

Semantics and Verification of Software

Lecture 16: Dataflow Analysis III (The Framework)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

`noll@cs.rwth-aachen.de`

`http://www-i2.informatik.rwth-aachen.de/i2/svsw08/`

Winter semester 2008/09

- 1 Repetition: Heading for a Dataflow Analysis Framework
- 2 Order-Theoretic Foundations
- 3 The Framework

- For each $l \in L_c$, $AE_l \subseteq AExp_c$ represents the **set of available expressions at the entry of block B^l**

- Formally, for $c \in Cmd$ with isolated entry:

$$AE_l = \begin{cases} \emptyset & \text{if } l = \text{init}(c) \\ \bigcap \{ \varphi_{l'}(AE_{l'}) \mid (l', l) \in \text{flow}(c) \} & \text{otherwise} \end{cases}$$

where $\varphi_{l'} : 2^{AExp_c} \rightarrow 2^{AExp_c}$ denotes the **transfer function** of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \text{kill}_{AE}(B^{l'})) \cup \text{gen}_{AE}(B^{l'})$$

- Characterization of analysis:

forward: starts in $\text{init}(c)$ and proceeds downwards

must: $\bigcap$ in equation for AE_l

flow-sensitive: results depending on order of assignments

- Later: solution **not necessarily unique**
 $\implies$ choose **greatest one**

- For each $l \in L_c$, $LV_l \subseteq Var_c$ represents the set of **live variables at the exit of block B^l**
- Formally, for a program $c \in Cmd$ with isolated exits:

$$LV_l = \begin{cases} Var_c & \text{if } l \in \text{final}(c) \\ \bigcup \{ \varphi_{l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise} \end{cases}$$

where $\varphi_{l'} : 2^{Var_c} \rightarrow 2^{Var_c}$ denotes the **transfer function** of block $B^{l'}$, given by

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

- Characterization of analysis:
 - backward**: starts in $\text{final}(c)$ and proceeds upwards
 - may**: $\bigcup$ in equation for LV_l
 - flow-sensitive**: results depending on order of assignments
- Later: solution **not necessarily unique**
 $\implies$ choose **least one**

Similarities between Analysis Problems

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

⇒ Look for underlying **framework**

- **Advantage:** possibility for designing (efficient) **generic algorithms for solving dataflow equations**
- **Overall pattern:** for $c \in Cmd$ and $l \in L_c$, 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

- ι specifies the initial analysis information
- E is $\{\text{init}(c)\}$ or $\{\text{final}(c)\}$
- $\bigsqcup$ is $\bigcap$ or $\bigcup$
- $\varphi_{l'}$ denotes the transfer function of block $B^{l'}$
- F is $\text{flow}(c)$ or $\text{flow}^R(c)$ ($:= \{(l', l) \mid (l, l') \in \text{flow}(c)\}$)

- **Direction of information flow:**

- **forward:**

- $F = \text{flow}(c)$
 - Al_l concerns entry of B^l
 - c has isolated entry

- **backward:**

- $F = \text{flow}^R(c)$
 - Al_l concerns exit of B^l
 - c has isolated exits

- **Quantification over paths:**

- **may:**

- $\sqcup = \bigcup$
 - property satisfied by some path
 - interested in least solution (later)

- **must:**

- $\sqcap = \bigcap$
 - property satisfied by all paths
 - interested in greatest solution (later)

- 1 Repetition: Heading for a Dataflow Analysis Framework
- 2 Order-Theoretic Foundations
- 3 The Framework

Partial Orders

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

Definition 16.1 (Partial order; repetition of Def. 6.1)

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 $d_2 \sqsubseteq d_1$.

Partial Orders

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

Definition 16.1 (Partial order; repetition of Def. 6.1)

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 16.2

① (Available Expressions) $(2^{AExp_c}, \supseteq)$ is a (non-total) partial order

Partial Orders

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

Definition 16.1 (Partial order; repetition of Def. 6.1)

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 16.2

- ➊ (Available Expressions) $(2^{AExp_c}, \supseteq)$ is a (non-total) partial order
- ➋ (Live Variables) $(2^{Var_c}, \subseteq)$ is a (non-total) partial order

Definition 16.3 (Upper and lower bound)

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

- 1 An element $d \in D$ is called an **upper/lower bound** of S if $s \sqsubseteq d/d \sqsubseteq s$ for every $s \in S$ (notation: $S \sqsubseteq d/d \sqsubseteq S$).

Definition 16.3 (Upper and lower bound)

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

- 1 An element $d \in D$ is called an **upper/lower bound** of S if $s \sqsubseteq d/d \sqsubseteq s$ for every $s \in S$ (notation: $S \sqsubseteq d/d \sqsubseteq S$).
- 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: $d = \bigsqcup S$).

Definition 16.3 (Upper and lower bound)

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

- 1 An element $d \in D$ is called an **upper/lower bound** of S if $s \sqsubseteq d/d \sqsubseteq s$ for every $s \in S$ (notation: $S \sqsubseteq d/d \sqsubseteq S$).
- 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: $d = \bigsqcup S$).
- 3 A lower bound d of S is called **greatest lower bound (GLB)** or **infimum** of S if $d' \sqsubseteq d$ for every lower bound d' of S (notation: $d = \bigsqcap S$).

Example 16.4

- ① (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$

Given $A_1, \dots, A_n \subseteq AExp_c$,

$$\bigsqcup\{A_1, \dots, A_n\} = \bigcap\{A_1, \dots, A_n\} \text{ and} \\ \bigsqcap\{A_1, \dots, A_n\} = \bigcup\{A_1, \dots, A_n\}$$

Example 16.4

- ❶ (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$

Given $A_1, \dots, A_n \subseteq AExp_c$,

$$\bigsqcup\{A_1, \dots, A_n\} = \bigcap\{A_1, \dots, A_n\} \text{ and} \\ \bigsqcap\{A_1, \dots, A_n\} = \bigcup\{A_1, \dots, A_n\}$$

- ❷ (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$

Given $V_1, \dots, V_n \subseteq Var_c$,

$$\bigsqcup\{V_1, \dots, V_n\} = \bigcup\{V_1, \dots, V_n\} \text{ and} \\ \bigsqcap\{V_1, \dots, V_n\} = \bigcap\{V_1, \dots, V_n\}$$

Definition 16.5 (Complete lattice)

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

$$\begin{aligned}\perp &:= \bigsqcup \emptyset = \bigsqcap D \text{ and} \\ \top &:= \bigsqcap \emptyset = \bigsqcup D\end{aligned}$$

denote the **least** and the **greatest element** of D , respectively.

Definition 16.5 (Complete lattice)

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

$$\begin{aligned}\perp &:= \bigsqcup \emptyset = \bigsqcap D \text{ and} \\ \top &:= \bigsqcap \emptyset = \bigsqcup D\end{aligned}$$

denote the **least** and the **greatest element** of D , respectively.

Example 16.6

- ① (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$ is a complete lattice with $\perp = AExp_c$ and $\top = \emptyset$

Definition 16.5 (Complete lattice)

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

$$\begin{aligned}\perp &:= \bigsqcup \emptyset = \bigsqcap D \text{ and} \\ \top &:= \bigsqcap \emptyset = \bigsqcup D\end{aligned}$$

denote the **least** and the **greatest element** of D , respectively.

Example 16.6

- 1 (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$ is a complete lattice with $\perp = AExp_c$ and $\top = \emptyset$
- 2 (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$ is a complete lattice with $\perp = \emptyset$ and $\top = Var_c$

Lemma 16.7

For a partial order $(D, \sqsubseteq)$ the claims

- ❶ *$(D, \sqsubseteq)$ is a complete lattice,*
- ❷ *every subset of D has a least upper bound, and*
- ❸ *every subset of D has a greatest lower bound*

are equivalent.

Lemma 16.7

For a partial order $(D, \sqsubseteq)$ the claims

- ❶ *$(D, \sqsubseteq)$ is a complete lattice,*
- ❷ *every subset of D has a least upper bound, and*
- ❸ *every subset of D has a greatest lower bound*

are equivalent.

Proof.

on the board



Chains represent the approximation of the analysis information.

Definition 16.8 (Chain; repetition of Def. 6.4 and 6.6)

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

- 1 A subset $S \subseteq D$ is called a **chain** in D if, for every $s_1, s_2 \in S$,
$$s_1 \sqsubseteq s_2 \text{ or } s_2 \sqsubseteq s_1$$

(that is, S is a totally ordered subset of D).

Chains represent the approximation of the analysis information.

Definition 16.8 (Chain; repetition of Def. 6.4 and 6.6)

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

- 1 A subset $S \subseteq D$ is called a **chain** in D if, for every $s_1, s_2 \in S$,
$$s_1 \sqsubseteq s_2 \text{ or } s_2 \sqsubseteq s_1$$

(that is, S is a totally ordered subset of D).
- 2 $(D, \sqsubseteq)$ is called **chain complete (CCPO)** if each of its chains has a least upper bound.

Chains represent the approximation of the analysis information.

Definition 16.8 (Chain; repetition of Def. 6.4 and 6.6)

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

- 1 A subset $S \subseteq D$ is called a **chain** in D if, for every $s_1, s_2 \in S$,
$$s_1 \sqsubseteq s_2 \text{ or } s_2 \sqsubseteq s_1$$

(that is, S is a totally ordered subset of D).
- 2 $(D, \sqsubseteq)$ is called **chain complete (CCPO)** if each of its chains has a least upper bound.
- 3 $(D, \sqsubseteq)$ satisfies the **Ascending Chain Condition (ACC)** if each ascending chain $d_1 \sqsubseteq d_2 \sqsubseteq \dots$ eventually stabilizes, i.e., there exists $n \in \mathbb{N}$ such that $d_n = d_{n+1} = \dots$.

Corollary 16.9

Every partial order that satisfies ACC is a CCPO.

Corollary 16.9

Every partial order that satisfies ACC is a CCPO.

Proof.

on the board



Corollary 16.9

Every partial order that satisfies ACC is a CCPO.

Proof.

on the board



Example 16.10

- 1 (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$ satisfies ACC since $AExp_c$ (unlike $AExp$) is finite

Corollary 16.9

Every partial order that satisfies ACC is a CCPO.

Proof.

on the board



Example 16.10

- 1 (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$ satisfies ACC since $AExp_c$ (unlike $AExp$) is finite
- 2 (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$ satisfies ACC since Var_c (unlike Var) is finite

Monotonicity of Functions

Transfer functions formalize the impact of a block in the program on the analysis information.

Definition 16.11 (Monotonicity; repetition of Def. 7.1)

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

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

Monotonicity of Functions

Transfer functions formalize the impact of a block in the program on the analysis information.

Definition 16.11 (Monotonicity; repetition of Def. 7.1)

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

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

Example 16.12

- 1 (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$
Each transfer function $\varphi_{l'}(A) := (A \setminus \text{kill}_{\text{AE}}(B^{l'})) \cup \text{gen}_{\text{AE}}(B^{l'})$ is monotonic

Monotonicity of Functions

Transfer functions formalize the impact of a block in the program on the analysis information.

Definition 16.11 (Monotonicity; repetition of Def. 7.1)

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

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

Example 16.12

- ❶ (Available Expressions) $(D, \sqsubseteq) = (2^{AExp_c}, \supseteq)$
Each transfer function $\varphi_{l'}(A) := (A \setminus \text{kill}_{AE}(B^{l'})) \cup \text{gen}_{AE}(B^{l'})$ is monotonic
- ❷ (Live Variables) $(D, \sqsubseteq) = (2^{Var_c}, \subseteq)$
Each transfer function $\varphi_{l'}(V) := (V \setminus \text{kill}_{LV}(B^{l'})) \cup \text{gen}_{LV}(B^{l'})$ is monotonic

Theorem 16.13 (Fixpoint Theorem; repetition of Thm. 7.7)

Let $(D, \sqsubseteq)$ be a CCPO and $F : D \rightarrow D$ continuous. Then

$$\text{fix}(F) := \bigsqcup \{F^n (\bigsqcup \emptyset) \mid n \in \mathbb{N}\}$$

is the least fixpoint of F .

Theorem 16.13 (Fixpoint Theorem; repetition of Thm. 7.7)

Let $(D, \sqsubseteq)$ be a CCPO and $F : D \rightarrow D$ continuous. Then

$$\text{fix}(F) := \bigsqcup \{F^n (\bigsqcup \emptyset) \mid n \in \mathbb{N}\}$$

is the least fixpoint of F .

Definition 16.14 (Continuity; repetition of Def. 7.5)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be CCPOs and $F : D \rightarrow D'$ monotonic. Then F is called **continuous** (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every non-empty chain $S \subseteq D$,

$$F (\bigsqcup S) = \bigsqcup F(S).$$

Theorem 16.13 (Fixpoint Theorem; repetition of Thm. 7.7)

Let $(D, \sqsubseteq)$ be a CCPO and $F : D \rightarrow D$ continuous. Then

$$\text{fix}(F) := \bigsqcup \{F^n (\bigsqcup \emptyset) \mid n \in \mathbb{N}\}$$

is the least fixpoint of F .

Definition 16.14 (Continuity; repetition of Def. 7.5)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be CCPOs and $F : D \rightarrow D'$ monotonic. Then F is called **continuous** (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every non-empty chain $S \subseteq D$,

$$F(\bigsqcup S) = \bigsqcup F(S).$$

Corollary 16.15

Monotonic functions on partial orders that satisfy ACC are continuous.

Fixpoints

Theorem 16.13 (Fixpoint Theorem; repetition of Thm. 7.7)

Let $(D, \sqsubseteq)$ be a CCPO and $F : D \rightarrow D$ continuous. Then

$$\text{fix}(F) := \bigsqcup \{F^n(\bigsqcup \emptyset) \mid n \in \mathbb{N}\}$$

is the least fixpoint of F .

Definition 16.14 (Continuity; repetition of Def. 7.5)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be CCPOs and $F : D \rightarrow D'$ monotonic. Then F is called **continuous** (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every non-empty chain $S \subseteq D$,

$$F(\bigsqcup S) = \bigsqcup F(S).$$

Corollary 16.15

Monotonic functions on partial orders that satisfy ACC are continuous.

Proof.

on the board



- 1 Repetition: Heading for a Dataflow Analysis Framework
- 2 Order-Theoretic Foundations
- 3 The Framework

Definition 16.16 (Dataflow system)

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

- a finite set of (program) **labels** L (here: L_c),
- a set of **extremal labels** $E \subseteq L$ (here: $\{\text{init}(c)\}$ or $\{\text{final}(c)\}$),
- a **flow relation** $F \subseteq L \times L$ (here: $\text{flow}(c)$ or $\text{flow}^R(c)$),
- a **complete lattice** $(D, \sqsubseteq)$ that satisfies 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** $\{\varphi_l \mid l \in L\}$ of type $\varphi_l : D \rightarrow D$.

Example 16.17

Problem	Available Expressions	Live Variables
E	$\{\text{init}(c)\}$	$\text{final}(c)$
F	$\text{flow}(c)$	$\text{flow}^R(c)$
D	2^{AExp_c}	2^{Var_c}
$\sqsubseteq$	$\supseteq$	$\subseteq$
$\sqcup$	$\bigcap$	$\bigcup$
$\perp$	$AExp_c$	$\emptyset$
ι	$\emptyset$	Var_c
φ_l	$\varphi_l(d) = (d \setminus \text{kill}(B^l)) \cup \text{gen}(B^l)$	