Static Program Analysis

Lecture 5: Dataflow Analysis IV (Worklist Algorithm & MOP Solution)

Winter Semester 2016/17

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

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

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

Re-Scheduling of First Exam

New date: Tue 21 Feb, 15:00 - 17:00, AH 2/3

Outline of Lecture 5

Recap: The Fixpoint Approach

Efficient Fixpoint Computation

The MOP Solution

Another Analysis: Constant Propagation

Recap: The Fixpoint Approach

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

Function requirements for dataflow analysis

All transfer functions must be a monotonic

Recap: The Fixpoint Approach

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: $\{\mathrm{init}(c)\}$ or $\mathrm{final}(c)$),
- a flow relation $F \subseteq Lab \times Lab$ (here: $\mathrm{flow}(c)$ or $\mathrm{flow}^R(c)$),
- a complete lattice $(D, \sqsubseteq)$ satisfying 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 $\varphi_l : D \to D$.

Recap: The Fixpoint Approach

Dataflow Systems and Fixpoints

Definition (Dataflow equation system)

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

- S determines the equation system (where $l \in Lab$)
$$\mathsf{AI}_l = \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$
- $(d_1, \ldots, d_n) \in D^n$ 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

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 $(d_1, \ldots, d_n) \sqsubseteq^n (d'_1, \ldots, d'_n)$ iff $d_i \sqsubseteq d'_i$ for every $1 \leq i \leq n$)
- Monotonicity of transfer functions φ_l in $(D, \sqsubseteq)$ implies monotonicity of Φ_S in $(D^n, \sqsubseteq^n)$ (since $\bigsqcup$ 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 $\perp_{D^n} = (\underbrace{\perp_D, \ldots, \perp_D}_{n \text{ times}})$
- If height of $(D, \sqsubseteq)$ is m
  $\Longrightarrow$ height of $(D^n, \sqsubseteq^n)$ is $m \cdot n$
  $\Longrightarrow$ fixpoint iteration requires at most $m \cdot n$ steps

Outline of Lecture 5

Recap: The Fixpoint Approach

Efficient Fixpoint Computation

The MOP Solution

Another Analysis: Constant Propagation

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

$\Longrightarrow$ optimisation by worklist over control-flow edges

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

$\Longrightarrow$ optimisation by worklist over control-flow edges

Algorithm 5.1 (Worklist algorithm)

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

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

$\Longrightarrow$ optimisation by worklist over control-flow edges

Algorithm 5.1 (Worklist algorithm)

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

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

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

$\Longrightarrow$ optimisation by worklist over control-flow edges

Algorithm 5.1 (Worklist algorithm)

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

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

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

for $l \in Lab$ **do**

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

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

$\implies$ optimisation by worklist over control-flow edges

Algorithm 5.1 (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** $W := W \cdot (l, l')$; *% Initialise W*

for $l \in Lab$ **do**

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

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

  $(l, l') :=$ **head**(W); $W :=$ **tail**(W); *% Next control-flow edge*

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

    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$; *% Update analysis information*

    for $(l', l'') \in F$ **do**

      if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$; *% Propagate modification*

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

$\Longrightarrow$ optimisation by worklist over control-flow edges

Algorithm 5.1 (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** $W := W \cdot (l, l')$; *% Initialise W*

for $l \in Lab$ **do**

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

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

  $(l, l') :=$ **head**(W); $W :=$ **tail**(W); *% Next control-flow edge*

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

    $\mathsf{AI}_{l'} := \mathsf{AI}_{l'} \sqcup \varphi_l(\mathsf{AI}_l)$; *% Update analysis information*

    for $(l', l'') \in F$ **do**

      if (l', l'') *not in* W **then** $W := (l', l'') \cdot W$; *% Propagate modification*

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

A Worklist Algorithm II

Example 5.2 (Worklist algorithm)

Available Expression analysis for $c =$
(cf. Examples 2.9 and 4.11)

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

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

Efficient Fixpoint Computation

An "Optimisation"

Conjecture: it suffices to initialise worklist with edges leaving extremal labels (such that analysis information will propagate through CFG)

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

An "Optimisation"

Conjecture: it suffices to initialise worklist with edges leaving extremal labels (such that analysis information will propagate through CFG)

But ...

Example 5.3 (Counterexample)

Live Variables analysis for $c = [\mathtt{x\ :=\ 0}]^1;$
$[\mathtt{x\ :=\ x\ +\ 1}]^2;$
$[\mathtt{x\ :=\ 2}]^3$

Solution: $\mathsf{LV}_1 = \{\mathtt{x}\}, \mathsf{LV}_2 = \emptyset, \mathsf{LV}_3 = \{\mathtt{x}\}$

An "Optimisation"

Conjecture: it suffices to initialise worklist with edges leaving extremal labels (such that analysis information will propagate through CFG)

But ...

Example 5.3 (Counterexample)

Live Variables analysis for $c = [\mathtt{x\ :=\ 0}]^1;$
$[\mathtt{x\ :=\ x\ +\ 1}]^2;$
$[\mathtt{x\ :=\ 2}]^3$

Solution: $\mathsf{LV}_1 = \{\mathtt{x}\}, \mathsf{LV}_2 = \emptyset, \mathsf{LV}_3 = \{\mathtt{x}\}$

"Optimised" worklist algorithm:

W	LV_1	LV_2	LV_3
$(3,2)$	$\emptyset$	$\emptyset$	$\{\mathtt{x}\}$

An "Optimisation"

Conjecture: it suffices to initialise worklist with edges leaving extremal labels (such that analysis information will propagate through CFG)

But ...

Example 5.3 (Counterexample)

Live Variables analysis for $c = [\texttt{x := 0}]^1; [\texttt{x := x + 1}]^2; [\texttt{x := 2}]^3$

Solution: $\mathsf{LV}_1 = \{\texttt{x}\}, \mathsf{LV}_2 = \emptyset, \mathsf{LV}_3 = \{\texttt{x}\}$

"Optimised" worklist algorithm:

W	LV_1	LV_2	LV_3
$(3,2)$	$\emptyset$	$\emptyset$	$\{\texttt{x}\}$
ε	$\emptyset$	$\emptyset$	$\{\texttt{x}\}$

$\Longrightarrow$ wrong result!

Correctness of Worklist Algorithm

Theorem 5.4 (Correctness of worklist algorithm)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$*, Algorithm 5.1 always terminates and computes* $\text{fix}(\Phi_S)$*.*

Correctness of Worklist Algorithm

Theorem 5.4 (Correctness of worklist algorithm)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$*, Algorithm 5.1 always terminates and computes* $\mathsf{fix}(\Phi_S)$*.*

Proof.

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

Outline of Lecture 5

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
  = most precise information for l (“reference solution”)

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
  = most precise information for l ("reference solution")

Definition 5.5 (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

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

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
  = most precise information for l (“reference solution”)

Definition 5.5 (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

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

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

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

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

The MOP Solution II

Definition 5.6 (MOP solution)

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

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

where, for every $l \in Lab$,

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

The MOP Solution II

Definition 5.6 (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

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

where, for every $l \in Lab$,

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

Remark:

- $Path(l)$ is generally infinite

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

The MOP Solution II

Definition 5.6 (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

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

where, for every $l \in Lab$,

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

Remark:

- $Path(l)$ is generally infinite

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

- In fact: MOP solution generally undecidable (later)

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 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
```

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 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}

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

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

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

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

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

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

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 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

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

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

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 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}

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

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

The MOP Solution III

Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

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

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

$$\begin{aligned}
\Longrightarrow \mathrm{mop}(1) &= \varphi_{[7,5,4,3,2]}(\iota) \sqcup \varphi_{[7,6,4,3,2]}(\iota) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\varphi_7(\{\mathtt{x},\mathtt{y},\mathtt{z}\}))))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\varphi_7(\{\mathtt{x},\mathtt{y},\mathtt{z}\}))))) \\
&= \varphi_2(\varphi_3(\varphi_4(\varphi_5(\{\mathtt{y},\mathtt{z}\})))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\varphi_6(\{\mathtt{y},\mathtt{z}\})))) \\
&= \varphi_2(\varphi_3(\varphi_4(\{\mathtt{x},\mathtt{y}\}))) \sqcup \\
&\quad\ \varphi_2(\varphi_3(\varphi_4(\{\mathtt{y}\}))) \\
&= \varphi_2(\varphi_3(\{\mathtt{x},\mathtt{y}\})) \sqcup \varphi_2(\varphi_3(\{\mathtt{y}\})) \\
&= \varphi_2(\{\mathtt{y}\}) \sqcup \varphi_2(\{\mathtt{y}\}) \\
&= \emptyset \sqcup \emptyset \\
&= \emptyset \quad \text{(same as } \mathrm{fix}(\Phi_S)[1] \text{ – Ex. 4.12)}
\end{aligned}$$

Outline of Lecture 5

Recap: The Fixpoint Approach

Efficient Fixpoint Computation

The MOP Solution

Another Analysis: Constant Propagation

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

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.

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

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)

$[\texttt{x := 1}]^1;[\texttt{y := 1}]^2;[\texttt{z := 1}]^3;$
$\texttt{while}\ [\texttt{z > 0}]^4\ \texttt{do}$
$\quad [\texttt{w := x+y}]^5;$
$\quad \texttt{if}\ [\texttt{w = 2}]^6\ \texttt{then}$
$\qquad [\texttt{x := y+2}]^7$
$\quad \texttt{end}$
$\texttt{end}$

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
  end
end
```

- y = z = 1 at labels 4–7

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
  end
end
```

- `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
  end
end
```

- `y` = `z` = 1 at labels 4–7
- `w`, `x` not constant at labels 4–7
- possible optimisations:
  `[true]^4`
  `[w := x+1]^5`
  `[x := 3]^7`

Formalising Constant Propagation Analysis I

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

- set of labels $Lab := Lab_c$,
- extremal labels $E := \{\mathrm{init}(c)\}$ (forward problem)
- flow relation $F := \mathrm{flow}(c)$ (forward problem)

Formalising Constant Propagation Analysis I

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

- set of labels $Lab := Lab_c$,
- 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 (i.e., possible values in $\{z\}$)
    - $\delta(x) = \bot$: x undefined (i.e., possible values in $\emptyset$)
    - $\delta(x) = \top$: x overdefined (i.e., possible values in $\mathbb{Z}$)
  - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\bot \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)

Formalising Constant Propagation Analysis I

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

- set of labels $Lab := Lab_c$,
- extremal labels $E := \{\mathrm{init}(c)\}$ (forward problem)
- flow relation $F := \mathrm{flow}(c)$ (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 (i.e., possible values in $\{z\}$)
    - $\delta(x) = \bot$: x undefined (i.e., possible values in $\emptyset$)
    - $\delta(x) = \top$: x overdefined (i.e., possible values in $\mathbb{Z}$)
  - $\sqsubseteq \subseteq D \times D$ defined by pointwise extension of $\bot \sqsubseteq z \sqsubseteq \top$ (for every $z \in \mathbb{Z}$)

Example 5.9

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

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

Formalising Constant Propagation Analysis II

Dataflow system $S = (\mathit{Lab}, E, F, (D, \sqsubseteq), \iota, \varphi)$ (continued):

- extremal value $\iota := \delta_\top \in D$ where $\delta_\top(x) := \top$ for every $x \in \mathit{Var}_c$ (i.e., every x has (unknown) default value)

Formalising Constant Propagation Analysis II

Dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ (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 $\{\varphi_l \mid l \in Lab\}$ 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

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

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

Formalising Constant Propagation Analysis III

Example 5.10

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

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