Semantics and Verification of Software

Lecture 18: Dataflow Analysis

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw/

Summer semester 2007

Outline

1. Repetition: Solving Dataflow Equation Systems
2. Uniqueness of Solutions
3. Efficient Fixpoint Computation
4. The MOP Solution

Dataflow Systems

Definition (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: $\{\mathsf{init}(c)\}$ or $\mathsf{final}(c)$),
- a flow relation $F \subseteq Lab \times Lab$ (here: $\mathsf{flow}(c)$ or $\mathsf{flow}^R(c)$),
- a complete lattice $(D, \sqsubseteq)$ that satisfies ACC (with LUB operator $\bigsqcup$ and least element $\bot$),
- 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 $\varphi_l : D \to D$.

The Equation System

Definition (Dataflow equation system)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. S defines the following equation system over the set of variables $\{\mathsf{AI}_l \mid l \in Lab\}$:

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

The Functional and Its Fixpoint

Definition (Dataflow functional)

The equation system of a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ induces a functional

$$\Phi_S : D^n \to D^n : (d_{l_1}, \dots, d_{l_n}) \mapsto (d'_{l_1}, \dots, d'_{l_n})$$

where $Lab = \{l_1, \dots, l_n\}$ and

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

Corollary

The fixpoint of Φ_S is effectively computable by iteration:

$$\mathsf{fix}(\Phi_S) = \bigsqcup\{\Phi_S^i(\bot_{D^n}) \mid i \in \mathbb{N}\}$$

where $\bot_{D^n} = (\bot_D, \dots, \bot_D)$

Outline

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup (\mathsf{LV}_3 \cup \{x\}) \\ \mathsf{LV}_2 &= \mathsf{LV}_1 \cup \{x\} \\ \mathsf{LV}_3 &= \mathsf{LV}_4 \setminus \{y\} \\ \mathsf{LV}_4 &= \emptyset \end{aligned}$$

$$\Longrightarrow \mathsf{LV}_3 = \emptyset$$

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup \{x\} \\ &= \mathsf{LV}_1 \cup \{x\} \end{aligned}$$

$\Longrightarrow$ Solutions: $\mathsf{LV}_1 = \mathsf{LV}_2 = \{x\}$ or $\{x, y\}$, $\mathsf{LV}_3 = \mathsf{LV}_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup (\mathsf{LV}_3 \cup \{x\}) \\ \mathsf{LV}_2 &= \mathsf{LV}_1 \cup \{x\} \\ \mathsf{LV}_3 &= \mathsf{LV}_4 \setminus \{y\} \\ \mathsf{LV}_4 &= \emptyset \end{aligned}$$

$$\Longrightarrow \mathsf{LV}_3 = \emptyset$$

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup \{x\} \\ &= \mathsf{LV}_1 \cup \{x\} \end{aligned}$$

$\Longrightarrow$ Solutions: $\mathsf{LV}_1 = \mathsf{LV}_2 = \{x\}$ or $\{x, y\}$, $\mathsf{LV}_3 = \mathsf{LV}_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup (\mathsf{LV}_3 \cup \{x\}) \\ \mathsf{LV}_2 &= \mathsf{LV}_1 \cup \{x\} \\ \mathsf{LV}_3 &= \mathsf{LV}_4 \setminus \{y\} \\ \mathsf{LV}_4 &= \emptyset \end{aligned}$$

$$\Longrightarrow \mathsf{LV}_3 = \emptyset$$

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup \{x\} \\ &= \mathsf{LV}_1 \cup \{x\} \end{aligned}$$

$\Longrightarrow$ Solutions: $\mathsf{LV}_1 = \mathsf{LV}_2 = \{x\}$ or $\{x, y\}$, $\mathsf{LV}_3 = \mathsf{LV}_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

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

$$\Longrightarrow LV_3 = \emptyset$$

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

$\Longrightarrow$ Solutions: $LV_1 = LV_2 = \{x\}$ or $\{x, y\}$, $LV_3 = LV_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup (\mathsf{LV}_3 \cup \{x\}) \\ \mathsf{LV}_2 &= \mathsf{LV}_1 \cup \{x\} \\ \mathsf{LV}_3 &= \mathsf{LV}_4 \setminus \{y\} \\ \mathsf{LV}_4 &= \emptyset \end{aligned}$$

$$\Longrightarrow \mathsf{LV}_3 = \emptyset$$

$$\Longrightarrow \mathsf{LV}_1 = \mathsf{LV}_2 \cup \{x\} = \mathsf{LV}_1 \cup \{x\}$$

$\Longrightarrow$ Solutions: $\mathsf{LV}_1 = \mathsf{LV}_2 = \{x\}$ or $\{x, y\}$, $\mathsf{LV}_3 = \mathsf{LV}_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

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

$$\Longrightarrow LV_3 = \emptyset$$

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

$\Longrightarrow$ Solutions: $LV_1 = LV_2 = \{x\}$ or $\{x, y\}$, $LV_3 = LV_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

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

$$\Longrightarrow LV_3 = \emptyset$$

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

$\Longrightarrow$ Solutions: $LV_1 = LV_2 = \{x\}$ or $\{x, y\}$, $LV_3 = LV_4 = \emptyset$

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

Uniqueness of Solutions

Just as in the denotational semantics of `while` loops, solutions of dataflow equation systems are not unique.

Example 18.1

1. Available Expressions: see Exercise 9.2
2. Live Variables: consider

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

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup (\mathsf{LV}_3 \cup \{x\}) \\ \mathsf{LV}_2 &= \mathsf{LV}_1 \cup \{x\} \\ \mathsf{LV}_3 &= \mathsf{LV}_4 \setminus \{y\} \\ \mathsf{LV}_4 &= \emptyset \end{aligned}$$

$$\Longrightarrow \mathsf{LV}_3 = \emptyset$$

$$\Longrightarrow \begin{aligned} \mathsf{LV}_1 &= \mathsf{LV}_2 \cup \{x\} \\ &= \mathsf{LV}_1 \cup \{x\} \end{aligned}$$

$\Longrightarrow$ Solutions: $\mathsf{LV}_1 = \mathsf{LV}_2 = \{x\}$ or $\{x, y\}$, $\mathsf{LV}_3 = \mathsf{LV}_4 = \emptyset$

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

Outline

1. Repetition: Solving Dataflow Equation Systems
2. Uniqueness of Solutions
3. Efficient Fixpoint Computation
4. The MOP Solution

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step

$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed

$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure: $W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; *% Initialize W*

for $l \in Lab$ **do** *% Initialize* AI

    if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;

while $W \neq \varepsilon$ **do**

    $(l, l') :=$ **head**(W); $W :=$ **tail**(W);

    if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*

        $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;

        for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step

$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed

$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure:
$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; *% Initialize* W
for $l \in Lab$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step
$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed
$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure:
$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; % *Initialize* W
for $l \in Lab$ **do** % *Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** % *Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step

$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed

$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure: $W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; *% Initialize W*
for $l \in Lab$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step
$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed
$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure: $W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; *% Initialize W*
for $l \in Lab$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step

$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed

$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure:
$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; *% Initialize W*
for $l \in Lab$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step
$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed
$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure: $W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W$; *% Initialize W*
for $l \in Lab$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;
while $W \neq \varepsilon$ **do**
  $(l, l') :=$ **head**(W); $W :=$ **tail**(W);
  if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** *% Fixpoint not yet reached*
    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$;
    for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W$;

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm I

Observation: fixpoint iteration computes every AI_l in every step

$\Longrightarrow$ redundant if $\mathsf{AI}_{l'}$ at no F–predecessor l' changed

$\Longrightarrow$ optimization by worklist

Algorithm 18.2 (Worklist algorithm)

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

Variables: $W \in (Lab \times Lab)^*$, $\{\mathsf{AI}_l \in D \mid l \in Lab\}$

Procedure:

$W := \varepsilon;$ **for** $(l, l') \in F$ **do** $W := (l, l') \cdot W;$ % *Initialize* W
for $l \in Lab$ **do** % *Initialize* AI
    if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D;$
while $W \neq \varepsilon$ **do**
    $(l, l') :=$ **head**$(W); W :=$ **tail**$(W);$
    if $\varphi_l(\mathsf{AI}_l) \not\sqsubseteq \mathsf{AI}_{l'}$ **then** % *Fixpoint not yet reached*
        $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l);$
        for $(l', l'') \in F$ **do** $W := (l', l'') \cdot W;$

Output: $\{\mathsf{AI}_l \mid l \in Lab\}$

A Worklist Algorithm II

Example 18.3 (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 15.1 and 17.3)

Transfer functions:

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

Computation protocol: on the board

A Worklist Algorithm III

Properties of the algorithm:

Theorem 18.4 (Correctnes of worklist algorithm)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$, Algorithm 18.2 always terminates and computes $\mathsf{fix}(\Phi_S)$.

Proof.

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

A Worklist Algorithm III

Properties of the algorithm:

Theorem 18.4 (Correctnes of worklist algorithm)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$, *Algorithm 18.2 always terminates and computes* $\mathsf{fix}(\Phi_S)$.

Proof.

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

Outline

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 18.5 (Paths)

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

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

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

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

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

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 18.5 (Paths)

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

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

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

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

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

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 18.5 (Paths)

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

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

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

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

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

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 18.5 (Paths)

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

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

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

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

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

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 18.5 (Paths)

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

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

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

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

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

The MOP Solution II

Definition 18.6 (MOP solution)

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

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

where, for every $l \in Lab$,

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

Remark:

- $Path(l)$ is generally infinite

$\Longrightarrow$ not clear how to compute $\mathsf{mop}(l)$

- In fact: MOP solution generally undecidable (later)

The MOP Solution II

Definition 18.6 (MOP solution)

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

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

where, for every $l \in Lab$,

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

Remark:

- $Path(l)$ is generally infinite

$\Longrightarrow$ not clear how to compute $\mathsf{mop}(l)$

- In fact: MOP solution generally undecidable (later)

The MOP Solution II

Definition 18.6 (MOP solution)

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

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

where, for every $l \in Lab$,

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

Remark:

- $Path(l)$ is generally infinite

$\Longrightarrow$ not clear how to compute $\mathsf{mop}(l)$

- In fact: MOP solution generally undecidable (later)

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$\begin{aligned}
\Longrightarrow \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
&= \varphi_2(\varphi_3(\{x,y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
&= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \; \mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$
\begin{aligned}
\Longrightarrow \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
&= \varphi_2(\varphi_3(\{x,y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
&= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}
$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \; Path(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$\begin{aligned}
\Longrightarrow \; \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
&= \varphi_2(\varphi_3(\{x,y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
&= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}$$

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow$ $\mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$\begin{aligned}
\Longrightarrow \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
&= \varphi_2(\varphi_3(\{x,y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
&= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \; \mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$
\begin{aligned}
\Longrightarrow \; \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{\mathtt{z}\})))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{\mathtt{z}\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{\mathtt{x}\}))) \sqcup \\
&\quad\; \varphi_2(\varphi_3(\varphi_4(\{\mathtt{y}\}))) \\
&= \varphi_2(\varphi_3(\{\mathtt{x},\mathtt{y}\})) \sqcup \varphi_2(\varphi_3(\{\mathtt{y}\})) \\
&= \varphi_2(\{\mathtt{y}\}) \sqcup \varphi_2(\{\mathtt{y}\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}
$$

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$\begin{aligned}
\Longrightarrow \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
&= \varphi_2(\varphi_3(\{x, y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
&= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

$c =$ `[x := 2]`^{1}`;`
`[y := 4]`^{2}`;`
`[x := 1]`^{3}`;`
`if [y > 0]`^{4}` then`
`[z := x]`^{5}
`else`
`[z := y*y]`^{6}`;`
`[x := z]`^{7}

$\Longrightarrow\ Path(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$
\begin{aligned}
\Longrightarrow \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{y\}))) \\
&= \varphi_2(\varphi_3(\{x,y\})) \sqcup \varphi_2(\varphi_3(\{y\})) \\
&= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}
$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$
\begin{aligned}
\Longrightarrow \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{\mathtt{z}\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{\mathtt{z}\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{\mathtt{x}\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{\mathtt{y}\}))) \\
&= \varphi_2(\varphi_3(\{\mathtt{x},\mathtt{y}\})) \sqcup \varphi_2(\varphi_3(\{\mathtt{y}\})) \\
&= \varphi_2(\{\mathtt{y}\}) \sqcup \varphi_2(\{\mathtt{y}\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset
\end{aligned}
$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\Longrightarrow \mathit{Path}(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$\Longrightarrow \mathsf{mop}(1) = \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota)$$
$$= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset)))))$$
$$= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{z\})))) \sqcup \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{z\}))))$$
$$= \varphi_2(\varphi_3(\varphi_4(\{x\}))) \sqcup \varphi_2(\varphi_3(\varphi_4(\{y\})))$$
$$= \varphi_2(\varphi_3(\{x,y\})) \sqcup \varphi_2(\varphi_3(\{y\}))$$
$$= \varphi_2(\{y\}) \sqcup \varphi_2(\{y\})$$
$$= \emptyset \sqcup \emptyset$$
$$= \emptyset$$

The MOP Solution III

Example 18.7 (Live Variables; cf. Examples 15.4 and 17.4)

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

$\implies Path(1) = \{[7,5,4,3,2], [7,6,4,3,2]\}$

$$
\begin{aligned}
\implies \mathsf{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota)\\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\emptyset))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\emptyset)))))\\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{\mathtt{z}\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{\mathtt{z}\}))))\\
&= \varphi_2(\varphi_3(\varphi_4(\{\mathtt{x}\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{\mathtt{y}\})))\\
&= \varphi_2(\varphi_3(\{\mathtt{x},\mathtt{y}\})) \sqcup \varphi_2(\varphi_3(\{\mathtt{y}\}))\\
&= \varphi_2(\{\mathtt{y}\}) \sqcup \varphi_2(\{\mathtt{y}\})\\
&= \emptyset \sqcup \emptyset\\
&= \emptyset
\end{aligned}
$$