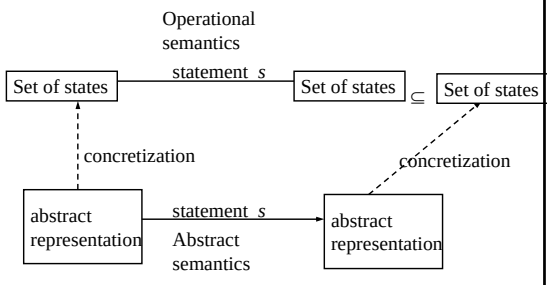
57

The Least Solution

- 2n sets of equations
  $CS_{entry}(1), \ldots, CS_{entry}(n), CS_{exit}(1), \ldots, CS_{exit}(n)$
- Can be written in vectorial form
  $$\overrightarrow{CS} = F_{cs}(\overrightarrow{CS})$$
- The least solution $lfp(F_{cs})$ is well-defined
- Every component is minimal
- Since F_{cs} is monotone such a solution always exists
- $CS_{entry}(v) = \{s \mid \exists s_0 \mid \langle P, s_0\rangle \Rightarrow^* (S', s)),$
  $init(S') = v\}$
- Simplify the soundness criteria

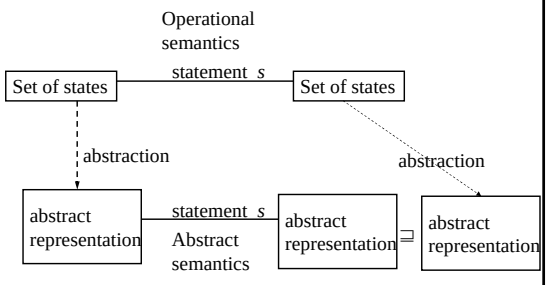
58

Abstract (Conservative) interpretation



59

Abstract (Conservative) interpretation



60

The Abstraction Function

- Map collecting states into constants
- The abstraction of an individual state
  β_{CP}:[Var$_\star \to$Z] $\to$ [Var$_\star \to$Z$\cup\{\bot, \top\}$]
  $\beta_{CP}(\sigma) = \sigma$
- The abstraction of set of states
  α_{CP}:P([Var$_\star \to$Z]) $\to$ [Var$_\star \to$Z$\cup\{\bot, \top\}$]
  α_{CP} (CS) = $\sqcup \{ \beta_{CP}(\sigma) \mid \sigma \in CS\} = \sqcup\{\sigma \mid \sigma \in CS\}$
- Soundness
  α_{CP} (CS_{entry} (v)) $\sqsubseteq$ df_{entry}(v)
- Completeness

The Concretization Function

- Map constants into collecting states
- The formal meaning of constants
- The concretization
  γ_{CP}: [Var$_\star \to$Z$\cup\{\bot, \top\}$] $\to$P([Var$_\star \to$Z])
  γ_{CP} (df) = $\{\sigma \mid \beta_{CP}(\sigma) \sqsubseteq df\} = \{\sigma \mid \sigma \sqsubseteq df\}$
- Soundness
  CS_{entry} (v) $\subseteq \gamma_{CP}$ (df_{entry}(v))
- Optimality

Galois Connection

- α_{CP} is monotone
- γ_{CP} is monotone
- $\forall$ df $\in$ [Var$_\star \to$Z$\cup\{\bot, \top\}$]
  - $\alpha_{CP}(\gamma_{CP}(df)) \sqsubseteq df$
- $\forall$ c $\in$ P([Var$_\star \to$Z])
  - $c_{CP} \sqsubseteq \gamma_{CP}(\alpha_{CP}(C))$

Local Concrete Semantics

- For every atomic statement S
  - $[\![S]\!]$: [Var$_\star \to$Z] $\to$[Var$_\star \to$Z]
  - $[\![x := a]\!]s = s[x \mapsto \mathbf{A}[\![a]\!]s]$
  - $[\![skip]\!]s = s$
- For Boolean conditions …

Local Abstract Semantics

- For every atomic statement S
  - $[\![S]\!]^\#$:Var$_\star \to$L $\to$ Var$_\star \to$L
  - $[\![x := a]\!]^\#(e) = e[x \mapsto [\![a]\!]^\#(e)]$
  - $[\![skip]\!]^\#(e) = e$
- For Booleans …

Local Soundness

- For every atomic statement S show one of the following
  - $\alpha_{CP}(\{[\![S]\!]\sigma \mid \sigma \in CS\} \sqsubseteq [\![S]\!]^\#(\alpha_{CP}(CS))$
  - $\{[\![S]\!]\sigma \mid \sigma \in \gamma_{CP}(df)\} \subseteq \gamma_{CP}([\![S]\!]^\#(df))$
  - $\alpha(\{[\![S]\!]\sigma \mid \sigma \in \gamma_{CP}(df)\}) \sqsubseteq [\![S]\!]^\#(df)$
- The above condition implies global soundness [Cousot & Cousot 1976]
  $\alpha(CS_{entry}(l)) \sqsubseteq df_{entry}(l)$
  $CS_{entry}(l) \subseteq \gamma(df_{entry}(l))$

Lemma 1

Consider a lattice L.

f: $L \to L$ is monotone iff

for all $X \subseteq L$: $\sqcup\{f(z) \mid z \in X\} \sqsubseteq f(\sqcup\{z \mid z \in X\})$

Assignments in constant propagation

- **Monotone**
  - $df_1 \sqsubseteq df_2 \to [\![x := e]\!]^{\#}(df_1) \sqsubseteq [\![x := e]\!]^{\#}(df_2)$
- Local Soundness
  - $\alpha(\{[\![x := e]\!]\sigma \mid \sigma \in \mathbf{CS}\}) \sqsubseteq [\![x := e]\!]^{*}(\alpha(\mathbf{CS}))$

Soundness Theorem(1)

1. Let (α, γ) 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. $\forall a \in A$: $f(\gamma(a)) \sqsubseteq \gamma(f^{\#}(a))$

$lfp(f) \sqsubseteq \gamma(lfp(f^{\#}))$

$\alpha(lfp(f)) \sqsubseteq lfp(f^{\#})$

Soundness Theorem(2)

1. Let (α, γ) 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. $\forall c \in C$: $\alpha(f(c)) \sqsubseteq f^{\#}(\alpha(c))$

$\alpha(lfp(f)) \sqsubseteq lfp(f^{\#})$

$lfp(f) \sqsubseteq \gamma(lfp(f^{\#}))$

Soundness Theorem(3)

1. Let (α, γ) 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. $\forall a \in A$: $\alpha(f(\gamma(a))) \sqsubseteq f^{\#}(a)$

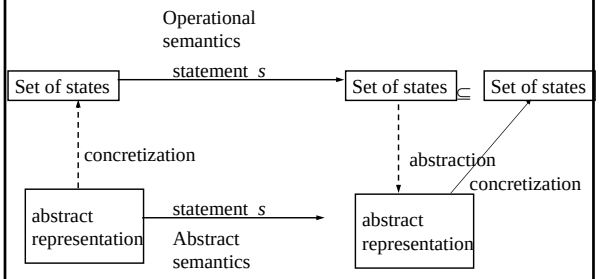
$\alpha(lfp(f)) \sqsubseteq lfp(f^{\#})$

$lfp(f) \sqsubseteq \gamma(lfp(f^{\#}))$

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**

Best (Conservative) interpretation



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 $[\![S]\!]^{\#}(df) = \alpha(\{[\![S]\!]\sigma \mid \sigma \in \gamma(df)\})$
- But this $[\![S]\!]^{\#}$ may not be computable
- Derive (at compiler-generation time) an alternative form for $[\![S]\!]^{\#}$
- 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

```
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 a 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
    x = 5; *p = 7 ; … x …
- 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

$a := x \mid *x \mid \&x \mid n \mid a_1\ op_a\ a_2$

$b :=$ true $\mid$ false $\mid$ not $b \mid\ b_1\ op_b\ b_2 \mid a_1\ op_r\ a_2$

$S := x := a \mid *x := a \mid$ skip $\mid S_1 ; S_2 \mid$
if b **then** S_1 **else** $S_2 \mid$ **while** b do S

Concrete Semantics 1 for PWhile

State1= [Loc→Loc∪Z]

For every atomic statement S

⟦S ⟧ : States1 →States1

⟦x := a ⟧(σ)=σ[loc(x) ↦**A**⟦a⟧ σ]

⟦x := &y ⟧(σ)

⟦x := *y ⟧(σ)

⟦x := y ⟧(σ)

⟦*x := y ⟧(σ)

79

Points-To Analysis

- **Lattice L_{pt} =**
- **Galois connection**

80

Abstract Semantics for PWhile

- For every atomic statement S

⟦S ⟧ #: P(Var*× Var*)→ P(Var*× Var*)

⟦x := &y ⟧#

⟦x := *y ⟧#

⟦x := y ⟧#

⟦*x := y ⟧#

81

Example

t := &a;
y := &b;

z := &c;

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

*p := t;

82

Example

/* ∅ */ 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)}*/

83

Flow-insensitive points-to-analysis Steengard '96

- **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**

84

Example

```
t := &a;
y := &b;
z := &c;
if x> 0;
      then p:= &y;
      else p:= &z;
*p := t;
```

85

Precision

- We cannot usually have
  - α(CS) = DF on all programs
- But can we say something about precision in all programs?

86

The Join-Over-All-Paths (JOP)

- Let paths(v) denote the potentially infinite set paths from start to v (written as sequences of labels)
- For a sequence of edges $[e_1, e_2, \ldots, e_n]$ define $f[e_1, e_2, \ldots, e_n]: L \rightarrow L$ by composing the effects of basic blocks
  $f[e_1, e_2, \ldots, e_n](l) = f(e_n)(\ldots(f(e_2)(f(e_1)(l)))\ldots)$
- $JOP[v] = \sqcup\{f[e_1, e_2, \ldots, e_n](\iota)$
  $[e_1, e_2, \ldots, e_n] \in paths(v)\}$

87

JOP vs. Least Solution

- The DF solution obtained by Chaotic iteration satisfies for every *l*:
  - $JOP[v] \sqsubseteq DF_{entry}(v)$
- A function f is additive (distributive) if
  - $f(\sqcup\{x \mid x \in X\}) = \sqcup\{f(x) \mid \in X\}$
- If every f_l is additive (distributive) for all the nodes v
  - $JOP[v] = DF_{entry}(v)$

88

Conclusions

- Chaotic iterations is a powerful technique
- Easy to implement
- Rather precise
- But expensive
  - More efficient methods exist for structured programs
- Abstract interpretation relates runtime semantics and static information
- The concrete semantics serves as a tool in designing abstractions
  - More intuition will be given in the sequel

89