

Lecture 6: Data-Flow Analysis Algorithm: Termination, Complexity, and Fixed Point

17-355/17-665/17-819: Program Analysis

Rohan Padhye

September 16, 2025

* Course materials developed with Jonathan Aldrich and Claire Le Goues

Worklist Algorithm [Kildall'73]

```
worklist =  $\emptyset$ 
for Node n in cfg
    input[n] = output[n] =  $\perp$ 
    add n to worklist
input[0] = initialDataflowInformation

while worklist is not empty
    take a Node n off the worklist
    output[n] = flow(n, input[n])
    for Node j in succs(n)
        newInput = input[j]  $\sqcup$  output[n]
        if newInput  $\neq$  input[j]
            input[j] = newInput
            add j to worklist
```

Ascending Chains & Lattice Height

Ascending Chain

A sequence σ_k is an *ascending chain* iff $n \leq m$ implies $\sigma_n \sqsubseteq \sigma_m$

We can define the height of an ascending chain, and of a lattice, in order to bound the number of new analysis values we can compute at each program point:

Height of an Ascending Chain

An ascending chain σ_k has finite height h if it contains $h + 1$ distinct elements.

Height of a Lattice

A lattice (L, \sqsubseteq) has finite height h if there is an ascending chain in the lattice of height h , and no ascending chain in the lattice has height greater than h

Worklist Algorithm [Kildall'73]

```
worklist =  $\emptyset$   
for Node n in cfg  
    input[n] = output[n] =  $\perp$   
    add n to worklist  
input[0] = initialDataflowInformation
```

```
while worklist is not empty  $O(n * h)$   
    take a Node n off the worklist  
    output[n] = flow(n, input[n])  $O(c)$   
    for Node j in succs(n)  $O(e * h)$  in total (across the while loop)  
        newInput = input[j]  $\sqcup$  output[n]  $O(c)$   
        if newInput  $\neq$  input[j]  
            input[j] = newInput  
            add j to worklist  $O(n * h)$ 
```

$$O(c * n * h) + O(e * n * h) \\ = O(c * (e + n) * h)$$

Worklist Algorithm [Kildall'73]

```
worklist =  $\emptyset$ 
for Node n in cfg
    input[n] = output[n] =  $\perp$ 
    add n to worklist
input[0] = initialDataflowInformation

while worklist is not empty
    take a Node n off the worklist
    output[n] = flow(n, input[n])
    for Node j in succs(n)
        newInput = input[j]  $\sqcup$  output[n]
        if newInput  $\neq$  input[j]
            input[j] = newInput
            add j to worklist
```

Worklist Algorithm [Kam & Ullman'76]

```
worklist =  $\emptyset$ 
for Node n in cfg
    input[n] = output[n] =  $\perp$ 
    add n to worklist
output[programStart] = initialDataflowInformation

while worklist is not empty
    take a Node n off the worklist
    input[n] =  $\sqcup_{k \in \text{preds}(n)} \text{output}[k]$ 
    newOutput = flow(n, input[n])
    if newOutput  $\neq$  output[n]
        output[n] = newOutput
        for Node j in succs(n)
            add j to worklist
```

More obviously computes the fixed point

Recall: Fixed point of Flow Functions

$$(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) \xrightarrow{f_Z} (\sigma'_0, \sigma'_1, \sigma'_2, \dots, \sigma'_n)$$

Fixed point!

$$(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n) = f_Z(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n)$$

Correctness theorem:

If data-flow analysis is well designed*, then any fixed point of the analysis is sound.

* Lattice has finite height and flow functions are monotonic.

$$\sigma'_0 = \sigma_0$$

$$\sigma'_1 = f_Z[\![x := 10]\!](\sigma_0)$$

$$\sigma'_2 = f_Z[\![y := 0]\!](\sigma_1)$$

$$\sigma'_3 = \sigma_2 \sqcup \sigma_7$$

$$\sigma'_4 = f_Z[\![\text{if } x = 10 \text{ goto } 7]\!]_F(\sigma_3)$$

$$\vdots$$

$$\sigma'_8 = f_Z[\![\text{if } x = 10 \text{ goto } 7]\!]_T(\sigma_3)$$

$$\sigma'_9 = f_Z[\![x := y]\!](\sigma_8)$$

Monotonicity of Flow Functions

Monotonicity

A function f is *monotonic* iff $\sigma_1 \sqsubseteq \sigma_2$ implies $f(\sigma_1) \sqsubseteq f(\sigma_2)$

For Zero-Analysis:

Case $f_Z[x := 0](\sigma) = \sigma[x \mapsto Z]$:

Let $f_Z[x := 0](\sigma_1) = \sigma_1[x \mapsto Z] = \sigma'_1$ and $f_Z[x := 0](\sigma_2) = \sigma_2[x \mapsto Z] = \sigma'_2$.

Given $\sigma_1 \sqsubseteq \sigma_2$, we have to show that $\sigma'_1 \sqsubseteq \sigma'_2$.

We know that $\sigma'_1(x) = \sigma'_2(x) = Z$ and that the mappings for all other variables are unchanged.

Since \sqsubseteq is defined pointwise, we therefore have that $\sigma_1[x \mapsto Z] \sqsubseteq \sigma_2[x \mapsto Z]$ or $\sigma'_1 \sqsubseteq \sigma'_2$ as required.

Case $f_Z[x := y](\sigma) = \sigma[x \mapsto \sigma(y)]$: **Exercise!**

Worklist Algorithm [Kam & Ullman'76]

```
worklist =  $\emptyset$ 
for Node n in cfg
    input[n] = output[n] =  $\perp$ 
    add n to worklist
output[programStart] = initialDataflowInformation

while worklist is not empty
    take a Node n off the worklist
    input[n] =  $\sqcup_{k \in \text{preds}(n)} \text{output}[k]$ 
    newOutput = flow(n, input[n])
    if newOutput  $\neq$  output[n]
        output[n] = newOutput
        for Node j in succs(n)
            add j to worklist
```

Can prove termination using induction!
(assuming monotonic flow functions)

Successive applications for the whole-program flow function results in an ascending chain i.e., $\Sigma \sqsubseteq F(\Sigma)$

Base case: $(\perp, \perp, \dots, \perp) \xrightarrow{F} (\sigma'_1, \sigma'_2, \dots, \sigma'_n)$

Inductive case: $(\sigma_1, \sigma_2, \dots, \sigma_n) \xrightarrow{F} (\sigma'_1, \sigma'_2, \dots, \sigma'_n)$

Since the height of the composite lattice of tuples $(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_n)$ is finite, the algorithm terminates! Max number of steps is the height of the composite lattice, which is $n \times$ height of the σ lattice, as before.

Fixed Point

Let $\mathcal{F}(\langle \sigma_1, \sigma_2, \dots, \sigma_{|P|} \rangle) = \langle \sigma'_1, \sigma'_2, \dots, \sigma'_{|P|} \rangle$ be a composite flow function for program P , such that:

$$\begin{aligned} \forall 1 < i \leq |P| : \sigma'_i &= \left(\bigsqcup_{j \in \text{preds}(i)} f[[P[j]]](\sigma_j) \right) \\ \sigma'_1 &= \left(\bigsqcup_{j \in \text{preds}(1)} f[[P[j]]](\sigma_j) \right) \sqcup \sigma_0 \end{aligned}$$

where σ_0 is the initial dataflow information. Then, a dataflow analysis result $\Sigma = \langle \sigma_1, \sigma_2, \dots, \sigma_{|P|} \rangle$ is a *fixed point* iff $\mathcal{F}(\Sigma) = \Sigma$.

Worklist Algorithm Terminates at Fixed Point

At the fixed point, we therefore have the following equations satisfied:

$$\forall i \in P : \left(\bigsqcup_{j \in \text{preds}(i)} f[P[j]](\sigma_j) \right) \sqsubseteq \sigma_i$$

The worklist algorithm shown above computes a fixed point when it terminates. We can prove this by showing that the following loop invariant is maintained:

$$\forall i . (\exists j \in \text{preds}(i) \text{ such that } f[P[j]](\sigma_j) \not\sqsubseteq \sigma_i) \Rightarrow i \in \text{worklist}$$