

Static Program Analysis

Lecture 5: Dataflow Analysis IV

(Worklist Algorithm & MOP Solution)

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/spa11/`

Summer Semester 2011

- 1 Repetition: Dataflow Systems
- 2 Uniqueness of Solutions
- 3 Efficient Fixpoint Computation
- 4 The MOP Solution
- 5 Another Analysis: Constant Propagation

Definition (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 $\varphi_l : D \rightarrow D$.

Dataflow Systems and Fixpoints

Definition (Dataflow equation system)

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

- S determines the **equation system** (where $I \in L$)

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

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

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

- S determines the **transformation**

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

where

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

Corollary

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

The Fixpoint Theorem



Alfred Tarski (1901–1983)



Bronislaw Knaster (1893–1990)

Theorem (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

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

Remark: $\text{ACC} \implies (\Phi^k(\perp) \mid k \in \mathbb{N})$ stabilizes at some $k_0 \in \mathbb{N}$ with $\text{fix}(\Phi) = \Phi^{k_0}(\perp)$ (where k_0 bounded by maximal chain length in $(D, \sqsubseteq)$)

- 1 Repetition: Dataflow Systems
- 2 Uniqueness of Solutions**
- 3 Efficient Fixpoint Computation
- 4 The MOP Solution
- 5 Another Analysis: Constant Propagation

Uniqueness of Solutions

(Non-minimal) solutions of dataflow equation systems are **not always unique**.

Example 5.1

- 1 Available Expressions: see Exercise 0.2
- 2 Live Variables: consider

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

$$\Rightarrow LV_3 = \{x\}$$

$$\begin{aligned}\Rightarrow LV_1 &= LV_2 \cup \{x\} \\ &= LV_1 \cup \{x\}\end{aligned}$$

$$\begin{aligned}\Rightarrow \text{Solutions: } LV_1 &= LV_2 = (\{x\} \text{ or } \{x, y\}), \\ LV_3 &= \{x\}, LV_4 = \{x, y\}\end{aligned}$$

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

- 1 Repetition: Dataflow Systems
- 2 Uniqueness of Solutions
- 3 Efficient Fixpoint Computation**
- 4 The MOP Solution
- 5 Another Analysis: Constant Propagation

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

⇒ optimization by **worklist**

Algorithm 5.2 (Worklist algorithm)

Input: *dataflow system* $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$

Variables: $W \in (L \times L)^*$, $\{AI_I \in D \mid I \in L\}$

Procedure: $W := \varepsilon$; **for** $(I, I') \in F$ **do** $W := W \cdot (I, I')$; % Initialize W
for $I \in L$ **do** % Initialize AI
 if $I \in E$ **then** $AI_I := \iota$ **else** $AI_I := \perp_D$;
 while $W \neq \varepsilon$ **do**
 $(I, I') := \text{head}(W)$; $W := \text{tail}(W)$;
 if $\varphi_I(AI_I) \not\sqsubseteq AI_{I'}$ **then** % Fixpoint not yet reached
 $AI_{I'} := AI_{I'} \sqcup \varphi_I(AI_I)$;
 for $(I', I'') \in F$ **do**
 if (I', I'') not in W **then** $W := (I', I'') \cdot W$;

Output: $\{AI_I \mid I \in L\}$

Example 5.3 (Worklist algorithm)

Available Expression analysis for $c =$

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

(cf. Examples 2.9 and 4.11)

Transfer functions:

$$\begin{aligned}\varphi_1(A) &= A \cup \{a+b\} \\ \varphi_2(A) &= A \cup \{a*b\} \\ \varphi_3(A) &= A \cup \{a+b\} \\ \varphi_4(A) &= A \setminus \{a+b, a*b, a+1\} \\ \varphi_5(A) &= A \cup \{a+b\}\end{aligned}$$

Computation protocol: on the board

A Worklist Algorithm III

Properties of the algorithm:

Theorem 5.4 (Correctness of worklist algorithm)

Given a dataflow system $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$, Algorithm 5.2 always terminates and computes $\text{fix}(\Phi_S)$.

Proof.

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



- 1 Repetition: Dataflow Systems
- 2 Uniqueness of Solutions
- 3 Efficient Fixpoint Computation
- 4 The MOP Solution**
- 5 Another Analysis: Constant Propagation

The MOP Solution I

- Other **solution method** for dataflow systems
- MOP = **Meet Over all Paths**
- Analysis information for block B^l = **least upper bound over all paths leading to l**

Definition 5.5 (Paths)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. For every $l \in L$, the set of **paths up to l** is given by

$$\text{Path}(l) := \{[l_1, \dots, l_{k-1}] \mid k \geq 1, l_1 \in E, \\ (l_i, l_{i+1}) \in F \text{ for every } 1 \leq i \leq k, l_k = l\}.$$

For a path $p = [l_1, \dots, l_{k-1}] \in \text{Path}(l)$, we define the **transfer function** $\varphi_p : D \rightarrow D$ by

$$\varphi_p := \varphi_{l_{k-1}} \circ \dots \circ \varphi_{l_1} \circ \text{id}_D$$

(so that $\varphi_{[]} = \text{id}_D$).

Definition 5.6 (MOP solution)

Let $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $L = \{l_1, \dots, l_n\}$. The **MOP solution** for S is determined by

$$\text{mop}(S) := (\text{mop}(l_1), \dots, \text{mop}(l_n)) \in D^n$$

where, for every $l \in L$,

$$\text{mop}(l) := \bigsqcup \{\varphi_p(\iota) \mid p \in \text{Path}(l)\}.$$

Remark:

- $\text{Path}(l)$ is generally infinite

⇒ not clear how to compute $\text{mop}(l)$

- In fact: MOP solution generally undecidable (later)

Example 5.7 (Live Variables; cf. Examples 3.3 and 4.12)

$$\begin{aligned}
 c = & \begin{array}{l} [x := 2]^1; \\ [y := 4]^2; \\ [x := 1]^3; \\ \text{if } [y > 0]^4 \text{ then} \\ \quad [z := x]^5 \\ \text{else} \\ \quad [z := y*y]^6; \\ [x := z]^7 \end{array} & \implies \text{mop}(1) = \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
 & = \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\{x, y, z\})))) \sqcup \\
 & \quad \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\{x, y, z\})))) \\
 & = \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{y, z\})))) \sqcup \\
 & \quad \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{y, z\})))) \\
 & = \varphi_2(\varphi_3(\varphi_4(\{x, y\}))) \sqcup \\
 & \quad \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
 & = \varphi_2(\varphi_3(\{x, y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
 & = \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
 & = \emptyset \sqcup \emptyset \\
 & = \emptyset \\
 \implies \text{Path}(1) = & \{[7, 5, 4, 3, 2], \\
 & \quad [7, 6, 4, 3, 2]\}
 \end{aligned}$$

- 1 Repetition: Dataflow Systems
- 2 Uniqueness of Solutions
- 3 Efficient Fixpoint Computation
- 4 The MOP Solution
- 5 Another Analysis: Constant Propagation

Goal of Constant Propagation Analysis

Constant Propagation Analysis

The goal of **Constant Propagation Analysis** is to determine, for each program point, whether a variable has a constant value whenever execution reaches that point.

Used for **Constant Folding**: replace reference to variable by constant value and evaluate constant expressions

Example 5.8 (Constant Propagation Analysis)

```
[x := 1]1;  
[y := 1]2;  
[z := 1]3;  
while [z > 0]4 do  
  [w := x+y]5;  
  if [w = 2]6 then  
    [x := y+2]7
```

- $y = z = 1$ at labels 4–7
- w, x not constant at labels 4–7
- possible optimizations:
 $[w := x+1]⁵ [x := 3]⁷$

Formalizing Constant Propagation Analysis I

The **dataflow system** $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ is given by

- set of labels $L := L_c$,
- extremal labels $E := \{\text{init}(c)\}$ (forward problem),
- flow relation $F := \text{flow}(c)$ (forward problem),
- complete lattice $(D, \sqsubseteq)$ where
 - $D := \{\delta \mid \delta : \text{Var}_c \rightarrow \mathbb{Z} \cup \{\perp, \top\}\}$
 - $\delta(x) = z \in \mathbb{Z}$: x has **constant value** z
 - $\delta(x) = \perp$: x **undefined**
 - $\delta(x) = \top$: x **overdefined** (i.e., different possible values)
 - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\perp \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)

Example 5.9

$$\text{Var}_c = \{w, x, y, z\},$$

$$\delta_1 = (\underbrace{\perp}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z), \quad \delta_2 = (\underbrace{3}_w, \underbrace{1}_x, \underbrace{4}_y, \underbrace{\top}_z)$$

$$\Rightarrow \delta_1 \sqcup \delta_2 = (\underbrace{3}_w, \underbrace{1}_x, \underbrace{\top}_y, \underbrace{\top}_z)$$

Dataflow system $S = (L, E, F, (D, \sqsubseteq), \iota, \varphi)$ (continued):

- extremal value $\iota := \delta_{\top} \in D$ where $\delta_{\top}(x) := \top$ for every $x \in \text{Var}_c$ (i.e., every x has (unknown) default value)
- transfer functions $\{\varphi_l \mid l \in L\}$ defined by

$$\varphi_l(\delta) := \begin{cases} \delta & \text{if } B^l = \text{skip or } B^l \in BExp \\ \delta[x \mapsto \text{val}_{\delta}(a)] & \text{if } B^l = (x := a) \end{cases}$$

where

$$\begin{aligned} \text{val}_{\delta}(x) &:= \delta(x) \\ \text{val}_{\delta}(z) &:= z \end{aligned} \quad \text{val}_{\delta}(a_1 \text{ op } a_2) := \begin{cases} z_1 \text{ op } z_2 & \text{if } z_1, z_2 \in \mathbb{Z} \\ \perp & \text{if } z_1 = \perp \text{ or } z_2 = \perp \\ \top & \text{otherwise} \end{cases}$$

for $z_1 := \text{val}_{\delta}(a_1)$ and $z_2 := \text{val}_{\delta}(a_2)$

Example 5.10

If $\delta = (\underbrace{\perp}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z)$, then

$$\varphi_I(\delta) = \begin{cases} (\underbrace{0}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z) & \text{if } B^I = (w := 0) \\ (\underbrace{3}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z) & \text{if } B^I = (w := y+1) \\ (\underbrace{\perp}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z) & \text{if } B^I = (w := w+x) \\ (\underbrace{\top}_w, \underbrace{1}_x, \underbrace{2}_y, \underbrace{\top}_z) & \text{if } B^I = (w := z+2) \end{cases}$$