

# Iterative Program Analysis

Mooly Sagiv

<http://www.cs.tau.ac.il/~msagiv/courses/pa.html>

Tel Aviv University

640-6706

Textbook: Principles of Program Analysis

Chapter 2.1 +6 (modified)

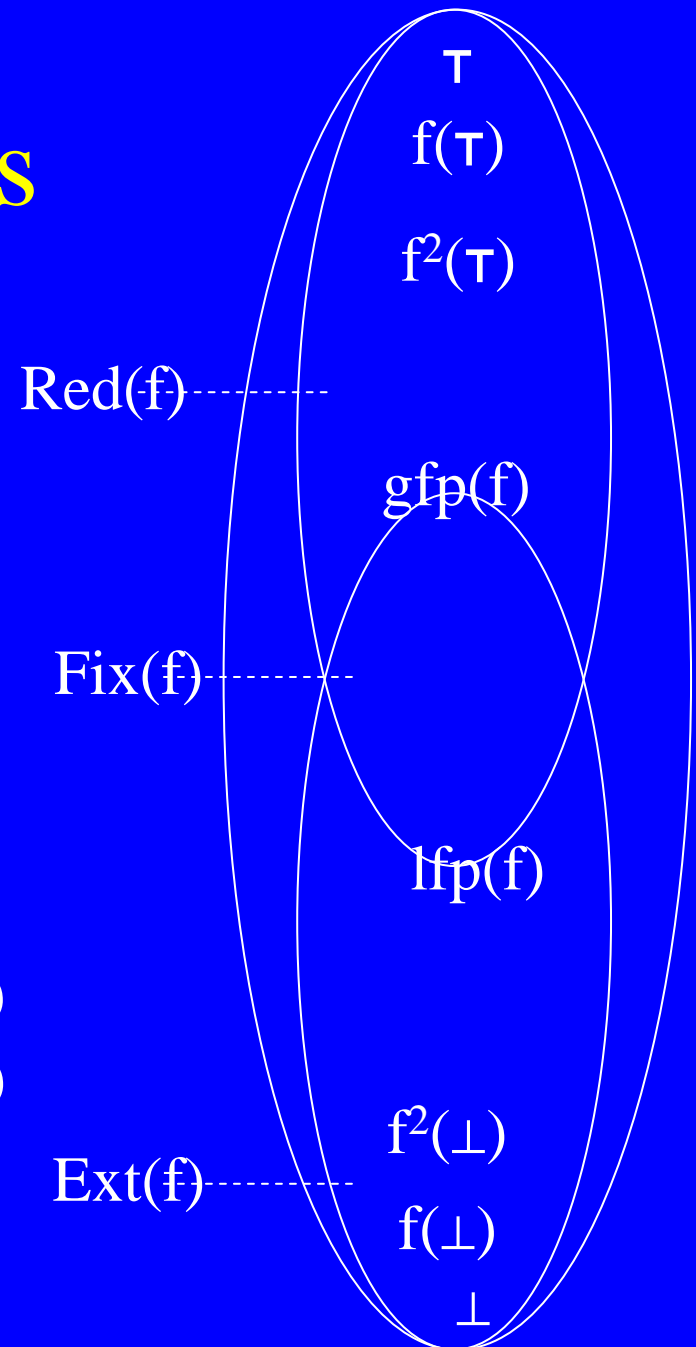
Appendix A

# Subjects

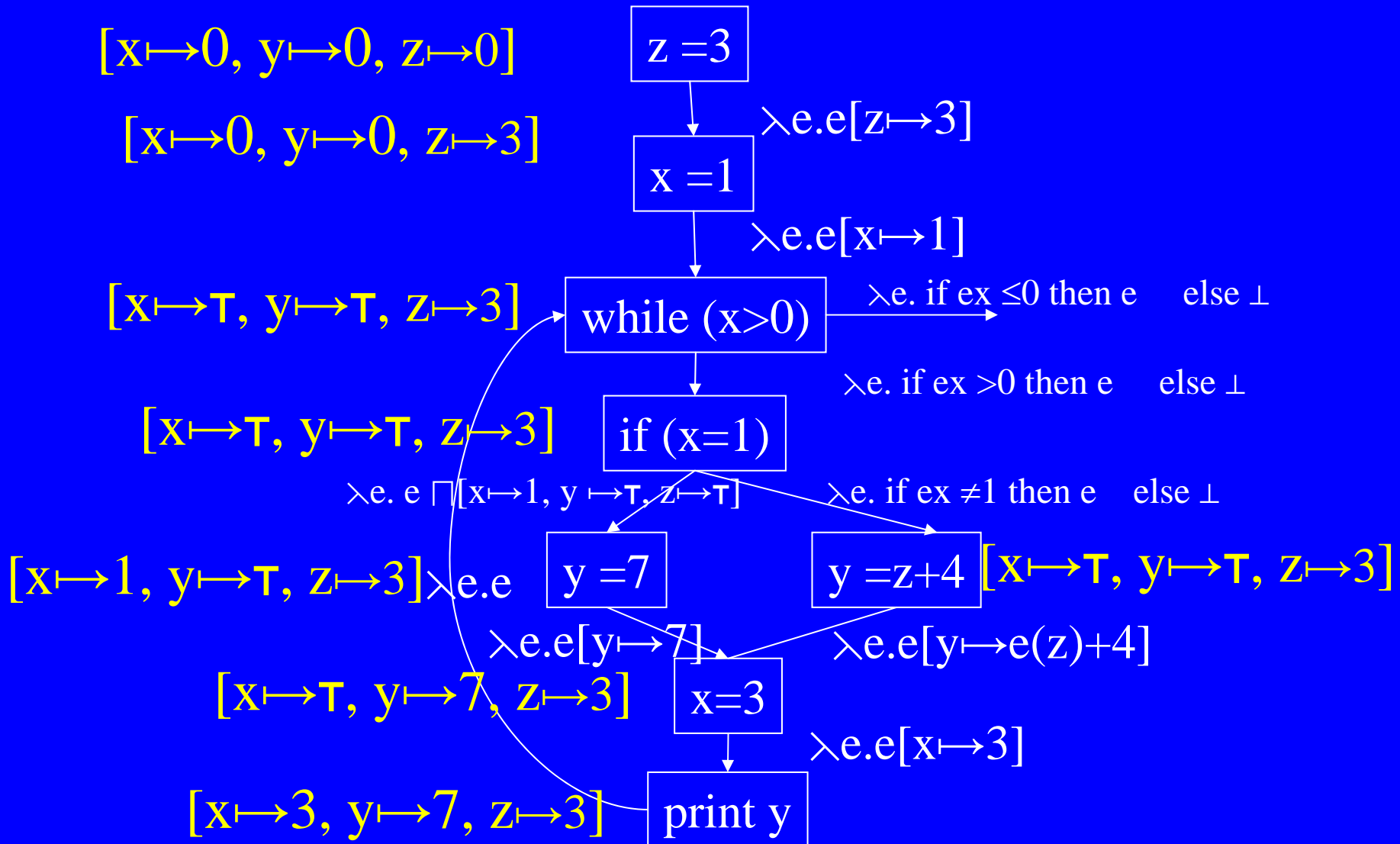
- ◆ **Why least fixed point**
- ◆ **The Monotone Framework**

# Fixed Points

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



# Least Fixed Point



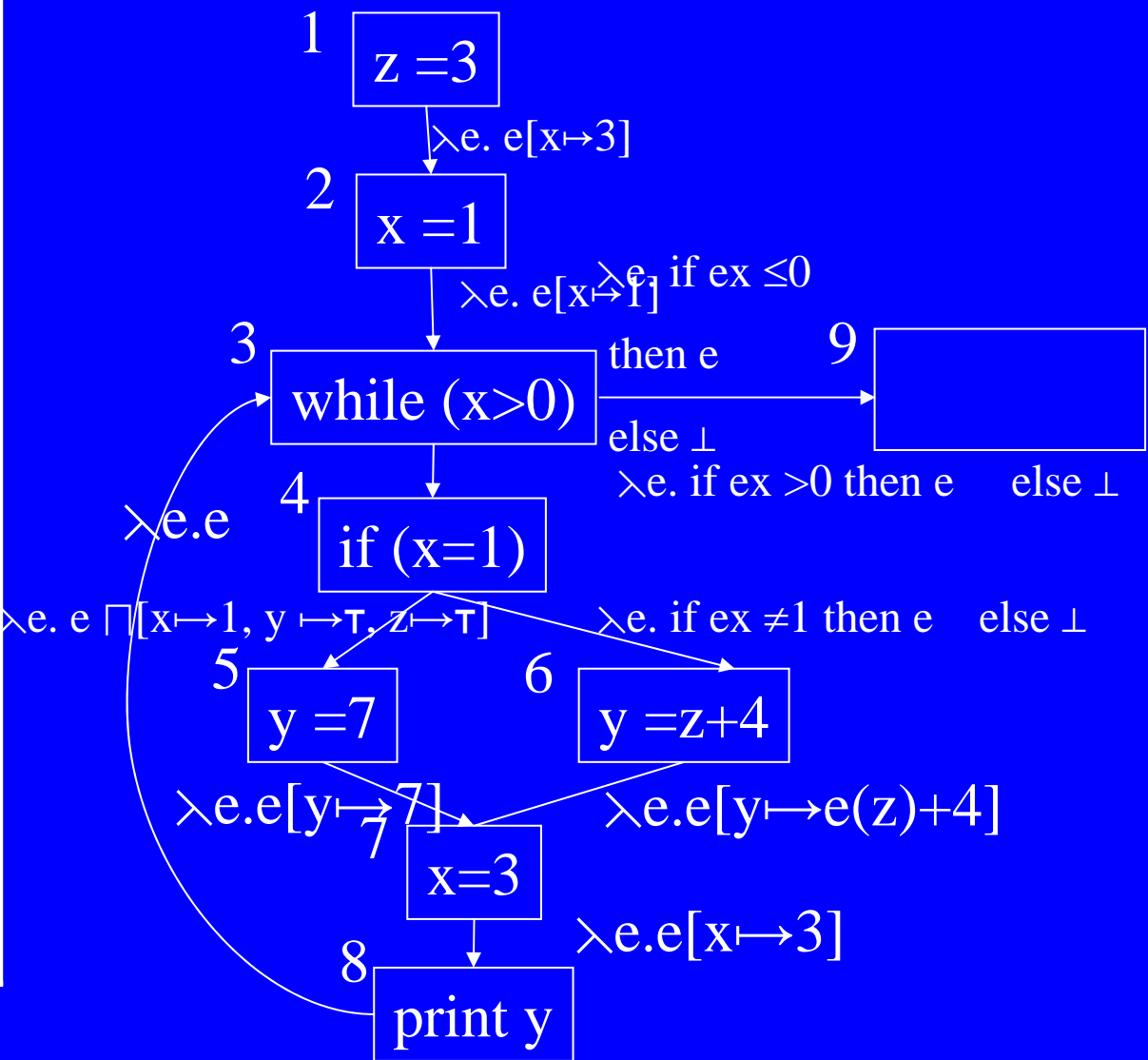
# Monotone Forward Frameworks

- ◆ A complete lattice  $(L, \sqsubseteq, \sqcup, \sqcap, \perp, \top)$  describes the “potential pieces of information”
- ◆ The initial value at entry is specified by  $\iota \in L$
- ◆ The effect of every control flow edge  $e$  is described by a monotone function  $f_e : L \rightarrow L$  (transfer function)
- ◆ Compute lfp of the following system of equations (forward)

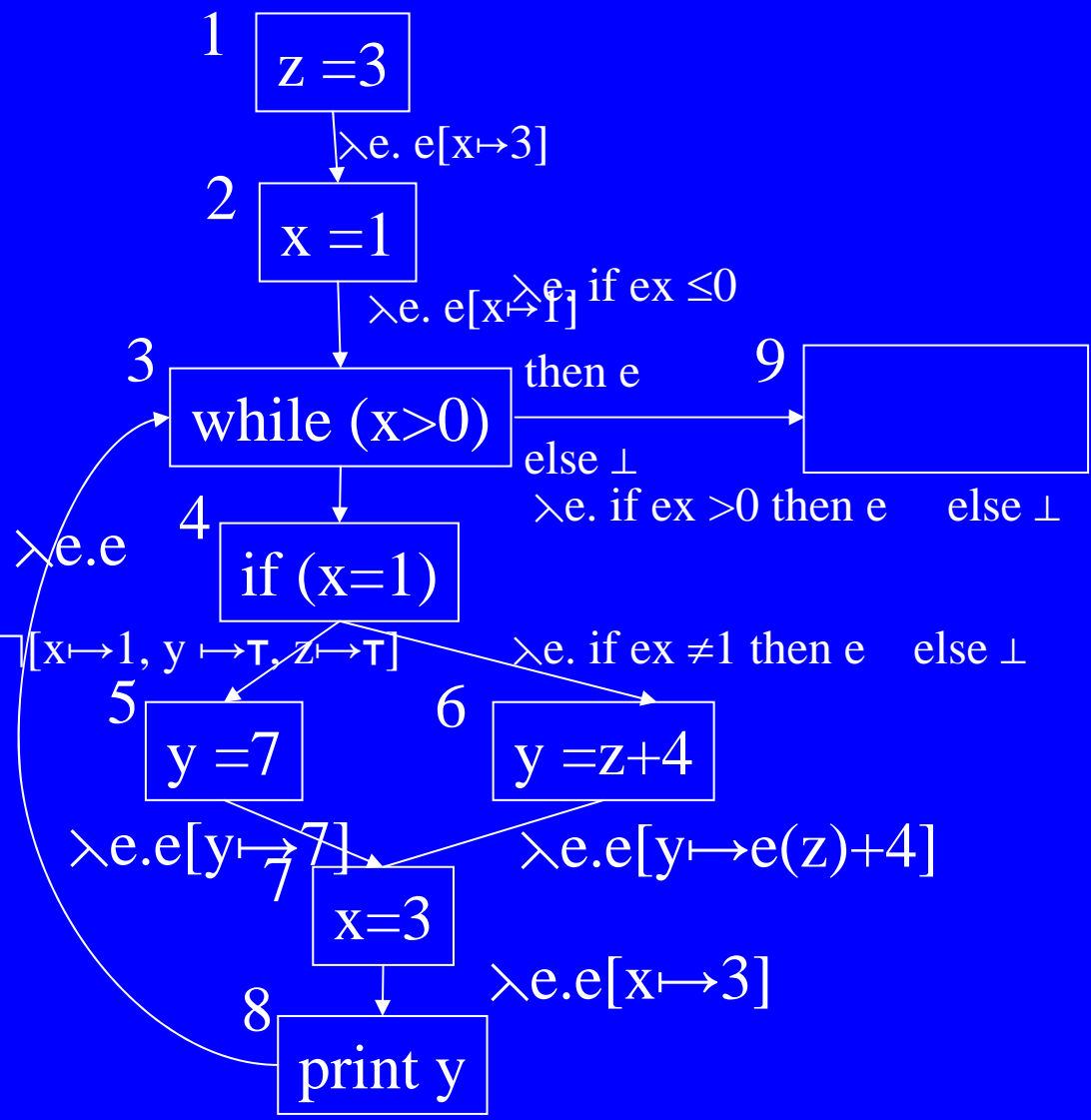
$$\text{DF}(\text{entry}) = \iota$$

$$\text{DF}(v) = \sqcup \{f_{(u,v)}(\text{DF}(u)) \mid (u, v) \in E\}$$

DF(1)= $\perp$
DF(2)= $\lambda e. e[x \mapsto 3]$ DF(1)
DF(3)= $\lambda e. e[x \mapsto 1]$ DF(2) $\sqcup$ $\lambda e. e$ DF(8)
DF(4)= $\lambda e. \text{if } ex > 0 \text{ then } e$ $\text{else } \perp$ DF(3)
DF(5)= $\lambda e. e \sqcap [x \mapsto 1, y$ $\mapsto \tau, z \mapsto \tau]$ DF(4)
DF(6)= $\lambda e. \text{if } ex \neq 1 \text{ then } e$ $\text{else } \perp$ DF(4)
DF(7)= $\lambda e. e[y \mapsto 7]$ DF(5) $\sqcup \lambda e. e[y \mapsto e(z)+4]$ DF(6)
DF(8)= $\lambda e. e[x \mapsto 3]$ DF(7)
DF(9)= $e. \text{if } ex \leq 0$ $\text{then } e \text{ else } \perp$ DF(3)



DF(1)= $\perp$
DF(2)=DF(1)[ $x \mapsto 3$ ]
DF(3)=DF(2)[ $x \mapsto 1$ ] $\sqcup$ DF(8)
DF(4)= if DF(3) $x > 0$ then DF(3) else $\perp$
DF(5)= DF(4) $\sqcap$ [ $x \mapsto 1, y \mapsto \top, z \mapsto \top$ ]
DF(6)= if DF(4) $x \neq 1$ then DF(4) else $\perp$
DF(7)= DF(5) [ $y \mapsto 7$ ] $\sqcup$ DF(6)[ $y \mapsto e(z)+4$ ]
DF(8)= DF(7)[ $x \mapsto 3$ ]
DF(9)=if DF(3) $x \leq 0$ then DF(3) else $\perp$



# Chaotic Iterations for forward problems

**for**  $v \in N$  **do**

$DF(v) := \perp$

$DF(\text{entry}) := \top$

$WL = N$

**while**  $WL \neq \emptyset$  **do**

    Select and remove an arbitrary  $u \in WL$

**for** every edge  $(u, v) \in E$

$\text{temp} = DF(v) \sqcup f_e(DF(u))$

**if**  $(\text{temp} \neq DF(v))$

$DF(v) := \text{temp}$

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

# Monotone Backward Frameworks

- ◆ A complete lattice  $(L, \sqsubseteq, \sqcup, \sqcap, \perp, \top)$  describes the “potential pieces of information”
- ◆ The initial value at exit is specified by  $\iota \in L$
- ◆ The (backward) effect of every control flow edge  $e$  is described by a monotone function  $f_e : L \rightarrow L$  (transfer function)
- ◆ Compute lfp of the following system of equations (backward)

$$DF(\text{exit}) = \iota$$

$$DF(v) = \sqcup \{ f(u,v)(DF(u)) \mid (v, u) \in E \}$$

# Chaotic Iterations for backward problems

**for**  $v \in N$  **do**

$DF(v) := \perp$

$DF(\text{exit}) := \top$

$WL = N$

**while**  $WL \neq \emptyset$  **do**

    Select and remove an arbitrary  $u \in WL$

**for** every edge  $(v, u) \in E$

$\text{temp} = DF(v) \sqcup f_e(DF(u))$

**if**  $(\text{temp} \neq DF(v))$

$DF(v) := \text{temp}$

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

# A Simple Example

*/\* c \*/*

L0:     a := 0

*/\* ac \*/*

L1:     b := a + 1

*/\* bc \*/*

      c := c + b

*/\* bc \*/*

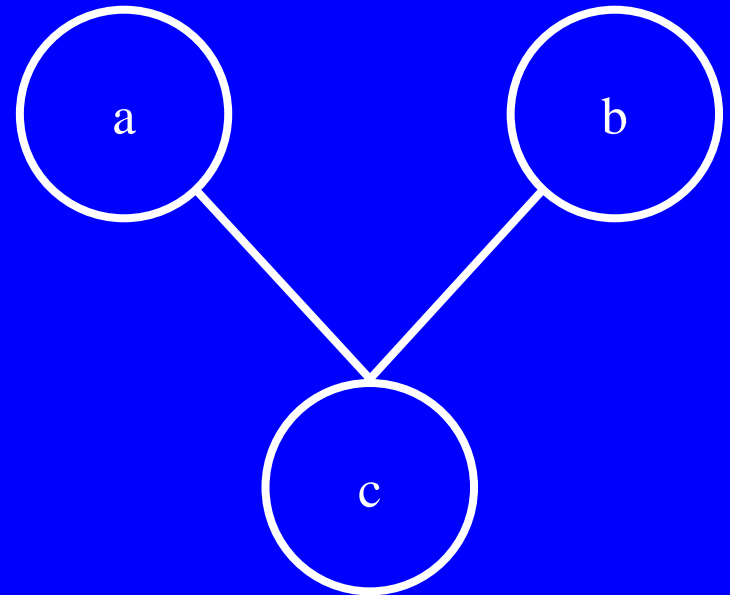
      a := b \* 2

*/\* ac \*/*

      if c < N goto L1

*/\* c \*/*

      return c



# Instances of Monotone Frameworks

- ◆ Kill/Gen Problems
  - Reaching Definitions
  - Available Expressions
  - Live Variables
  - $\sqcup = \cup$  or  $\sqcup = \cap$
  - $f_e(\text{entry}) = (\text{entry} - \text{kill}(e)) \cup \text{gen}(e)$
- ◆ Forward
  - May be uninitialized (garbage) variables
  - Constant propagation
  - Sign-analysis
  - Parity analysis
  - Points-to analysis
  - Shape Analysis
  - Driver verification
- ◆ Backward
  - Truly-live variables

# May-be-garbage variables

- ◆ A variable **may-be-garbage at a label  $l$**  if there may be a path to  $l$  in which it is either uninitialized or set using an uninitialized variable

```
x := 5 ;
```

```
if z > 2
```

```
    then y := 17 ;
```

```
    else skip ;
```

```
t := y + x ;
```

# May-be-garbage variables(cont)

- ◆  $L = (\mathcal{P}(\text{Var}), \subseteq, \cup, \cap, \emptyset, \text{Var})$
- ◆ Initial value  $\perp = \text{Var}_*$
- ◆ Transfer functions  $f_e(\text{DF})$

$x := a$       if  $\text{FV}(a) \cap \text{DF} \neq \emptyset$   
                  then  $\text{DF} \cup \{x\}$   
                  else  $\text{DF} - \{x\}$

skip           $\text{DF}$

$b$              $\text{DF}$

# The Factorial Program

```
y := x;
```

```
z := 1;
```

```
while y > 1 do (
```

```
    z := z * y;
```

```
    y := y - 1;
```

```
)
```

```
y := 0;
```

# Over Conservative Solution

```
if  $y > 1$   
    then  $z := 1$ ;
```

...

```
if  $y > 1$   
    then  $y := z$ ;
```

# Constant Propagation

- ◆ Determine variables with constant values
- ◆ Information Lattice
  - Extended integer lattice  $(L_1, \sqsubseteq_1, \sqcup_1, \sqcap_1, \perp_1, \top_1)$ 
    - »  $L_1 = \mathbb{Z} \cup \{\perp_1, \top_1\}$
    - »  $\perp_1 \sqsubseteq_1 z \sqsubseteq_1 \top_1$
  - Define  $L = (S \rightarrow L_1, \sqsubseteq)$  where  $S = \text{Var}_*$
  - Initial value  $\iota$  ?
- ◆ Transfer functions  $A_{cp}: AExp \rightarrow (L \rightarrow L_1)$   
 $f_e(\text{DF})$

$$x := a \quad \text{DF}[x \mapsto A_{cp}[[a]](\text{DF})]$$

$$\text{skip} \quad \text{DF}$$

$$x = c \quad \text{DF} \sqcap [x \mapsto c, y \mapsto \top]$$

```
y := 5;  
x := 8;  
while x > 1 do (  
    x := y + 3;  
    x := (y * y) - 17;  
)
```

# Points-To Analysis

- ◆ Determine if a pointer variable  $p$  may point to  $q$
- ◆  $L = (\mathcal{P}(\text{Var} \times \text{Var}), \subseteq, \cup, \cap, \emptyset, \text{Var} \times \text{Var})$
- ◆  $(x, y) \in l \dashv\vdash x$  may hold the address of  $y$
- ◆ The initial value  $\iota = \emptyset$
- ◆ Transfer functions  $f_e(\text{DF})$

$x := \&y$

$x := y$

$x := *y$

$*x := y$

# Usage of Points-To-Information

- ◆ “Adapt” other optimizations

- Constant propagation

```
x = 5;  
*p = 7 ;  
... x ...
```

- ◆ Pointer aliases

- Variables p and q are **may-aliases** at l if the points-to set at l contains entries (p, x) and (q, x)

- ◆ Side effect analysis

```
*p = *q + * * t
```

t := &a;

y := &b;

z := &c;

if x > 0;

    then p := &y;

    else p := &z;

p := t;

```

/*  $\emptyset$  */ 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;

# Complexity of Chaotic Iterations

## ◆ Parameters:

- N number of nodes
- k is the maximum outdegree
- A lattice of height h
- c is the maximum cost of
  - » applying  $f_i$
  - »  $\sqcup$
  - » L comparisons

## ◆ Complexity

$$O(N * h * c * k)$$

# Soundness of Chaotic Iterations

- ◆ Later with abstract interpretation

# Precision of Chaotic Iterations

- ◆ Optimal
- ◆ Join-over-all-paths - No loss of information w.r.t. straight line code
- ◆ Relatively optimal (induced) w.r.t. the abstraction
- ◆ Compare at run-time
- ◆ Good enough for the used optimization

# The Join-Over-All-Paths (JOP)

- ◆ Let  $\text{paths}(\text{entry}, v)$  denote the potentially infinite set of paths from entry to  $v$  (written as sequences of edges)
- ◆ For a sequence of edges  $[e_1, e_2, \dots, e_n]$  define  $f[e_1, e_2, \dots, e_n]: L \rightarrow L$  by composing the effects of edges
  - $f[\varepsilon](s) = s$
  - $f[p.e](s) = f[p](f_e(s))$
- ◆  $\text{JOP}_l = \sqcup \{f[e_1, e_2, \dots, e](l) \mid [e_1, e_2, \dots, e] \in \text{paths}(\text{entry}, l)\}$

# JOP vs. Least Solution

- ◆ The DF solution obtained by Chaotic iteration satisfies for every  $v$ :
  - $\text{JOP}_v \sqsubseteq \text{DF}(v)$
- ◆ If every  $f_e$  is additive (distributive) for all the labels  $l$ 
  - $\text{JOP}_v = \text{DF}(v)$

# Additive (Distributive) Monotone Problems

- ◆ Kill/Gen Problems
- ◆ May-be uninitialized
- ◆ Truly-live variables
- ◆ Linear constant propagation with one variable
- ◆ Points-to analysis with one level pointers

# Non Distributive Monotone Problems

- ◆ Points-to-analysis
- ◆ Constant Propagation on arbitrary expressions

# Converting into Distributive Frameworks

## ◆ Consider

- a **finite** lattice  $(L, \sqsubseteq, \sqcup, \sqcap, \perp, \top)$
- An initial value at entry  $\iota \in L$
- The effect of every edge at  $e$  is described by a monotone function  $f_e : L \rightarrow L$  (transfer function)
- Define a distributive function on  $(P(L), \subseteq, \cup, \cap, \emptyset, L)$  by  $F(X) = \{f_i(x) : x \in X\}$
- Solve the following system of equations over  $P(L)$

$$DF(\text{entry}) = \{\iota\}$$

$$DF(v) = \cup \{f_{(u,v)}(x) \mid (u, v) \in E, x \in DF(u)\}$$

# The Constant Propagation

- ◆ It is undecidable to find the JOP in the constant propagation problem
- ◆ A Sketch of a proof

```
while (...) (  
    if (...) x_1 = x_1 + 1;  
    if (...) x_2 = x_2 + 1;  
    ...  
    if (...) x_n = x_n + 1;  
}  
y = truncate (1/ (1 + p2(x_1, x_2, ..., x_n))  
/* Is y=0 here? */
```

# Static Analysis problems beyond Monotone Frameworks

## ◆ Infinite heights

- integer intervals
- Linear relationships between variables

## ◆ Bi-directional problems

- Partial-redundant expressions
- Automatic inference of variable types in imperative (weakly typed) programs

$x := b[z]$

$a [b[y]] := x$

## ◆ Procedures

# Historical Perspective

- ◆ 1973 Kildall defined the basic framework but required distributive frameworks
- ◆ 1976 Kam and Ulmann defined Monotone Framework
- ◆ 1980 Tarjan suggested an almost linear time algorithm for reducible flow graph
- ◆ 1980 Rosen suggested a linear time algorithm for high level language

# Static Analysis problems beyond Monotone Frameworks

## ◆ Infinite heights

- integer intervals
- Linear relationships between variables

## ◆ Bi-directional problems

- Partial-redundant expressions
- Automatic inference of variable types in imperative (weakly typed) programs

$x := b[z]$

$a [b[y]] := x$

## ◆ Procedures

# Conclusions

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