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

Outline

Dataflow Systems

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: $\{\mathrm{init}(c)\}$ or $\mathrm{final}(c)$),
- a flow relation $F \subseteq L \times L$ (here: $\mathrm{flow}(c)$ or $\mathrm{flow}^R(c)$),
- 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 $\{\varphi_l \mid l \in L\}$ of type $\varphi_l : D \to D$.

Dataflow Systems and Fixpoints

Definition (Dataflow equation system)

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

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

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

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

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

  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

$(d_1, \ldots, d_n) \in D^n$ 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 $\Phi : D \to D$ monotonic. Then

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

is the least fixpoint of Φ where

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

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

Outline

Uniqueness of Solutions

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

Uniqueness of Solutions

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

Example 5.1

1. Available Expressions: see Exercise 0.2

(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

$$
\begin{array}{l}
\texttt{while } [\texttt{x>1}]^1 \texttt{ do} \\
\quad [\texttt{skip}]^2\texttt{;} \\
[\texttt{x := x+1}]^3\texttt{;} \\
[\texttt{y := 0}]^4
\end{array}
$$

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

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

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

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

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

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

$$\Longrightarrow LV_3 = \{x\}$$

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

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

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

$$\Longrightarrow LV_3 = \{x\}$$

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

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

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

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

$$\Longrightarrow LV_3 = \{x\}$$

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

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

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

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

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

$$\Longrightarrow LV_3 = \{x\}$$

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

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

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

Outline

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

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

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

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

$\Longrightarrow$ optimization by worklist

A Worklist Algorithm I

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

Algorithm 5.2 (Worklist algorithm)

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

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

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

$\Longrightarrow$ optimization by worklist

Algorithm 5.2 (Worklist algorithm)

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

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

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

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

$\Longrightarrow$ optimization by worklist

Algorithm 5.2 (Worklist algorithm)

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

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

Procedure: $W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := W \cdot (l, l')$; *% Initialize W*
for $l \in L$ **do** *% Initialize* AI
  if $l \in E$ **then** $\mathsf{AI}_l := \iota$ **else** $\mathsf{AI}_l := \bot_D$;

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

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

$\Longrightarrow$ optimization by worklist

Algorithm 5.2 (Worklist algorithm)

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

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

Procedure:

$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := W \cdot (l, l')$; *% Initialize W*
for $l \in L$ **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**
      if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$;

A Worklist Algorithm I

Observation: fixpoint iteration re-computes every AI_l in every step

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

$\Longrightarrow$ optimization by worklist

Algorithm 5.2 (Worklist algorithm)

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

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

Procedure:

$W := \varepsilon$; **for** $(l, l') \in F$ **do** $W := W \cdot (l, l')$; *% Initialize W*
for $l \in L$ **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**
      if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$;

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

A Worklist Algorithm II

Example 5.3 (Worklist algorithm)

Available Expression analysis for $c =$ [x := a+b][1];

```
[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:

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

Computation protocol: on the board

A Worklist Algorithm III

Properties of the algorithm:

Theorem 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* $\text{fix}(\Phi_S)$*.*

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* $\mathrm{fix}(\Phi_S)$*.*

Proof.

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

Outline

The MOP Solution I

- Other solution method for dataflow systems

The MOP Solution I

- Other solution method for dataflow systems
- MOP = Meet Over all Paths

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

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

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

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

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

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

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

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

The MOP Solution II

Definition 5.6 (MOP solution)

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

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

where, for every $l \in L$,

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

The MOP Solution II

Definition 5.6 (MOP solution)

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

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

where, for every $l \in L$,

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

Remark:

- $Path(l)$ is generally infinite

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

The MOP Solution II

Definition 5.6 (MOP solution)

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

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

where, for every $l \in L$,

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

Remark:

- $Path(l)$ is generally infinite

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

- In fact: MOP solution generally undecidable (later)

The MOP Solution III

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

$c =$
```
[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
```

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

$c =$ [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

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

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

$c =$
```
[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
```

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

$\Longrightarrow \text{mop}(1) = \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota)$

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

$c =$
```
[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
```

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

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

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

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

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

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

The MOP Solution III

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

$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
```

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

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

The MOP Solution III

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

$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}

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

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

The MOP Solution III

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

```
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
```

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

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

The MOP Solution III

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

$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}

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

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

The MOP Solution III

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

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

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

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

Outline

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.

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

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
```

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

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

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:
  $[\texttt{w := x+1}]^5$ $[\texttt{x := 3}]^7$

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
  - $D := \{\delta \mid \delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}\}$
    - $\delta(x) = z \in \mathbb{Z}$: x has constant value z
    - $\delta(x) = \bot$: x undefined
    - $\delta(x) = \top$: x overdefined (i.e., different possible values)
  - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\bot \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)

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
  - $D := \{\delta \mid \delta : Var_c \to \mathbb{Z} \cup \{\bot, \top\}\}$
    - $\delta(x) = z \in \mathbb{Z}$: x has constant value z
    - $\delta(x) = \bot$: x undefined
    - $\delta(x) = \top$: x overdefined (i.e., different possible values)
  - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\bot \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)

Example 5.9

$Var_c = \{\mathtt{w}, \mathtt{x}, \mathtt{y}, \mathtt{z}\}$,

$\delta_1 = (\underbrace{\bot}_{\mathtt{w}}, \underbrace{1}_{\mathtt{x}}, \underbrace{2}_{\mathtt{y}}, \underbrace{\top}_{\mathtt{z}}), \ \delta_2 = (\underbrace{3}_{\mathtt{w}}, \underbrace{1}_{\mathtt{x}}, \underbrace{4}_{\mathtt{y}}, \underbrace{\top}_{\mathtt{z}})$

$\Longrightarrow \ \delta_1 \sqcup \delta_2 = (\underbrace{3}_{\mathtt{w}}, \underbrace{1}_{\mathtt{x}}, \underbrace{\top}_{\mathtt{y}}, \underbrace{\top}_{\mathtt{z}})$

Formalizing Constant Propagation Analysis II

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 $x \in Var_c$ (i.e., every x has (unknown) default value)
- transfer functions $\{\varphi_l \mid l \in L\}$ defined by

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

where

$$\begin{aligned} val_\delta(x) &:= \delta(x) \\ val_\delta(z) &:= z \end{aligned} \qquad val_\delta(a_1 \; op \; a_2) := \begin{cases} z_1 \; op \; z_2 & \text{if } z_1, z_2 \in \mathbb{Z} \\ \bot & \text{if } z_1 = \bot \text{ or } z_2 = \bot \\ \top & \text{otherwise} \end{cases}$$

for $z_1 := val_\delta(a_1)$ and $z_2 := val_\delta(a_2)$

Example 5.10

If $\delta = (\underbrace{\bot}_{\texttt{w}}, \underbrace{1}_{\texttt{x}}, \underbrace{2}_{\texttt{y}}, \underbrace{\top}_{\texttt{z}})$, then

$$\varphi_l(\delta) = \begin{cases} (\underbrace{0}_{\texttt{w}}, \underbrace{1}_{\texttt{x}}, \underbrace{2}_{\texttt{y}}, \underbrace{\top}_{\texttt{z}}) & \text{if } B^l = (\texttt{w := 0}) \\ (\underbrace{3}_{\texttt{w}}, \underbrace{1}_{\texttt{x}}, \underbrace{2}_{\texttt{y}}, \underbrace{\top}_{\texttt{z}}) & \text{if } B^l = (\texttt{w := y+1}) \\ (\underbrace{\bot}_{\texttt{w}}, \underbrace{1}_{\texttt{x}}, \underbrace{2}_{\texttt{y}}, \underbrace{\top}_{\texttt{z}}) & \text{if } B^l = (\texttt{w := w+x}) \\ (\underbrace{\top}_{\texttt{w}}, \underbrace{1}_{\texttt{x}}, \underbrace{2}_{\texttt{y}}, \underbrace{\top}_{\texttt{z}}) & \text{if } B^l = (\texttt{w := z+2}) \end{cases}$$