



Static Program Analysis

Lecture 4: Dataflow Analysis III (The Framework)

Winter Semester 2016/17

Thomas Noll

Software Modeling and Verification Group

RWTH Aachen University

<https://moves.rwth-aachen.de/teaching/ws-1617/spa/>

Organisational Matters

- **Written exams**
 - first: Tue 21 Feb, 15:00 - 17:00, AH 2/3
 - second: Thu 23 Mar, 10:00 – 12:30, AH 2
- **Lecture Thu Nov 3** will take place!

Recap: Heading for a Dataflow Analysis Framework

Outline of Lecture 4

Recap: Heading for a Dataflow Analysis Framework

Order-Theoretic Foundations: The Function

Application to Dataflow Analysis

Uniqueness of Solutions

Recap: Heading for a Dataflow Analysis Framework

Similarities Between Analysis Problems

- **Observation:** the analyses presented so far have some **similarities**

⇒ Look for underlying **framework**

- **Advantages:**

- possibility for designing (efficient) **generic algorithms** for solving dataflow equations
- enables generic **correctness proofs** of analyses and algorithms

- **Overall pattern:** for $c \in \text{Cmd}$ and $l \in \text{Lab}_c$, the **analysis information (AI)** is described by **equations** of the form

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

where

- the set of **extremal labels**, E , is $\{\text{init}(c)\}$ or $\{\text{final}(c)\}$
- ι specifies the **extremal analysis information**
- the **combination operator**, \bigsqcup , is \cap or \cup
- $\varphi_{l'}$ denotes the **transfer function** of block $B_{l'}$
- the **flow relation** F is $\text{flow}(c)$ or $\text{flow}^R(c)$ ($:= \{(l', l) \mid (l, l') \in \text{flow}(c)\}$)

Recap: Heading for a Dataflow Analysis Framework

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: Heading for a Dataflow Analysis Framework

Partial Orders

The domain of analysis information usually forms a partial order where the ordering relation compares the “precision” of information.

Definition (Partial order)

A **partial order (PO)** (D, \sqsubseteq) consists of a set D , called **domain**, and of a relation $\sqsubseteq \subseteq D \times D$ such that, for every $d_1, d_2, d_3 \in D$,

reflexivity: $d_1 \sqsubseteq d_1$

transitivity: $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_3 \implies d_1 \sqsubseteq d_3$

antisymmetry: $d_1 \sqsubseteq d_2$ and $d_2 \sqsubseteq d_1 \implies d_1 = d_2$

It is called **total** if, in addition, always $d_1 \sqsubseteq d_2$ or $d_2 \sqsubseteq d_1$.

Example

1. (\mathbb{N}, \leq) is a total partial order
2. $(\mathbb{N}, <)$ is not a partial order (since not reflexive)
3. (Live Variables) $(2^{\text{Var}_c}, \sqsubseteq)$ is a (non-total) partial order
4. (Available Expressions) $(2^{\text{CExp}_c}, \supseteq)$ is a (non-total) partial order

Recap: Heading for a Dataflow Analysis Framework

Upper Bounds

In the dataflow equation system, analysis information from several predecessors is combined by taking the least upper bound.

Definition ((Least) upper bound)

Let (D, \sqsubseteq) be a partial order and $S \subseteq D$.

1. An element $d \in D$ is called an **upper bound** of S if $s \sqsubseteq d$ for every $s \in S$ (notation: $S \sqsubseteq d$).
2. An upper bound d of S is called **least upper bound (LUB)** or **supremum** of S if $d \sqsubseteq d'$ for every upper bound d' of S (notation: $d = \bigsqcup S$).

Example

1. $S \subseteq \mathbb{N}$ has a LUB in (\mathbb{N}, \leq) iff it is finite
2. (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$. Given $V_1, \dots, V_n \subseteq Var_c$,
$$\bigsqcup \{V_1, \dots, V_n\} = \bigcup \{V_1, \dots, V_n\}$$
3. (Available Expressions) $(D, \sqsubseteq) = (2^{CExp_c}, \supseteq)$. Given $A_1, \dots, A_n \subseteq CExp_c$,
$$\bigsqcup \{A_1, \dots, A_n\} = \bigcap \{A_1, \dots, A_n\}$$

Recap: Heading for a Dataflow Analysis Framework

Complete Lattices

Since $\{\varphi_{l'}(AI_{l'}) \mid (l', l) \in F\}$ 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) $(D, \sqsubseteq) = (2^{\text{Var}_c}, \subseteq)$ 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: Heading for a Dataflow Analysis Framework

Chains

Chains are generated by the approximation of the analysis information in the fixpoint iteration.

Definition (Chain)

Let (D, \sqsubseteq) be a partial order.

- A subset $S \subseteq D$ is called a **chain** in D if, for every $d_1, d_2 \in S$,
$$d_1 \sqsubseteq d_2 \text{ or } d_2 \sqsubseteq d_1$$
(that is, S is a totally ordered subset of D).
- (D, \sqsubseteq) has **finite height** if all chains are finite. In this case, its **height** is
$$\max\{|S| \mid S \text{ chain in } D\} - 1.$$

Example

1. Every $S \subseteq \mathbb{N}$ is a chain in (\mathbb{N}, \leq) (which is of infinite height)
2. $\{\emptyset, \{0\}, \{0, 1\}, \{0, 1, 2\}, \dots\}$ is a chain in $(2^{\mathbb{N}}, \subseteq)$
3. $\{\{0\}, \{1\}, \{2\}\}$ is not a chain in $(2^{\mathbb{N}}, \subseteq)$

Recap: Heading for a Dataflow Analysis Framework

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 – see next slide)
- The complete lattice and ACC properties are orthogonal (see next slide)

Order-Theoretic Foundations: The Function

Outline of Lecture 4

Recap: Heading for a Dataflow Analysis Framework

Order-Theoretic Foundations: The Function

Application to Dataflow Analysis

Uniqueness of Solutions

Monotonicity of Functions

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

Definition 4.1 (Monotonicity)

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

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

Monotonicity of Functions

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

Definition 4.1 (Monotonicity)

Let (D, \sqsubseteq) and (D', \sqsubseteq') be partial orders, and let $\Phi : D \rightarrow D'$. Φ 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 4.2

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) .

Monotonicity of Functions

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

Definition 4.1 (Monotonicity)

Let (D, \sqsubseteq) and (D', \sqsubseteq') be partial orders, and let $\Phi : D \rightarrow D'$. Φ 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 4.2

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 $\Phi_2(\emptyset) = \mathbb{N} \not\subseteq \Phi_2(\mathbb{N}) = \emptyset$).

Monotonicity of Functions

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

Definition 4.1 (Monotonicity)

Let (D, \sqsubseteq) and (D', \sqsubseteq') be partial orders, and let $\Phi : D \rightarrow D'$. Φ 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 4.2

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 $\varphi_{l'}(V) := (V \setminus \text{kill}_{LV}(B')) \cup \text{gen}_{LV}(B')$ is obviously monotonic

Monotonicity of Functions

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

Definition 4.1 (Monotonicity)

Let (D, \sqsubseteq) and (D', \sqsubseteq') be partial orders, and let $\Phi : D \rightarrow D'$. Φ 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 4.2

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 $\varphi_{l'}(V) := (V \setminus \text{kill}_{LV}(B')) \cup \text{gen}_{LV}(B')$ is obviously monotonic
4. (Available Expressions) $(D, \sqsubseteq) = (D', \sqsubseteq') = (2^{\text{CExp}_c}, \supseteq)$ ditto

Fixpoints

Definition 4.3 (Fixpoint)

Let D be some domain, $d \in D$, and $\Phi : D \rightarrow D$. If

$$\Phi(d) = d$$

then d is called a **fixpoint** of Φ .

Order-Theoretic Foundations: The Function

Fixpoints

Definition 4.3 (Fixpoint)

Let D be some domain, $d \in D$, and $\Phi : D \rightarrow D$. If

$$\Phi(d) = d$$

then d is called a **fixpoint** of Φ .

Example 4.4

The (only) fixpoints of $\Phi : \mathbb{N} \rightarrow \mathbb{N} : n \mapsto n^2$ are 0 and 1

The Fixpoint Theorem I



Alfred Tarski (1901–1983)



Bronislaw Knaster (1893–1990)

Theorem 4.5 (Fixpoint Theorem by Tarski and Knaster)

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 Φ where $\Phi^0(d) := d$ and $\Phi^{k+1}(d) := \Phi(\Phi^k(d))$.

Order-Theoretic Foundations: The Function

The Fixpoint Theorem I



Alfred Tarski (1901–1983)



Bronislaw Knaster (1893–1990)

Theorem 4.5 (Fixpoint Theorem by Tarski and Knaster)

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 Φ 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**

The Fixpoint Theorem II

The proof of Theorem 4.5 requires the following lemma.

Lemma 4.6

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

$$\Phi(\bigsqcup S) = \bigsqcup \Phi(S)$$

Order-Theoretic Foundations: The Function

The Fixpoint Theorem II

The proof of Theorem 4.5 requires the following lemma.

Lemma 4.6

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

$$\Phi(\bigsqcup S) = \bigsqcup \Phi(S)$$

Proof (Lemma 4.6).

on the board



Order-Theoretic Foundations: The Function

The Fixpoint Theorem II

The proof of Theorem 4.5 requires the following lemma.

Lemma 4.6

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

$$\Phi(\bigsqcup S) = \bigsqcup \Phi(S)$$

Proof (Lemma 4.6).

on the board

Proof (Theorem 4.5).

on the board

Order-Theoretic Foundations: The Function

The Fixpoint Theorem II

The proof of Theorem 4.5 requires the following lemma.

Lemma 4.6

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

$$\Phi(\bigsqcup S) = \bigsqcup \Phi(S)$$

Proof (Lemma 4.6).

on the board

Proof (Theorem 4.5).

on the board

Remark: $(\Phi^k(\perp))_{k \in \mathbb{N}}$ is obviously an ascending chain which (by ACC) stabilises at some $k_0 \in \mathbb{N}$ with $\text{fix}(\Phi) = \Phi^{k_0}(\perp)$ (where k_0 bounded by height of (D, \sqsubseteq))

Application to Dataflow Analysis

Outline of Lecture 4

Recap: Heading for a Dataflow Analysis Framework

Order-Theoretic Foundations: The Function

Application to Dataflow Analysis

Uniqueness of Solutions

Dataflow Systems I

Definition 4.7 (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 collection 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 4.8

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
ι	\emptyset	Var_c
φ_1	$\varphi_1(d) = (d \setminus \text{kill}(B^l)) \cup \text{gen}(B^l)$	

Application to Dataflow Analysis

Dataflow Systems and Fixpoints

Definition 4.9 (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 $l \in Lab$)

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

Application to Dataflow Analysis

Dataflow Systems and Fixpoints

Definition 4.9 (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

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

Application to Dataflow Analysis

Dataflow Systems and Fixpoints

Definition 4.9 (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

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

Application to Dataflow Analysis

Dataflow Systems and Fixpoints

Definition 4.9 (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 $l \in Lab$)

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

- $(d_1, \dots, d_n) \in D^n$ 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 4.10

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

Solving Dataflow Problems by Fixpoint Iteration

Remarks:

- (D, \sqsubseteq) being a **complete lattice** ensures that Φ_S is well defined

Solving Dataflow Problems by Fixpoint Iteration

Remarks:

- (D, \sqsubseteq) being a **complete lattice** ensures that Φ_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 $1 \leq i \leq n$)

Solving Dataflow Problems by Fixpoint Iteration

Remarks:

- (D, \sqsubseteq) being a **complete lattice** ensures that Φ_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 φ_i in (D, \sqsubseteq) implies **monotonicity of Φ_S** in (D^n, \sqsubseteq^n) (since \sqcup also monotonic)

Solving Dataflow Problems by Fixpoint Iteration

Remarks:

- (D, \sqsubseteq) being a **complete lattice** ensures that Φ_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 φ_i in (D, \sqsubseteq) implies **monotonicity of Φ_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 $\perp_{D^n} = \underbrace{(\perp_D, \dots, \perp_D)}_{n \text{ times}}$

Solving Dataflow Problems by Fixpoint Iteration

Remarks:

- (D, \sqsubseteq) being a **complete lattice** ensures that Φ_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 φ_i in (D, \sqsubseteq) implies **monotonicity of Φ_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
 - \implies height of (D^n, \sqsubseteq^n) is $m \cdot n$
 - \implies **fixpoint iteration requires at most $m \cdot n$ steps**

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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
```

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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:

```
AE1 = ∅  
AE2 = AE1 ∪ {a+b}  
AE3 = (AE2 ∪ {a*b}) ∩ (AE5 ∪ {a+b})  
AE4 = AE3 ∪ {a+b}  
AE5 = AE4 \ {a+b, a*b, a+1}
```

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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:

i	1	2	3	4	5
0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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:

i	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

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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:

<i>i</i>	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

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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:

i	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

Application to Dataflow Analysis

Example: Available Expressions

Example 4.11 (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:

<i>i</i>	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 4.12 (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
```

Application to Dataflow Analysis

Example: Live Variables

Example 4.12 (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:

```
LV1 = LV2 \ {y}
LV2 = LV3 \ {x}
LV3 = LV4 ∪ {y}
LV4 = ((LV5 \ {z}) ∪ {x}) ∪ ((LV6 \ {z}) ∪ {y})
LV5 = (LV7 \ {x}) ∪ {z}
LV6 = (LV7 \ {x}) ∪ {z}
LV7 = {x, y, z}
```

Application to Dataflow Analysis

Example: Live Variables

Example 4.12 (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:

```
LV1 = LV2 \ {y}
LV2 = LV3 \ {x}
LV3 = LV4 ∪ {y}
LV4 = ((LV5 \ {z}) ∪ {x}) ∪ ((LV6 \ {z}) ∪ {y})
LV5 = (LV7 \ {x}) ∪ {z}
LV6 = (LV7 \ {x}) ∪ {z}
LV7 = {x, y, z}
```

Fixpoint iteration:

<i>i</i>	1	2	3	4	5	6	7
0	∅	∅	∅	∅	∅	∅	∅

Application to Dataflow Analysis

Example: Live Variables

Example 4.12 (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:

i	1	2	3	4	5	6	7
0	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset	\emptyset
1	\emptyset	\emptyset	$\{y\}$	$\{x, y\}$	$\{z\}$	$\{z\}$	$\{x, y, z\}$

Application to Dataflow Analysis

Example: Live Variables

Example 4.12 (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:

```
LV1 = LV2 \ {y}
LV2 = LV3 \ {x}
LV3 = LV4 ∪ {y}
LV4 = ((LV5 \ {z}) ∪ {x}) ∪ ((LV6 \ {z}) ∪ {y})
LV5 = (LV7 \ {x}) ∪ {z}
LV6 = (LV7 \ {x}) ∪ {z}
LV7 = {x, y, z}
```

Fixpoint iteration:

<i>i</i>	1	2	3	4	5	6	7
0	∅	∅	∅	∅	∅	∅	∅
1	∅	∅	{y}	{x, y}	{z}	{z}	{x, y, z}
2	∅	{y}	{x, y}	{x, y}	{y, z}	{y, z}	{x, y, z}

Application to Dataflow Analysis

Example: Live Variables

Example 4.12 (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:

```
LV1 = LV2 \ {y}
LV2 = LV3 \ {x}
LV3 = LV4 ∪ {y}
LV4 = ((LV5 \ {z}) ∪ {x}) ∪ ((LV6 \ {z}) ∪ {y})
LV5 = (LV7 \ {x}) ∪ {z}
LV6 = (LV7 \ {x}) ∪ {z}
LV7 = {x, y, z}
```

Fixpoint iteration:

<i>i</i>	1	2	3	4	5	6	7
0	∅	∅	∅	∅	∅	∅	∅
1	∅	∅	{y}	{x, y}	{z}	{z}	{x, y, z}
2	∅	{y}	{x, y}	{x, y}	{y, z}	{y, z}	{x, y, z}
3	∅	{y}	{x, y}	{x, y}	{y, z}	{y, z}	{x, y, z}

Uniqueness of Solutions

Outline of Lecture 4

Recap: Heading for a Dataflow Analysis Framework

Order-Theoretic Foundations: The Function

Application to Dataflow Analysis

Uniqueness of Solutions

Uniqueness of Solutions

Uniqueness of Solutions I

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

Uniqueness of Solutions

Uniqueness of Solutions I

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

Example 4.13 (Available Expressions)

```
[z := x+y]1;  
while [true]2 do  
  [skip]3  
end
```

Uniqueness of Solutions

Uniqueness of Solutions I

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

Example 4.13 (Available Expressions)

```
[z := x+y]1;  
while [true]2 do  
  [skip]3  
end
```

$$\begin{aligned} \implies AE_1 &= \emptyset \\ AE_2 &= (AE_1 \cup \{x+y\}) \cap AE_3 \\ AE_3 &= AE_2 \end{aligned}$$

Uniqueness of Solutions

Uniqueness of Solutions I

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

Example 4.13 (Available Expressions)

```
[z := x+y]1;  
while [true]2 do  
  [skip]3  
end
```

$$\begin{aligned} \implies & AE_1 = \emptyset \\ & AE_2 = (AE_1 \cup \{x+y\}) \cap AE_3 \\ & AE_3 = AE_2 \\ \implies & AE_1 = \emptyset \\ & AE_2 = \{x+y\} \cap AE_3 \\ & AE_3 = AE_2 \end{aligned}$$

Uniqueness of Solutions

Uniqueness of Solutions I

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

Example 4.13 (Available Expressions)

$[z := x+y]^1;$	$\implies AE_1 = \emptyset$
$\text{while } [true]^2 \text{ do}$	$AE_2 = (AE_1 \cup \{x+y\}) \cap AE_3$
$[skip]^3$	$AE_3 = AE_2$
end	$\implies AE_1 = \emptyset$
	$AE_2 = \{x+y\} \cap AE_3$
	$AE_3 = AE_2$
\implies Solutions:	$AE_1 = AE_2 = AE_3 = \emptyset$ or
	$AE_1 = \emptyset, AE_2 = AE_3 = \{x+y\}$

Uniqueness of Solutions

Uniqueness of Solutions I

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

Example 4.13 (Available Expressions)

$[z := x+y]^1;$	$\implies AE_1 = \emptyset$
$\text{while } [true]^2 \text{ do}$	$AE_2 = (AE_1 \cup \{x+y\}) \cap AE_3$
$[skip]^3$	$AE_3 = AE_2$
end	$\implies AE_1 = \emptyset$
	$AE_2 = \{x+y\} \cap AE_3$
	$AE_3 = AE_2$

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

Here: **greatest** solution $\{x+y\}$ (maximal potential for optimisation)

Uniqueness of Solutions

Uniqueness of Solutions II

Example 4.14 (Live Variables)

```
while [x>1]1 do
  [skip]2
end;
[x := x+1]3;
[y := 0]4
```

Uniqueness of Solutions

Uniqueness of Solutions II

Example 4.14 (Live Variables)

```
while [x>1]1 do
  [skip]2
end;
[x := x+1]3;
[y := 0]4
```

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

Uniqueness of Solutions

Uniqueness of Solutions II

Example 4.14 (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	$LV_3 = \{x\}$

Uniqueness of Solutions

Uniqueness of Solutions II

Example 4.14 (Live Variables)

```
while [x>1]1 do
  [skip]2
end;
[x := x+1]3;
[y := 0]4
```

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

Uniqueness of Solutions

Uniqueness of Solutions II

Example 4.14 (Live Variables)

```
while [x>1]1 do
  [skip]2
end;
[x := x+1]3;
[y := 0]4
```

$$\begin{aligned} &\implies LV_1 = LV_2 \cup (LV_3 \cup \{x\}) \\ &\quad LV_2 = LV_1 \cup \{x\} \\ &\quad LV_3 = LV_4 \setminus \{y\} \\ &\quad LV_4 = \{x, y\} \\ &\implies LV_3 = \{x\} \\ &\implies LV_1 = LV_2 \cup \{x\} \\ &\quad = LV_1 \cup \{x\} \end{aligned}$$

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

Uniqueness of Solutions

Uniqueness of Solutions II

Example 4.14 (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)