Chaotic Iterations

Mooly Sagiv
http://www.cs.tau.ac.il/~msagiv/courses/pa16.html
Tel Aviv University
640-6706

Textbook: Principles of Program Analysis

F. Nielson, H. Nielson, C.L. Hankin

Appendix A

Caterina's thesis

Content

- Mathematical Background
- Chaotic Iterations
- Examples
- Soundness, Precision and more examples next week

Mathematical Background

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

Posets

- A partial ordering is a binary relation $\sqsubseteq : L \times L \to \{\text{false}, \text{true}\}$
  - 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)
- Denoted by $(L, \sqsubseteq)$

Posets

- 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 $(P(S), \supseteq)$
  - Even/Odd
  - Constant propagation
    - » Single variable
    - » Multiple variables
  - Intervals
- Bad examples
  - $(N, <)$
  - Non-transitive $x \sqsubseteq y \Leftrightarrow x = y \vee (x = 0 \wedge y=1) \vee (x=1 \wedge y=2)$
  - Non anti-symmetric

Posets

- More notations
  - $l_1 \sqsupseteq l_2 \Leftrightarrow l_2 \sqsubseteq l_1$
  - $l_1 \sqsubset l_2 \Leftrightarrow l_1 \sqsubseteq l_2 \wedge l_1 \neq l_2$
  - $l_1 \sqsupset l_2 \Leftrightarrow l_2 \sqsubset l_1$

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 $l' \in L'$: $l' \sqsubseteq u$
- 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) $a \sqcap b = \sqcap\{a, b\}$
  - The lowest upper bound of L' is unique if at all exists
    - » $\sqcup L'$ (join) $a \sqcup b = \sqcup\{a, b\}$

Complete Lattices

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

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 $(L_1, \sqsubseteq_1) = (L_1, \sqsubseteq, \sqcup_1, \sqcap_1, \perp_1, \top_1)$
- A complete lattice $(L_2, \sqsubseteq_2) = (, \sqsubseteq, \sqcup_2, \sqcap_2, \perp_2, \top_2)$
- 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 $(L_1, \sqsubseteq_1) = (L_1, \sqsubseteq, \sqcup_1, \sqcap_1, \perp_1, \top_1)$
- A finite set V
- Define a Poset $L = (V \rightarrow L_1, \sqsubseteq)$ 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

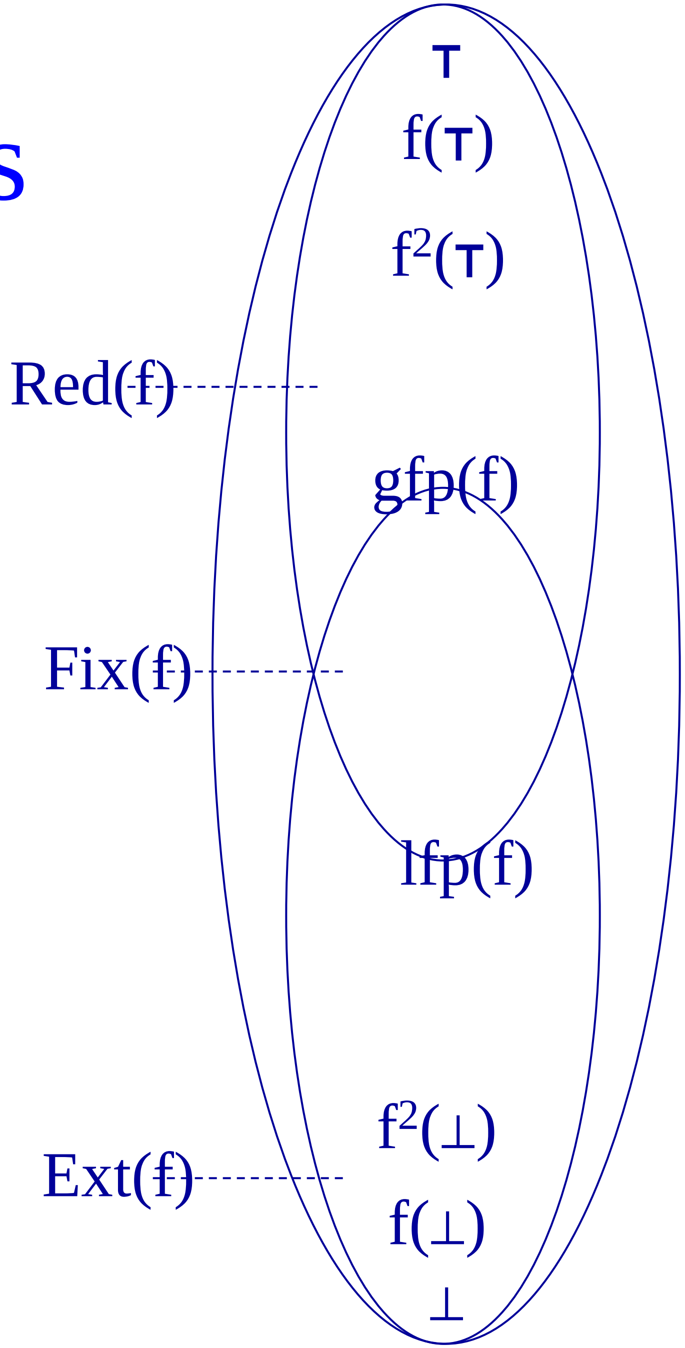
- A subset $Y \subseteq L$ in a poset $(L, \sqsubseteq)$ is a chain if every two elements in Y are ordered
  - For all $l_1, l_2 \in Y$: $l_1 \sqsubseteq l_2$ or $l_2 \sqsubseteq l_1$
- An ascending chain is a sequence of values
  - $l_1 \sqsubseteq l_2 \sqsubseteq l_3 \sqsubseteq \ldots$
- A strictly 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 $l_1, l_2 \in L$:
  - $l_1 \sqsubseteq l_2 \Rightarrow f(l_1) \sqsubseteq f(l_2)$

Fixed Points

- A monotone function f: $L \to L$ where $(L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$ is a complete lattice
- $Fix(f) = \{ l: l \in L, f(l) = l\}$
- $Red(f) = \{l: l \in L, f(l) \sqsubseteq l\}$
- $Ext(f) = \{l: l \in L, l \sqsubseteq f(l)\}$
  – $l_1 \sqsubseteq l_2 \Rightarrow f(l_1) \sqsubseteq f(l_2)$
- Tarski's Theorem 1955: if f is monotone then:
  – $lfp(f) = \sqcap Fix(f) = \sqcap Red(f) \in Fix(f)$
  – $gfp(f) = \sqcup Fix(f) = \sqcup Ext(f) \in Fix(f)$



Chaotic Iterations

- A lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \perp, \top)$ 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 $\mathrm{lfp}(\underline{f})$
- The simultaneous least fixed of the system $\{x[i] = \underline{f}_i(x) : 1 \leq i \leq n\}$

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

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

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

for i :=1 to n do
 $x[i] = \perp$

WL = {1, 2, …, n}

while (WL ≠ ∅) do

select and remove an element $i \in$ WL

new := $f_i(\underline{x})$

if (new ≠ x[i]) then

x[i] := new;

Add all the indexes that directly depends on i to WL

Specialized Chaotic Iterations System of Equations

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

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

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

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

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

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 $df_{entry}[v] := \perp$

$df[s] = \iota$

WL = {s}

while (WL ≠ Ø) do

select and remove an element u ∈ WL

for each v, such that. (u, v) ∈E do

temp = $f(e)(df_{entry}[u])$

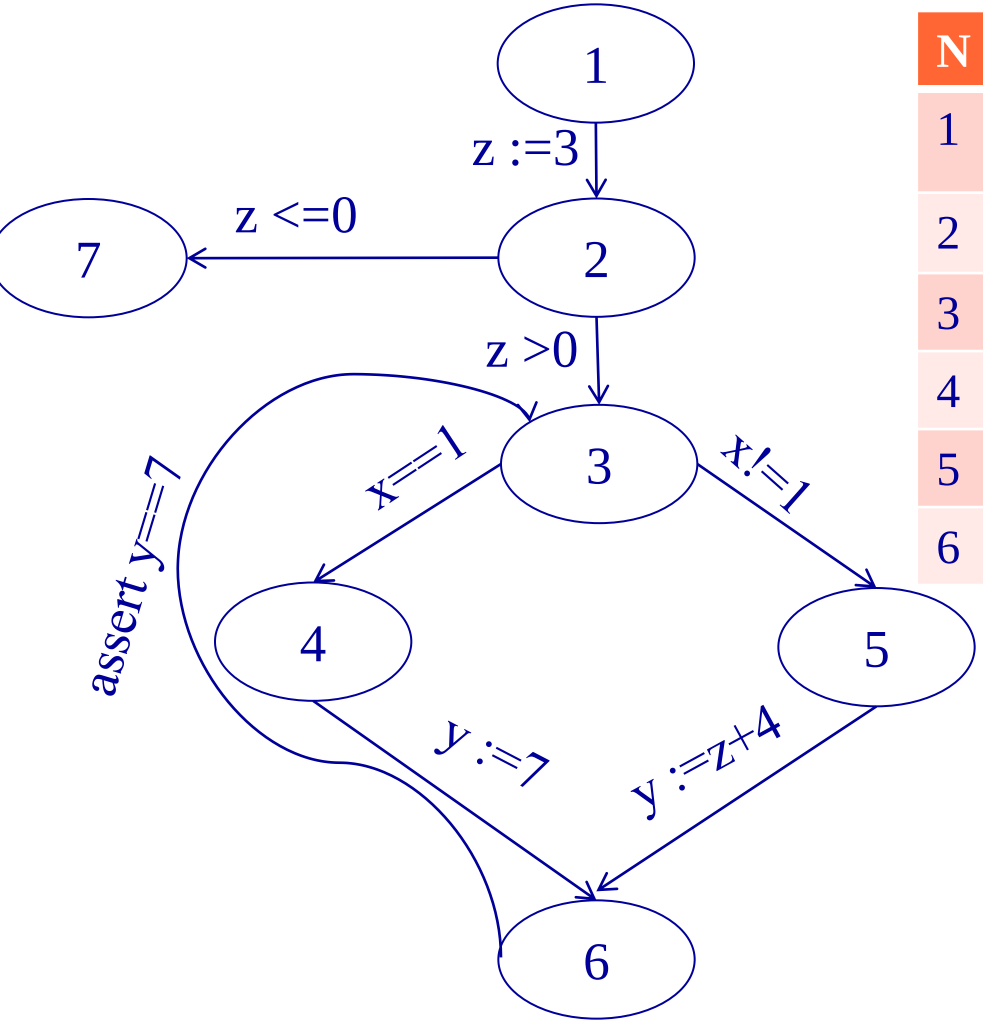
new := $df_{entry}(v) \sqcup$ temp

if (new ≠ $df_{entry}[v]$) then

$df_{entry}[v]$:= new;

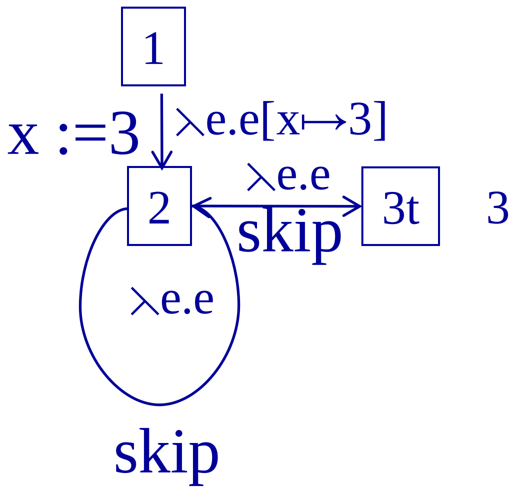
WL := WL ∪{v}

Constant Propagation



N	Value	WL
1	$[x \mapsto ?, y \mapsto ?, z \mapsto ?]$	{2, 3, 4, 5, 6, 7}
2	$[x \mapsto ?, y \mapsto ?, z \mapsto 3]$	{ 3, 4, 5, 6, 7}
3	$[x \mapsto ?, y \mapsto ?, z \mapsto 3]$	{ 4, 5, 6, 7}
4	$[x \mapsto 1, y \mapsto 7, z \mapsto 3]$	{5, 6, 7}
5	$[x \mapsto ?, y \mapsto 7, z \mapsto 3]$	{6, 7}
6	$[x \mapsto ?, y \mapsto 7, z \mapsto 3]$	{7}

Example Constant Propagation



$$DF(1) = [x \mapsto 0]$$

$$DF(2) = DF(1)[x \mapsto 3] \sqcup DF(2)$$

$$DF(3) = DF(2)$$

DF[1]	DF[2]	DF[3]
$[x \mapsto 0]$	$[x \mapsto 3]$	$[x \mapsto 3]$
$[x \mapsto 0]$	$[x \mapsto ?]$	$[x \mapsto ?]$
$[x \mapsto 7]$	$[x \mapsto 9]$	$[x \mapsto 7]$
$[x \mapsto ?]$	$[x \mapsto 3]$	$[x \mapsto 3]$

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)

Soundness

- Every detected constant is indeed such
- Every error will be detected
- The least fixed points represents all occurring runtime states
- Next week

Completeness

- Every constant is indeed detected as such
- Every detected error is real
- Cannot be guaranteed in general

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 $f[e_1, e_2, \ldots, e_n]: L \to L$ 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] $= \sqcup\{f[e_1, e_2, \ldots, e_n](\iota)$
  $[e_1, e_2, \ldots, e_n] \in \text{paths}(v)\}$

JOP vs. Least Solution

- The DF solution obtained by Chaotic iteration satisfies for every *l*:
  - $\mathrm{JOP}[v] \sqsubseteq \mathrm{DF}_{entry}(v)$
- 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
  - $\mathrm{JOP}[v] = \mathrm{DF}_{entry}(v)$

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

Widening

- Accelerate the termination of Chaotic iterations by computing a more conservative solution
- Can handle lattices of infinite heights

Specialized Chaotic Iterations+ ∇

Chaotic(G(V, E): Graph, s: Node, L: lattice, ι: L, f: E $\rightarrow$(L $\rightarrow$L)){

for each v in V to n do $df_{entry}[v] := \bot$

In[v] = ι

WL = {s}

while (WL $\neq$ Ø) do

select and remove an element u $\in$ WL

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

temp = $f(e)(df_{entry}[u])$

new := $df_{entry}(v) \nabla$ temp

if (new $\neq df_{entry}[v]$) then

$df_{entry}[v]$:= new;

WL := WL $\cup$\{v\}

Example Interval Analysis

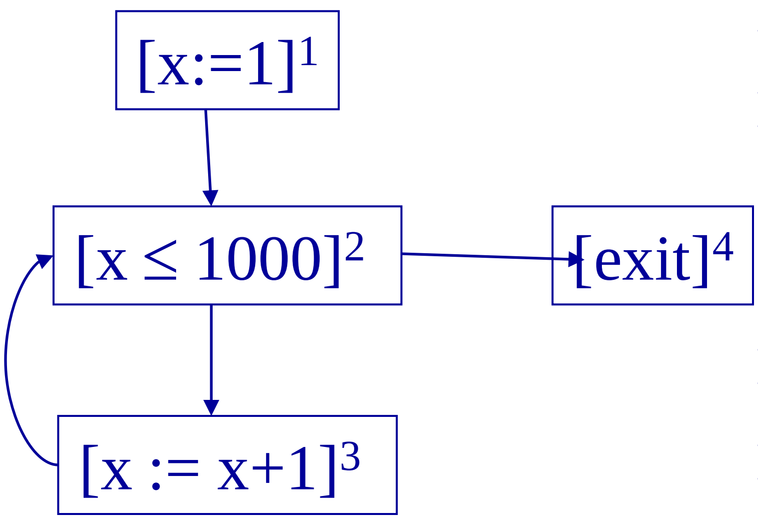
- Find a lower and an upper bound of the value of a variable
- Usages?
- Lattice
  $L = (Z \cup \{-\infty, \infty\} \times Z \cup \{-\infty, \infty\}, \sqsubseteq, \sqcup, \sqcap, \bot, \top)$
  - $[a, b] \sqsubseteq [c, d]$ if $c \leq a$ and $d \geq b$
  - $[a, b] \sqcup [c, d] = [\min(a, c), \max(b, d)]$
  - $[a, b] \sqcap [c, d] = [\max(a, c), \min(b, d)]$
  - $\top =$
  - $\bot =$

Example Program
Interval Analysis

$[x := 1]^1$;

while $[x \leq 1000]^2$ do

$[x := x + 1;]^3$



IntEntry(1) = [minint,maxint]

IntExit(1) = [1,1]

IntEntry(2) = IntExit(1) $\sqcup$ IntExit(3)

IntExit(2) = IntEntry(2)

IntEntry(3) = IntExit(2) $\sqcap$ [minint,1000]

IntExit(3) = IntEntry(3)+[1,1]

IntEntry(4) = IntExit(2) $\sqcap$ [1001,maxint]

IntExit(4) = IntEntry(4)

Widening for Interval Analysis

- $\perp \nabla [c, d] = [c, d]$
- $[a, b] \ \nabla [c, d] = [$

  if $a \leq c$

  then a

  else $-\infty$,

  if $b \geq d$

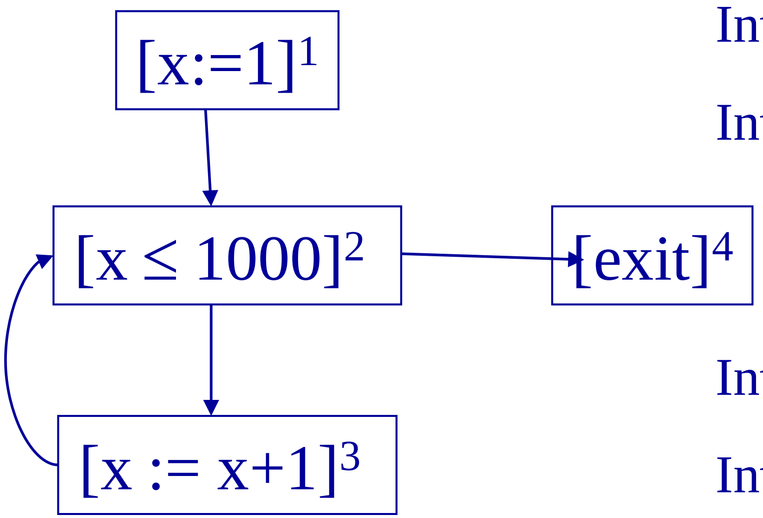
  then b

  else ∞

  $]$

Example Program
Interval Analysis

$[x := 1]^1$;
while $[x \leq 1000]^2$ do
$\quad [x := x + 1;]^3$



IntEntry(1) = $[-\infty, \infty]$

IntExit(1) = $[1,1]$

IntEntry(2) = InExit(2) ∇ (IntExit(1) $\sqcup$ IntExit(3))

IntExit(2) = IntEntry(2)

IntEntry(3) = IntExit(2) $\sqcap$ $[-\infty,1000]$

IntExit(3) = IntEntry(3)+$[1,1]$

IntEntry(4) = IntExit(2) $\sqcap$ $[1001, \infty]$

IntExit(4) = IntEntry(4)

Requirements on Widening

- For all elements $l_1 \sqcup l_2 \sqsubseteq l_1 \nabla l_2$
- For all ascending chains
  $l_0 \sqsubseteq l_1 \sqsubseteq l_2 \sqsubseteq \ldots$
  the following sequence is finite
  - $y_0 = l_0$
  - $y_{i+1} = y_i \nabla l_{i+1}$
- For a monotonic function
  $f: L \to L$
  define
  - $x_0 = \bot$
  - $x_{i+1} = x_i \nabla f(x_i)$
- Theorem:
  - There exits k such that $x_{k+1} = x_k$
  - $x_k \in Red(f) = \{l: l \in L, f(l) \sqsubseteq l\}$

Narrowing

- Improve the result of widening
- $y \sqsubseteq x \Rightarrow y \sqsubseteq (x \triangle y) \sqsubseteq x$
- For all decreasing chains $x_0 \sqsupseteq x_1 \sqsupseteq \ldots$ the following sequence is finite
  - $y_0 = x_0$
  - $y_{i+1} = y_i \triangle x_{i+1}$
- For a monotonic function $f: L \rightarrow L$ and $x \in Red(f) = \{l: l \in L, f(l) \sqsubseteq l\}$ define
  - $y_0 = x$
  - $y_{i+1} = y_i \triangle f(y_i)$
- Theorem:
  - There exits k such that $y_{k+1} = y_k$
  - $y_k \in Red(f) = \{l: l \in L, f(l) \sqsubseteq l\}$

Narrowing for Interval Analysis

- $[a, b] \bigtriangleup \perp = [a, b]$
- $[a, b] \bigtriangleup [c, d] = [$

  if $a = -\infty$

  then c

  else a,

  if $b = \infty$

  then d

  else b

  $]$

Example Program
Interval Analysis

$[x := 1]^1$;
while $[x \leq 1000]^2$ do
$\quad [x := x + 1;]^3$

$IntEntry(1) = [-\infty , \infty]$

$IntExit(1) = [1,1]$

$IntEntry(2) = InExit(2) \bigtriangleup (IntExit(1) \sqcup IntExit(3))$

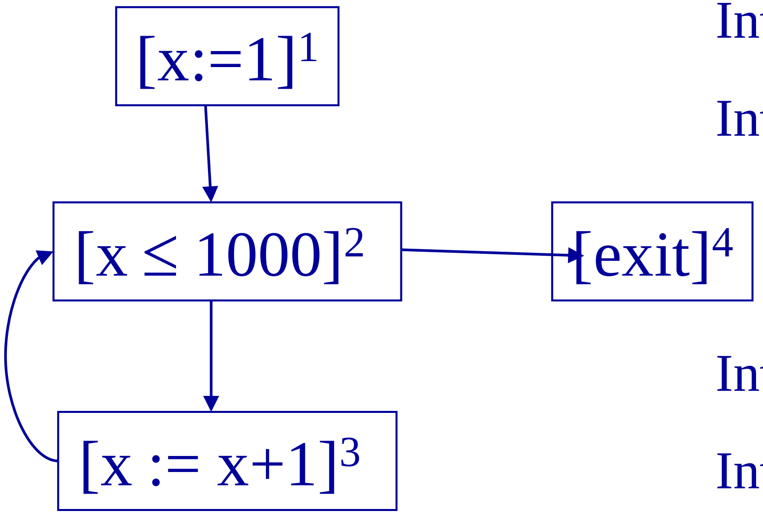
$IntExit(2) = IntEntry(2)$

$IntEntry(3) = IntExit(2) \sqcap [-\infty,1000]$

$IntExit(3) = IntEntry(3)+[1,1]$

$IntEntry(4) = IntExit(2) \sqcap [1001, \infty]$

$IntExit(4) = IntEntry(4)$



Non Montonicity of Widening

- $[0,1] \bigtriangledown [0,2] = [0, \infty]$
- $[0,2] \bigtriangledown [0,2] = [0,2]$

Widening and Narrowing Summary

- Very simple but produces impressive precision
- Sometimes non-monotonic
- The McCarthy 91 function

```
int f(x) [-∞ , ∞]
  if x > 100
     then [101, ∞] return x -10 [91, ∞-10];
     else [-∞, 100] return f(f(x+11)) [91, 91] ;
```

- Also useful in the finite case
- Can be used as a methodological tool

Conclusions

- Chaotic iterations is a powerful technique
- Easy to implement
- Rather precise
- But expensive
  - More efficient methods exist for structured programs