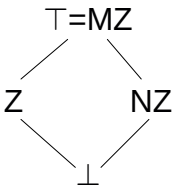
Dataflow Analysis, Continued

17-654/17-765
Analysis of Software Artifacts
Jonathan Aldrich

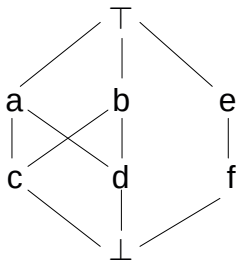
Lattice

- A lattice is a tuple (L, $\sqsubseteq$, $\sqcup$, $\bot$, $\top$)
  - L is a set of abstract elements
  - $\sqsubseteq$ is a partial order on L
    - Means *at least as precise as*
  - $\sqcup$ is the least upper bound of two elements
    - Must exist for every two elements in L
    - Used to merge two abstract values
  - $\bot$ (bottom) is the least element of L
    - Means we haven't yet analyzed this yet
    - Will become clear later
  - $\top$ (top) is the greatest element of L
    - Means we don't know anything



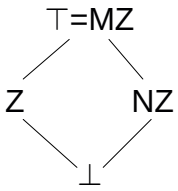
Clarification: Least Upper Bounds

- x ⊔ y = z iff
  - z is an upper bound of x and y
    - x ⊑ z and y ⊑ z
  - z is the least such bound
    - ∀w∈*L* such that x ⊑ w and y ⊑ w we have z ⊑ w
- Also called a join
- Not a lattice
  - What is c ⊔ d?
  - a, b, and ⊤ are upper bounds
    - Assume ⊑ is transitive
  - None is least upper bound



Zero Analysis Lattice

- Integer zero lattice
  - L_{ZI} = { ⊥, Z, NZ, MZ }
  - ⊥ ⊑ Z, ⊥ ⊑ NZ, NZ ⊑ MZ, Z ⊑ MZ
    - ⊥ ⊑ MZ holds by transitivity
  - ⊔ defined as join for ⊑
    - x ⊔ y = z iff
      - z is an upper bound of x and y
      - z is the least such bound
    - Obeys laws: ⊥ ⊔ $\mathcal{X}$ = $\mathcal{X}$, ⊤ ⊔ $\mathcal{X}$ = ⊤, $\mathcal{X}$ ⊔ $\mathcal{X}$ = $\mathcal{X}$
    - Also Z ⊔ NZ = MZ
  - ⊥ = ⊥
    - ∀$\mathcal{X}$. ⊥ ⊑ $\mathcal{X}$
  - ⊤ = MZ
    - ∀$\mathcal{X}$. $\mathcal{X}$ ⊑ ⊤



Abstraction Function

- Maps each concrete program state to a lattice element
  - For tuple lattices, the function can be defined for values and lifted to tuples
- Integer Zero abstraction function α_{ZI}:
  - $\alpha_{ZI}(0) = Z$
  - $\alpha_{ZI}(n) = NZ$ for all $n \neq 0$
- Zero Analysis abstraction function α_{ZA}:
  - $\alpha_{ZA}(\eta) = \{x \mapsto \alpha_{ZI}(\eta(x)) \mid x \in \textbf{Var}\}$
  - This is just the tuple form of $\alpha_{ZI}(n)$
    - Can be done for any tuple lattice

Control Flow Graph (CFG)

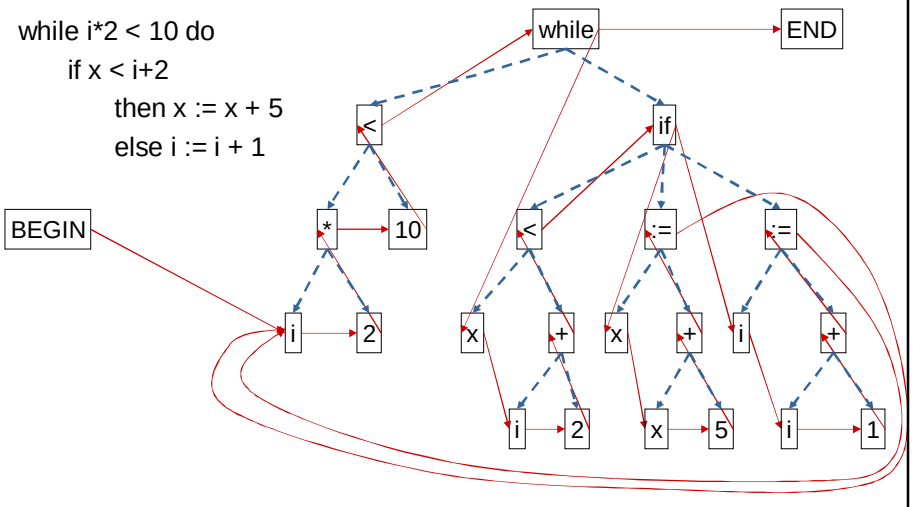
- Shows order of statement execution
  - Determines where data flows
- Decomposes expressions into primitive operations
  - Crystal: One CFG node per "useful" AST node
    - constants, variables, binary operations, assignments, if, while…
  - Loops are written out
    - Form a loop in the CFG
  - Benefit: analysis is defined one operation at a time

Intuition for Building a CFG

- Connect nodes in order of operation
  - Defined by language
- Java order of operation
  - Expressions, assignment, sequence
    - Evaluate subexpressions left to right
    - Evaluate node after children (postfix)
  - While, If
    - Evaluate condition first, then if/while
    - if branches to else and then
    - while branches to loop body and exit

Control Flow Graph Example

```
while i*2 < 10 do
    if x < i+2
        then x := x + 5
        else i := i + 1
```



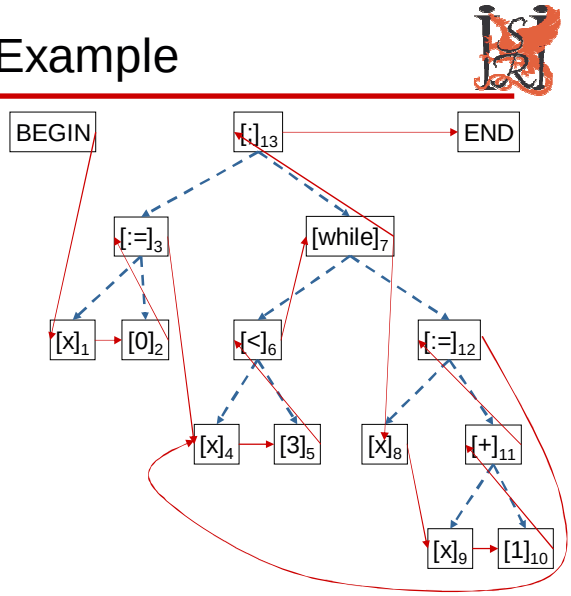
Flow Functions

- Compute dataflow information after a statement from dataflow information before the statement
  - Formally, map a lattice element and a CFG node to a new lattice element
- Expression flow functions
  - Treat each expression as an assignment to a temporary variable
    - x+3*y ➔ t_1:=x; t_2:=3; t_3:=y; t_4:=t_2*t_3; t_5:=t_1+t_4
  - That variable is used in containing expressions
  - Instead of explicitly writing temporaries, we'll keep track of them by labeling nodes
    - $[[x]_1 + [[3]_2 * [y]_3]_4]_5$

Zero Analysis Flow Functions

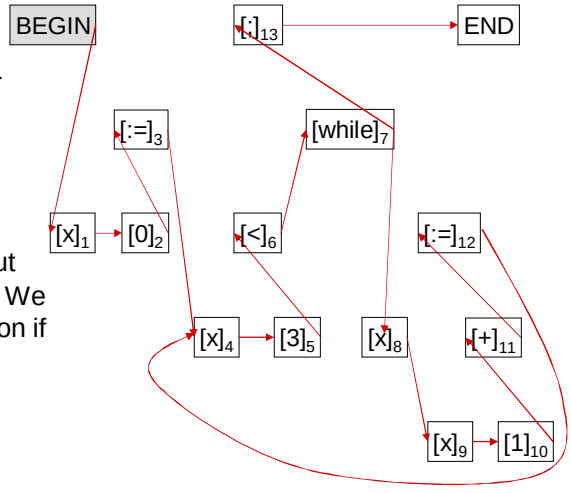
- $f_{ZA}(\sigma, [x]_k) = [t_k \mapsto \sigma(x)]\ \sigma$
- $f_{ZA}(\sigma, [n]_k)$ = if n==0
  then $[t_k \mapsto Z]\sigma$
  else $[t_k \mapsto NZ]\sigma$
- $f_{ZA}(\sigma, [x := [\ldots]_n]_k) = [x \mapsto \sigma(t_n)]\ \sigma$
- $f_{ZA}(\sigma, [[\ldots]_n\ op\ [\ldots]_m]_k) = [t_k \mapsto MZ]\ \sigma$
  - Could be more precise, e.g.
    $f_{ZA}(\sigma, [[\ldots]_n + [\ldots]_m]_k)$ =
    if $\sigma[t_n]$=Z && $\sigma[t_m]$=Z
    then $[t_k \mapsto Z]\sigma$ else $[t_k \mapsto MZ]\sigma$
- $f_{ZA}(\sigma$, /* *any other* */$) = \sigma$

Zero Analysis Example



x := 0;
while x > 3 do
    x := x+1

Zero Analysis Example



Initial dataflow

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }

Intuition:

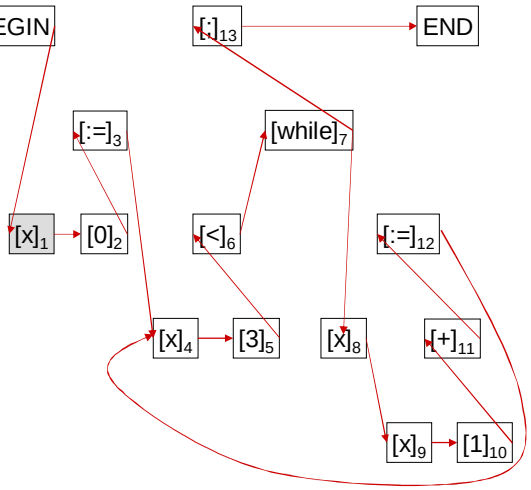
We know nothing about initial variable values. We could use a precondition if we had one.

Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }

$\sigma_1 = f_{ZA}(\sigma_\iota, [x]_1)$
$= [t_1 \mapsto \sigma_\iota(x)]\ \sigma_\iota$
$= [t_1 \mapsto MZ]\ \sigma_\iota$
$= \sigma_\iota$

$f_{ZA}(\sigma, [x]_k) = [t_k \mapsto \sigma(x)]\ \sigma$

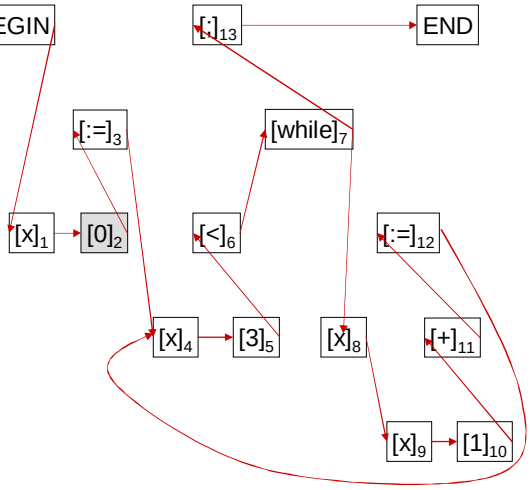


Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
$\sigma_1 = \sigma_\iota$

$\sigma_2 = f_{ZA}(\sigma_1, [0]_2)$
$= [t_2 \mapsto Z]\ \sigma_1$
$= [t_2 \mapsto Z]\ \sigma_\iota$

$f_{ZA}(\sigma, [n]_k) =$
if n==0
then $[t_k \mapsto Z]\sigma$
else $[t_k \mapsto NZ]\sigma$

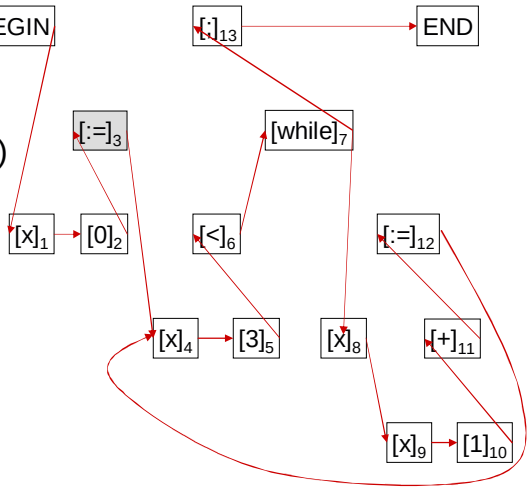


Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
σ_2 = $[t_2 \mapsto Z]\ \sigma_\iota$

σ_3 = $f_{ZA}(\sigma_2, [[x]_1 := [\ldots]_2]_3)$
= $[x \mapsto \sigma_2(t_2)]\ \sigma_2$
= $[x \mapsto Z]\ \sigma_2$
= $[x \mapsto Z, t_2 \mapsto Z]\ \sigma_\iota$

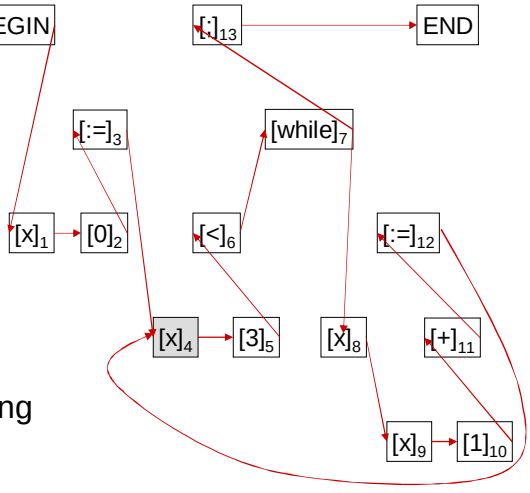
$f_{ZA}(\sigma, [x := [\ldots]_n]_k)$ =
$[x \mapsto \sigma(t_n)]\ \sigma$



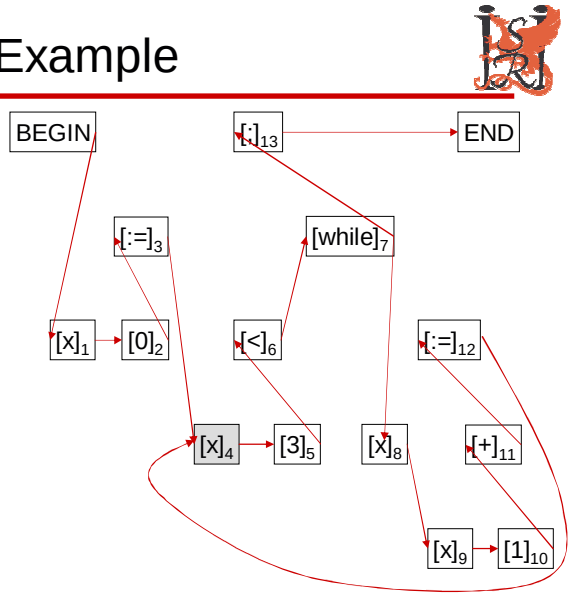
Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
σ_3 = $[x \mapsto Z, t_2 \mapsto Z]\ \sigma_\iota$

Input to $[x]_4$ comes from
$[:=]_3$ and $[:=]_{12}$
Input should be $\sigma_3 \sqcup \sigma_{12}$
What is σ_{12}?
Solution: assume $\perp$
Benefit: $\sigma_3 \sqcup \perp = \sigma_3$
Same result as ignoring back edge first time



Zero Analysis Example

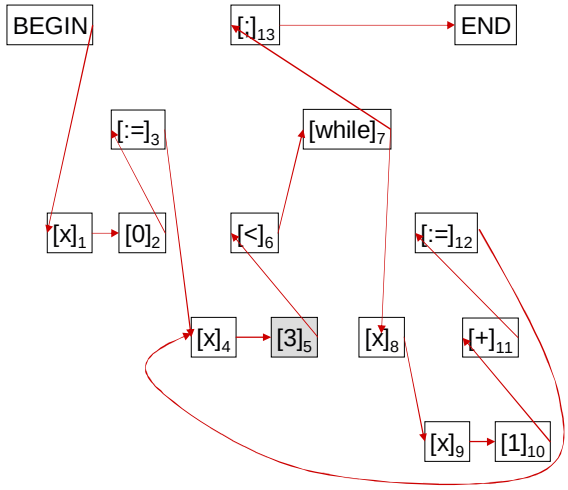


σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
σ_3 = [x$\mapsto Z$, $t_2 \mapsto$ Z] σ_ι
$\sigma_{12} = \bot$

σ_4 = $f_{ZA}(\sigma_3 \sqcup \sigma_{12}, [x]_4)$
= $f_{ZA}(\sigma_3 \sqcup \bot, [x]_4)$
= $f_{ZA}(\sigma_3, [x]_4)$
= $[t_4 \mapsto \sigma_3(x)]\ \sigma_3$
= $[t_4 \mapsto Z]\ \sigma_3$

$f_{ZA}(\sigma, [x]_k) = [t_k \mapsto \sigma(x)]\ \sigma$

Zero Analysis Example



σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
σ_3 = [x$\mapsto Z$, $t_2 \mapsto$ Z] σ_ι
$\sigma_{12} = \bot$
σ_4 = $[t_4 \mapsto Z]\ \sigma_3$

σ_5 = $f_{ZA}(\sigma_4, [3]_5)$
= $[t_5 \mapsto NZ]\ \sigma_4$
= $[t_5 \mapsto NZ, t_4 \mapsto Z]\ \sigma_3$

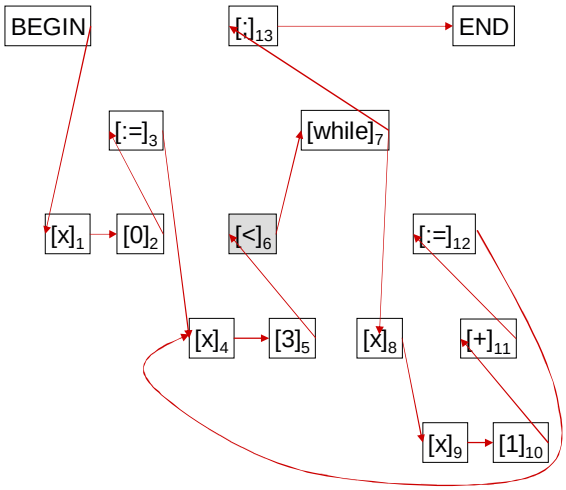
$f_{ZA}(\sigma, [n]_k)$ =
if n==0
then $[t_k \mapsto Z]\sigma$
else $[t_k \mapsto NZ]\sigma$

Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
σ_3 = [x$\mapsto Z$, $t_2 \mapsto$ Z] σ_ι
σ_{12} = $\perp$
σ_5 = [$t_5 \mapsto$ NZ, $t_4 \mapsto$ Z] σ_3

σ_6 = $f_{ZA}(\sigma_5, [<]_6)$
= σ_5
= [$t_5 \mapsto$ NZ, $t_4 \mapsto$ Z] σ_3

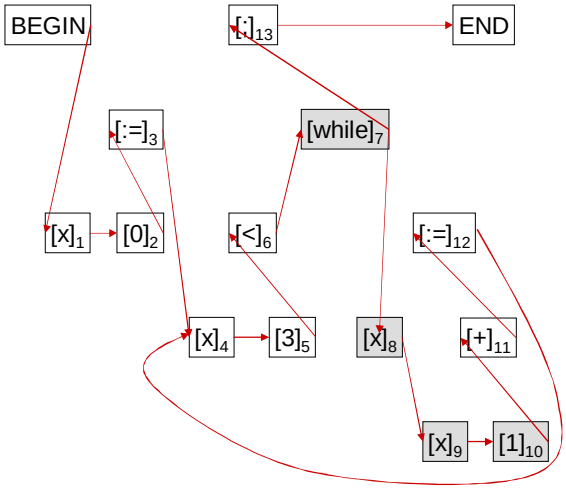
$f_{ZA}(\sigma$, /* *any other* */) = σ



Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | x$\in$ **Var** }
σ_3 = [x$\mapsto Z$, $t_2 \mapsto$ Z] σ_ι
σ_{12} = $\perp$
σ_6 = [$t_5 \mapsto$ NZ, $t_4 \mapsto$ Z] σ_3

Skipping similar nodes…

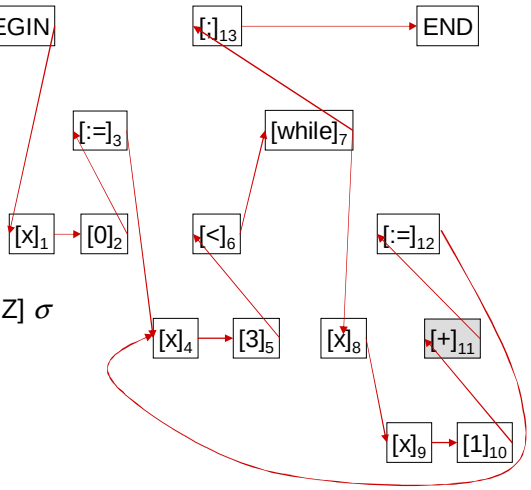


Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | $x \in$ **Var** }
σ_3 = [$x \mapsto Z$, $t_2 \mapsto Z$] σ_ι
σ_{12} = $\perp$
σ_{10} = [$t_9 \mapsto Z$,$t_{10} \mapsto$ NZ,…] σ_3

σ_{11} = $f_{ZA}(\sigma_{10}, [[\ldots]_9 + [\ldots]_{10}]_{11})$
= [$t_{11} \mapsto$ MZ] σ_{10}

$f_{ZA}(\sigma, [[\ldots]_n \; op \; [\ldots]_m]_k) = [t_k \mapsto MZ]\ \sigma$

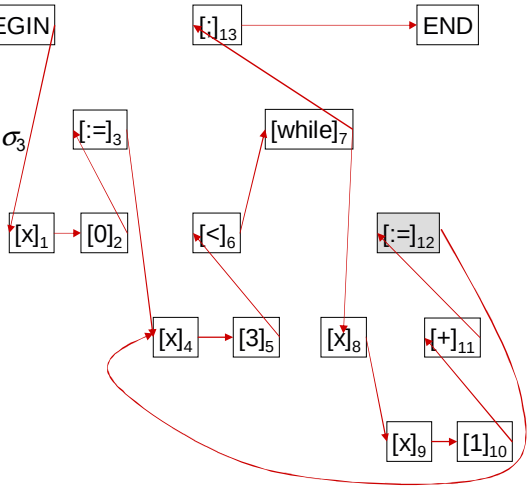


Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | $x \in$ **Var** }
σ_3 = [$x \mapsto Z$, $t_2 \mapsto Z$] σ_ι
σ_{12} = $\perp$
σ_{11} = [$t_9 \mapsto Z$,$t_{10} \mapsto$ NZ,$t_{11} \mapsto$ MZ,…]σ_3

σ_{12} = $f_{ZA}(\sigma_{11}, [[x]_8 := [\ldots]_{11}]_{12})$
= [$x \mapsto \sigma_{11}(t_{11})$] σ_{11}
= [$x \mapsto$ MZ] σ_{11}
= [$x \mapsto$ MZ,…] σ_3

$f_{ZA}(\sigma, [x := [\ldots]_n]_k)$ =
[$x \mapsto \sigma(t_n)$] σ

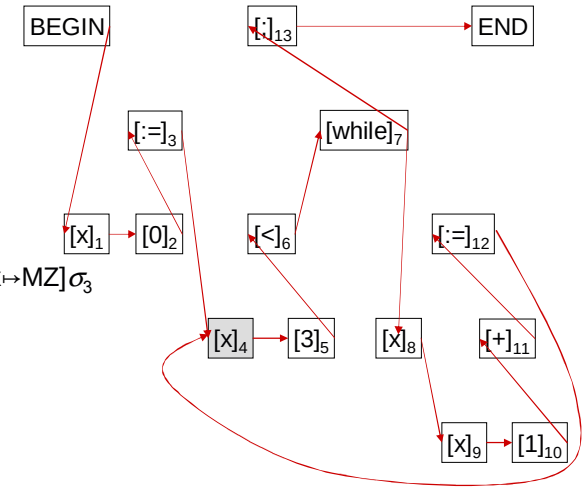


Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | $x \in$ **Var** }
σ_3 = $[x \mapsto Z, t_2 \mapsto Z]\ \sigma_\iota$
σ_{12} = $[x \mapsto MZ, \ldots]\ \sigma_3$

σ_4 = $f_{ZA}(\sigma_3 \sqcup \sigma_{12}, [x]_4)$
= $f_{ZA}([x \mapsto MZ]\sigma_3, [x]_4)$
= $[t_4 \mapsto [x \mapsto MZ]\sigma_3(x)]\ [x \mapsto MZ]\sigma_3$
= $[t_4 \mapsto MZ, x \mapsto MZ]\ \sigma_3$

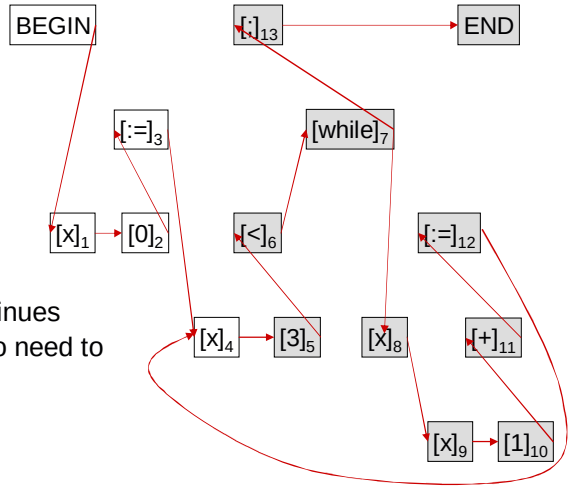
$f_{ZA}(\sigma, [x]_k) = [t_k \mapsto \sigma(x)]\ \sigma$



Zero Analysis Example

σ_ι = { $x \mapsto$ MZ | $x \in$ **Var** }
σ_3 = $[x \mapsto Z, t_2 \mapsto Z]\ \sigma_\iota$
σ_{12} = $[x \mapsto MZ, \ldots]\ \sigma_3$

Propagation of $x \mapsto MZ$ continues
σ_{12} does not change, so no need to iterate again



Worklist Dataflow Analysis Algorithm

```
worklist = new Set();
for all node indexes i do
    results[i] = ⊥_A;
results[entry] = ι_A;
worklist.add(all nodes);

while (!worklist.isEmpty()) do
    i = worklist.pop();
    before = ⊔_{k∈pred(i)} results[k];
    after = f_A(before, node(i));
    if (!(after ⊑ results[i]))
        results[i] = after;
        for all k∈succ(i) do
            worklist.add(k);
```

Ok to just add entry node if flow functions cannot return $\bot_A$ (examples will assume this)

Pop removes the most recently added element from the set (performance optimization)

Example of Worklist

Simplified to the statement level

$[a := 0]_1$

$[b := 0]_2$

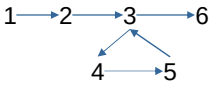
while $[a < 2]_3$ do

  $[b := a]_4$;

  $[a := a + 1]_5$;

$[a := 0]_6$

Control Flow Graph



Position	Worklist	a	b
0	1	MZ	MZ
1	2	Z	MZ
2	3	Z	Z
3	4,6	Z	Z
4	5,6	Z	Z
5	3,6	MZ	Z
3	4,6	MZ	Z
4	5,6	MZ	MZ
5	3,6	MZ	MZ
3	4,6	MZ	MZ
4	6	MZ	MZ
6		Z	MZ

Worklist Algorithm Performance

- Performance
  - Visits node whenever input gets less precise
    - up to h = height of lattice
  - Propagates data along control flow edges
    - up to e = max outbound edges per node
  - Assume lattice operation cost is o
  - Overall, O(h*e*o)
    - Typically h,o,e bounded by n = number of statements in program
    - $O(n^3)$ for many data flow analyses
    - $O(n^2)$ if you assume a number of edges per node is small
  - Good enough to run on a function
    - Usually not run on an entire program at once, because n is too big

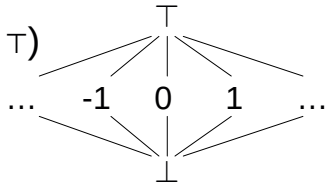
Constant Propagation

- Goal: determine which variables hold a constant value:

```
x := 3;
y := x+7;
if b
    then z := x+2
    else z := y-5;
w := z-2
```

- What is w?
  - Useful for optimization, error checking
  - Zero analysis is a special case

Constant Propagation Definition



- Constant lattice (L_C, $\sqsubseteq_C$, $\sqcup_C$, $\bot$, $\top$)
  - L_C = **Integer** ∪ { $\bot$, $\top$ }
  - $\forall n \in$ **Integer** : $\bot \sqsubseteq_C n$ && $n \sqsubseteq_C \top$
- Constant propagation lattice
  - Tuple lattice formed from above lattice
  - See notes on zero analysis for details
- Abstraction function:
  - $\alpha_C(n) = n$
  - $\alpha_{CP}(\eta) = \{ x \mapsto \alpha_C(\eta(x)) \mid x \in$ **Var** $\}$
- Initial data:
  - $\iota_{CP} = \{ x \mapsto \top \mid x \in$ **Var** $\}$

Constant Propagation Definition

- $f_{CP}(\sigma, [x]_k) = [t_k \mapsto \sigma(x)]\ \sigma$
- $f_{CP}(\sigma, [n]_k) = [t_k \mapsto n]\sigma$
- $f_{CP}(\sigma, [x := [\ldots]_n]_k) = [x \mapsto \sigma(t_n)]\ \sigma$
- $f_{CP}(\sigma, [[\ldots]_n\ op\ [\ldots]_m]_k) = [t_k \mapsto (\sigma(t_n)\ op_\top\ \sigma(t_m))]\ \sigma$
  - $n\ op_\top\ m$ = $n\ op\ m$
  - $n\ op_\top \top = \top$
  - $\top\ op_\top\ m = \top$
  - *Note: we could define for $\bot$ too, but we won't actually ever see $\bot$ during analysis*
- $f_{CP}(\sigma,$ */* any other */*$) = \sigma$

Constant Propagation Example

Simplified to the statement level

$[x := 3]_1$;
$[y := x+7]_2$;
if $[b]_3$
  then $[z := x+2]_4$
  else $[z := y-5]_5$;
$[w := z-2]_6$

Position	Worklist	x	y	z	w
0	1	⊤	⊤	⊤	⊤
1	2	3	⊤	⊤	⊤
2	3	3	10	⊤	⊤
3	4,5	3	10	⊤	⊤
4	6,5	3	10	5	⊤
6	5	3	10	5	3
5	6	3	10	5	⊤
6		3	10	5	3

Constant Propagation Example

Simplified to the statement level

$[x := 3]_1$;
$[y := x+7]_2$;
if $[b]_3$
  then $[z := x+1]_4$
  else $[z := y-5]_5$;
$[w := z-2]_6$

Position	Worklist	x	y	z	w
0	1	⊤	⊤	⊤	⊤
1	2	3	⊤	⊤	⊤
2	3	3	10	⊤	⊤
3	4,5	3	10	⊤	⊤
4	6,5	3	10	4	⊤
6	5	3	10	4	2
5	6	3	10	5	⊤
6		3	10	⊤	⊤