



# Static Program Analysis

Lecture 4: Dataflow Analysis III (The Framework)

Summer Semester 2018

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ss-18/spa/>

---

## Written Exams

- First: Thu Jul 26, 10:00–13:00 AH 1/2
- Second: Tue Sep 18, 10:00–13:00 AH 2

# Recap: Algebraic Foundations of Dataflow Analysis

---

## Roadmap

**Goal:** solve dataflow equation system by **fixpoint iteration**

1. Characterise solution of equation system as **fixpoint** of a transformation
2. Introduce **partial order** for comparing analysis results
3. Establish **least upper bound** as combination operator
4. Ensure **monotonicity** of transfer functions
5. Guarantee termination of fixpoint iteration by **ascending chain condition**
6. Optimise fixpoint iteration by **worklist algorithm**

# Recap: Algebraic Foundations of Dataflow Analysis

## Complete Lattices

Since  $\{\varphi_{l'}(A_{l'}) \mid (l', l) \in F\}$  can contain arbitrary elements, the existence of least upper bounds must be ensured for arbitrary subsets.

### Definition (Complete lattice)

A **complete lattice** is a partial order  $(D, \sqsubseteq)$  such that all subsets of  $D$  have least upper bounds. In this case,

$$\perp := \bigsqcup \emptyset$$

denotes the **least element** of  $D$ .

### Example

1.  $(\mathbb{N}, \leq)$  is not a complete lattice as, e.g.,  $\mathbb{N}$  does not have a LUB
2. (Live Variables)  $(D, \sqsubseteq) = (2^{\text{Var}_c}, \sqsubseteq)$  is a complete lattice with  $\perp = \emptyset$
3. (Available Expressions)  $(D, \sqsubseteq) = (2^{\text{CExp}_c}, \supseteq)$  is a complete lattice with  $\perp = \text{CExp}_c$

# Recap: Algebraic Foundations of Dataflow Analysis

## The Ascending Chain Condition

Termination of fixpoint iteration is guaranteed by the following condition.

### Definition (Ascending Chain Condition)

- A sequence  $(d_i)_{i \in \mathbb{N}}$  is called an **ascending chain** in  $D$  if  $d_i \sqsubseteq d_{i+1}$  for each  $i \in \mathbb{N}$ .
- A partial order  $(D, \sqsubseteq)$  satisfies the **Ascending Chain Condition (ACC)** if each ascending chain  $d_0 \sqsubseteq d_1 \sqsubseteq \dots$  eventually stabilises, i.e., there exists  $n \in \mathbb{N}$  such that  $d_n = d_{n+1} = \dots$

### Notes:

- The finite height property implies ACC, but not vice versa (as there might be non-stabilising descending chains or stabilising chains of unbounded size – see following slide)
- The complete lattice and ACC properties are orthogonal (see following slide)

### Domain requirements for dataflow analysis

$(D, \sqsubseteq)$  must be a **complete lattice satisfying ACC**

# Recap: Algebraic Foundations of Dataflow Analysis

## Monotonicity of Functions

Monotonicity of transfer functions excludes “oscillating behaviour” in fixpoint iteration.

### Definition (Monotonicity)

Let  $(D, \sqsubseteq)$  and  $(D', \sqsubseteq')$  be partial orders, and let  $\Phi : D \rightarrow D'$ .  $\Phi$  is called **monotonic** (w.r.t.  $(D, \sqsubseteq)$  and  $(D', \sqsubseteq')$ ) if, for every  $d_1, d_2 \in D$ ,

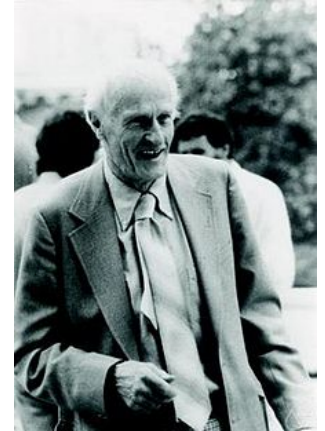
$$d_1 \sqsubseteq d_2 \implies \Phi(d_1) \sqsubseteq' \Phi(d_2).$$

### Example

1. Let  $T := \{S \subseteq \mathbb{N} \mid S \text{ finite}\}$ . Then  $\Phi_1 : T \rightarrow \mathbb{N} : S \mapsto \sum_{n \in S} n$  is monotonic w.r.t.  $(2^{\mathbb{N}}, \subseteq)$  and  $(\mathbb{N}, \leq)$ .
2.  $\Phi_2 : 2^{\mathbb{N}} \rightarrow 2^{\mathbb{N}} : S \mapsto \mathbb{N} \setminus S$  is not monotonic w.r.t.  $(2^{\mathbb{N}}, \subseteq)$  (since, e.g.,  $\emptyset \subseteq \mathbb{N}$  but  $\Phi_2(\emptyset) = \mathbb{N} \not\subseteq \Phi_2(\mathbb{N}) = \emptyset$ ).
3. (Live Variables)  $(D, \sqsubseteq) = (D', \sqsubseteq') = (2^{\text{Var}_c}, \subseteq)$   
Each transfer function  $\varphi_{l'}(V) := (V \setminus \text{kill}_{\text{LV}}(B')) \cup \text{gen}_{\text{LV}}(B')$  is obviously monotonic
4. (Available Expressions)  $(D, \sqsubseteq) = (D', \sqsubseteq') = (2^{\text{CExp}_c}, \supseteq)$  ditto

# Recap: Algebraic Foundations of Dataflow Analysis

## The Fixpoint Theorem



Stephen Kleene (1909–1994)

### Theorem (Fixpoint Theorem by Kleene)

Let  $(D, \sqsubseteq)$  be a complete lattice satisfying ACC and  $\Phi : D \rightarrow D$  monotonic. Then

$$\text{fix}(\Phi) := \bigsqcup \{ \Phi^k(\perp) \mid k \in \mathbb{N} \}$$

is the *least fixpoint* of  $\Phi$  where  $\Phi^0(d) := d$  and  $\Phi^{k+1}(d) := \Phi(\Phi^k(d))$ .

### Function requirements for dataflow analysis

All transfer functions must be a **monotonic**

## Dataflow Systems I

### Definition 4.1 (Dataflow system)

A **dataflow system**  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$  consists of

- a finite set of (program) **labels**  $Lab$  (here:  $Lab_c$ ),
- a set of **extremal labels**  $E \subseteq Lab$  (here:  $\{\text{init}(c)\}$  or  $\{\text{final}(c)\}$ ),
- a **flow relation**  $F \subseteq Lab \times Lab$  (here:  $\text{flow}(c)$  or  $\text{flow}^R(c)$ ),
- a **complete lattice**  $(D, \sqsubseteq)$  satisfying ACC (with LUB operator  $\sqcup$  and least element  $\perp$ ),
- an **extremal value**  $\iota \in D$  (for the extremal labels), and
- a family of **monotonic transfer functions**  $\{\varphi_l \mid l \in Lab\}$  of type  $\varphi_l : D \rightarrow D$ .



# Application to Dataflow Analysis

## Dataflow Systems II

### Example 4.2

Problem	Available Expressions	Live Variables
$E$	$\{\text{init}(c)\}$	$\text{final}(c)$
$F$	$\text{flow}(c)$	$\text{flow}^R(c)$
$D$	$2^{CExp_c}$	$2^{Var_c}$
$\sqsubseteq$	$\supseteq$	$\subseteq$
$\sqcup$	$\cap$	$\cup$
$\perp$	$CExp_c$	$\emptyset$
$\iota$	$\emptyset$	$Var_c$
$\varphi_I$	$\varphi_I(d) = (d \setminus \text{kill}(B^I)) \cup \text{gen}(B^I)$	

# Application to Dataflow Analysis

## Dataflow Systems and Fixpoints

### Definition 4.3 (Dataflow equation system)

Given: dataflow system  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ ,  $Lab = \{1, \dots, n\}$  (w.l.o.g.)

- $S$  determines the **equation system** (where  $l \in Lab$ )

$$A_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{\varphi_{l'}(A_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

- $(d_1, \dots, d_n) \in D^n$  is called a **solution** if

$$d_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{\varphi_{l'}(d_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

- $S$  determines the **transformation**

$$\Phi_S : D^n \rightarrow D^n : (d_1, \dots, d_n) \mapsto (d'_1, \dots, d'_n)$$

where

$$d'_l := \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup \{\varphi_{l'}(d_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

### Corollary 4.4

$(d_1, \dots, d_n) \in D^n$  **solves** the equation system iff it is a **fixpoint** of  $\Phi_S$

## Solving Dataflow Problems by Fixpoint Iteration

- $(D, \sqsubseteq)$  being a **complete lattice** ensures that  $\Phi_S$  is well defined
- Since  $(D, \sqsubseteq)$  is a **complete lattice satisfying ACC**, so is  $(D^n, \sqsubseteq^n)$  (where  $(d_1, \dots, d_n) \sqsubseteq^n (d'_1, \dots, d'_n)$  iff  $d_i \sqsubseteq d'_i$  for every  $1 \leq i \leq n$ )
- Monotonicity of transfer functions  $\varphi_I$  in  $(D, \sqsubseteq)$  implies **monotonicity of  $\Phi_S$**  in  $(D^n, \sqsubseteq^n)$  (since  $\sqcup$  also monotonic)
- Thus the **(least) fixpoint is effectively computable** by iteration:

$$\text{fix}(\Phi_S) = \bigsqcup \{ \Phi_S^k(\perp_{D^n}) \mid k \in \mathbb{N} \}$$

where  $\perp_{D^n} = \underbrace{(\perp_D, \dots, \perp_D)}_{n \text{ times}}$

- If height of  $(D, \sqsubseteq)$  is  $m \in \mathbb{N}$ 
  - $\implies$  height of  $(D^n, \sqsubseteq^n)$  is  $(m - 1) \cdot n + 1$
  - $\implies$  **fixpoint iteration requires at most  $m \cdot n$  steps**

# Application to Dataflow Analysis

## Example: Available Expressions

### Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

```
c = [x := a+b]1;  
    [y := a*b]2;  
    while [y > a+b]3 do  
        [a := a+1]4;  
        [x := a+b]5  
    end
```

Equation system:

$$\begin{aligned} AE_1 &= \emptyset \\ AE_2 &= AE_1 \cup \{a+b\} \\ AE_3 &= (AE_2 \cup \{a*b\}) \cap (AE_5 \cup \{a+b\}) \\ AE_4 &= AE_3 \cup \{a+b\} \\ AE_5 &= AE_4 \setminus \{a+b, a*b, a+1\} \end{aligned}$$

Fixpoint iteration:	$k$	1	2	3	4	5
	0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$
	1	$\emptyset$	$CExp_c$	$CExp_c$	$CExp_c$	$\emptyset$
	2	$\emptyset$	$\{a+b\}$	$\{a+b\}$	$CExp_c$	$\emptyset$
	3	$\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset$
	4	$\emptyset$	$\{a+b\}$	$\{a+b\}$	$\{a+b\}$	$\emptyset$

# Application to Dataflow Analysis

## Example: Live Variables

### Example 4.6 (Live Variables; cf. Example 2.12)

Program:

```
[x := 2]1; [y := 4]2; [x := 1]3;
if [y > 0]4 then
  [z := x]5
else
  [z := y*y]6
end;
[x := z]7
```

Equation system:

$$\begin{aligned} LV_1 &= LV_2 \setminus \{y\} \\ LV_2 &= LV_3 \setminus \{x\} \\ LV_3 &= LV_4 \cup \{y\} \\ LV_4 &= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup ((LV_6 \setminus \{z\}) \cup \{y\}) \\ LV_5 &= (LV_7 \setminus \{x\}) \cup \{z\} \\ LV_6 &= (LV_7 \setminus \{x\}) \cup \{z\} \\ LV_7 &= \{x, y, z\} \end{aligned}$$

Fixpoint iteration:	$k$	1	2	3	4	5	6	7
	0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
	1	$\emptyset$	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{z\}$	$\{z\}$	$\{x, y, z\}$
	2	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{x, y\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\}$
	3	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{x, y\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\}$

# Uniqueness of Solutions

## Uniqueness of Solutions I

**Observation:** (non-minimal) solutions of dataflow equation systems are **not always unique**.

### Example 4.7 (Available Expressions)

$[z := x+y]^1;$   
 $\text{while } [\text{true}]^2 \text{ do}$   
     $[\text{skip}]^3$   
 $\text{end}$

$$\implies AE_1 = \emptyset$$

$$AE_2 = (AE_1 \cup \{x+y\}) \cap AE_3$$

$$AE_3 = AE_2$$

$$\implies AE_1 = \emptyset$$

$$AE_2 = \{x+y\} \cap AE_3$$

$$AE_3 = AE_2$$

$\implies$  **Solutions:**  $AE_1 = AE_2 = AE_3 = \emptyset$  or  
 $AE_1 = \emptyset, AE_2 = AE_3 = \{x+y\}$

Here: **greatest** solution (w.r.t.  $\subseteq$ )  $\{x+y\}$  (maximal potential for optimisation)

# Uniqueness of Solutions

## Uniqueness of Solutions II

### Example 4.8 (Live Variables)

while $[x > 1]^1$ do	$\implies$	$LV_1 = LV_2 \cup (LV_3 \cup \{x\})$
$[skip]^2$		$LV_2 = LV_1 \cup \{x\}$
end;		$LV_3 = LV_4 \setminus \{y\}$
$[x := x+1]^3$ ;		$LV_4 = \{x, y\}$
$[y := 0]^4$	$\implies$	$LV_3 = \{x\}$
	$\implies$	$LV_1 = LV_2 \cup \{x\}$
		$= LV_1 \cup \{x\}$

$\implies$  **Solutions:**  $LV_1 = LV_2 = (\{x\} \text{ or } \{x, y\})$ ,  
 $LV_3 = \{x\}, LV_4 = \{x, y\}$

Here: **least** solution  $\{x\}$  (maximal potential for optimisation)

# Efficient Fixpoint Computation

## A Worklist Algorithm I

**Observation:** fixpoint iteration re-computes every  $AI_I$  in every step

⇒ **redundant** if  $AI_{I'}$  at no  $F$ -predecessor  $I'$  changed

⇒ optimisation by **worklist** over control-flow edges

### Algorithm 4.9 (Worklist algorithm)

```
Input: dataflow system  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$   
Variables:  $W \in (Lab \times Lab)^*$ ,  $\{AI_I \in D \mid I \in Lab\}$   
Procedure:  $W := \varepsilon$ ; for  $(I, I') \in F$  do  $W := W \cdot (I, I')$ ;           % Initialise  $W$   
  for  $I \in Lab$  do  
    if  $I \in E$  then  $AI_I := \iota$  else  $AI_I := \perp_D$ ;           % Initialise  $AI$   
  while  $W \neq \varepsilon$  do  
     $(I, I') := \text{head}(W)$ ;  $W := \text{tail}(W)$ ;           % Next control-flow edge  
    if  $\varphi_I(AI_I) \not\sqsubseteq AI_{I'}$  then           % Fixpoint not yet reached  
       $AI_{I'} := AI_{I'} \sqcup \varphi_I(AI_I)$ ;           % Update analysis information  
      for  $(I', I'') \in F$  do  
        if  $(I', I'')$  not in  $W$  then  $W := (I', I'') \cdot W$ ; % Propagate modification  
Output:  $\{AI_I \mid I \in Lab\}$ 
```



# Efficient Fixpoint Computation

## A Worklist Algorithm II

### Example 4.10 (Worklist algorithm)

Available Expression analysis for  $c = [x := a+b]^1;$   
(cf. Examples 2.9 and 4.5)  $[y := a*b]^2;$   
 $\text{while } [y > a+b]^3 \text{ do}$   
     $[a := a+1]^4;$   
     $[x := a+b]^5$   
 $\text{end}$

Transfer functions:  $\varphi_1(A) = A \cup \{a+b\}$   
 $\varphi_2(A) = A \cup \{a*b\}$   
 $\varphi_3(A) = A \cup \{a+b\}$   
 $\varphi_4(A) = A \setminus \{a+b, a*b, a+1\}$   
 $\varphi_5(A) = A \cup \{a+b\}$

Computation protocol: on the board

# Efficient Fixpoint Computation

## An “Optimisation”

**Conjecture:** it suffices to initialise worklist with **edges leaving extremal labels** (such that analysis information will propagate through CFG)

**But ...**

### Example 4.11 (Counterexample)

Live Variables analysis for  $c = [x := 0]^1;$   
 $[x := x + 1]^2;$   
 $[x := 2]^3$

Solution:  $LV_1 = \{x\}, LV_2 = \emptyset, LV_3 = \{x\}$

“Optimised” worklist algorithm:

$W$	$LV_1$	$LV_2$	$LV_3$
$(3, 2)$	$\emptyset$	$\emptyset$	$\{x\}$
$\varepsilon$	$\emptyset$	$\emptyset$	$\{x\}$

$\Rightarrow$  **wrong result!**

# Efficient Fixpoint Computation

---

## Correctness of Worklist Algorithm

### Theorem 4.12 (Correctness of worklist algorithm)

*Given a dataflow system  $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ , Algorithm 4.9 always terminates and computes  $\text{fix}(\Phi_S)$ .*

### Proof.

see [Nielson/Nielson/Hankin 2005, p. 75 ff] □