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/
Re-Scheduling of First Exam

New date: Tue 21 Feb, 15:00 - 17:00, AH 2/3
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 \(\Phi : D \rightarrow D\) monotonic. Then

\[
\text{fix}(\Phi) := \bigsqcup \left\{ \Phi^k(\bot) \mid k \in \mathbb{N} \right\}
\]

is the least fixpoint of \(\Phi\) 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 \( S = (\text{Lab}, E, F, (D, \sqsubseteq), \iota, \varphi) \) consists of

- a finite set of (program) labels \( \text{Lab} \) (here: \( \text{Lab}_c \)),
- a set of extremal labels \( E \subseteq \text{Lab} \) (here: \{init\( (c) \)\} or \text{final}(c)),
- a flow relation \( F \subseteq \text{Lab} \times \text{Lab} \) (here: \text{flow}(c) or \text{flow}^R(c)),
- a complete lattice \( (D, \sqsubseteq) \) satisfying ACC (with LUB operator \( \bigvee \) and least element \( \bot \)),
- an extremal value \( \iota \in D \) (for the extremal labels), and
- a collection of monotonic transfer functions \( \{\varphi_l \mid l \in \text{Lab}\} \) of type \( \varphi_l : D \rightarrow D \).
### Recap: The Fixpoint Approach

#### Dataflow Systems and Fixpoints

**Definition (Dataflow equation system)**

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

- \( S \) determines the equation system (where \( l \in \text{Lab} \))
  \[
  \text{AI}_l = \begin{cases} 
  \iota & \text{if } l \in E \\
  \bigsqcup \{ \varphi_{l'}(\text{AI}_{l'}) \mid (l', l) \in F \} & \text{otherwise}
  \end{cases}
  \]

- \((d_1, \ldots, d_n) \in D^n\) 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
  \[
  \Phi_S : D^n \rightarrow D^n : (d_1, \ldots, d_n) \mapsto (d'_1, \ldots, d'_n)
  \]
  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 \( \Phi_S \)
Recap: The Fixpoint Approach

Solving Dataflow Problems by Fixpoint Iteration

Remarks:

- \((D, \sqsubseteq)\) being a complete lattice ensures that \(\Phi_S\) 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 \(\varphi_I\) in \((D, \sqsubseteq)\) implies monotonicity of \(\Phi_S\) in \((D^n, \sqsubseteq^n)\) (since \(\sqcup\) also monotonic)
- Thus the (least) fixpoint is effectively computable by iteration:

\[
\text{fix}(\Phi_S) = \bigsqcup \{\Phi^k_S(\bot_{D^n}) \mid k \in \mathbb{N}\}
\]

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

- If height of \((D, \sqsubseteq)\) is \(m\)
  \(\implies\) height of \((D^n, \sqsubseteq^n)\) is \(m \cdot n\)
  \(\implies\) fixpoint iteration requires at most \(m \cdot n\) steps
A Worklist Algorithm I

Observation: fixpoint iteration re-computes every $A_l$ in every step

$\implies$ redundant if $A_{l'}$ at no $F$-predecessor $l'$ changed

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

Algorithm 5.1 (Worklist algorithm)

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

Variables: $W \in (\text{Lab} \times \text{Lab})^*$, $\{A_l \in D \mid l \in \text{Lab}\}$

Procedure:

1. $W := \varepsilon$; for $(l, l') \in F$ do $W := W \cdot (l, l')$; % Initialise $W$
2. for $l \in \text{Lab}$ do
   - if $l \in E$ then $A_l := \iota$ else $A_l := \bot_D$; % Initialise $A_l$
3. while $W \neq \varepsilon$ do
   - $(l, l') := \text{head}(W)$; $W := \text{tail}(W)$; % Next control-flow edge
   - if $\varphi_{l}(A_l) \not\sqsubseteq A_{l'}$ then
     - $A_{l'} := A_{l'} \sqcup \varphi_{l}(A_l)$; % Update analysis information
   - for $(l', l'') \in F$ do
     - if $(l', l'')$ not in $W$ then $W := (l', l'') \cdot W$; % Propagate modification

Output: $\{A_l \mid l \in \text{Lab}\}$
Efficient Fixpoint Computation

A Worklist Algorithm II

Example 5.2 (Worklist algorithm)

Available Expression analysis for
\[c = [x := a+b]^1; \]
\[y := a*b]^2;\]
while \([y > a+b]^3\) do
\[a := a+1]^4;\]
\[x := a+b]^5\]
end

Transfer functions:
\[\varphi_1(A) = A \cup \{a+b\}\]
\[\varphi_2(A) = A \cup \{a*b\}\]
\[\varphi_3(A) = A \cup \{a+b\}\]
\[\varphi_4(A) = A \setminus \{a+b, a*b, a+1\}\]
\[\varphi_5(A) = A \cup \{a+b\}\]

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)

**But ...**

### Example 5.3 (Counterexample)

Live Variables analysis for \( c = [x := 0]^1; \\
[x := x + 1]^2; \\
[x := 2]^3 \)

Solution: \( LV_1 = \{x\}, LV_2 = \emptyset, LV_3 = \{x\} \)

“Optimised” worklist algorithm:

<table>
<thead>
<tr>
<th>( W )</th>
<th>( LV_1 )</th>
<th>( LV_2 )</th>
<th>( LV_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(3, 2)</td>
<td>\emptyset</td>
<td>\emptyset</td>
<td>{x}</td>
</tr>
<tr>
<td>( \varepsilon )</td>
<td>\emptyset</td>
<td>\emptyset</td>
<td>{x}</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \text{wrong result!} \]
Efficient Fixpoint Computation

Correctness of Worklist Algorithm

Theorem 5.4 (Correctness of worklist algorithm)

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

Proof.

see [Nielson/Nielson/Hankin 2005, p. 75 ff]
The MOP 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 $S = (\text{Lab}, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. For every $l \in \text{Lab}$, the set of paths up to $l$ is given by

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

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

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

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

The MOP Solution II

Definition 5.6 (MOP solution)

Let $S = (\text{Lab}, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $\text{Lab} = \{l_1, \ldots, l_n\}$. 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 $l \in \text{Lab}$,

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

Remark:
- $\text{Path}(l)$ is generally infinite
- $\Rightarrow$ not clear how to compute $\text{mop}(l)$
- In fact: MOP solution generally undecidable (later)
Example 5.7 (Live Variables; cf. Examples 2.12 and 4.12)

c = [x := 2]¹;
[y := 4]²;
x := 1³;
if [y > 0]⁴ then
  [z := x]⁵
else
  [z := y*y]⁶;
x := z]⁷

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

⇒ mop(1) = ϕ[7,5,4,3,2](l) ⊔ ϕ[7,6,4,3,2](l)
           = ϕ₂(ϕ₃(ϕ₄(ϕ₅(ϕ₇({x, y, z})))))) ⊔
             ϕ₂(ϕ₃(ϕ₄(ϕ₆(ϕ₇({x, y, z}))))))
           = ϕ₂(ϕ₃(ϕ₄(ϕ₅({y, z})))) ⊔
             ϕ₂(ϕ₃(ϕ₄({y})))
           = ϕ₂({x, y}) ⊔ ϕ₂({y})
           = ∅ ⊔ ∅
           = ∅    (same as fix(Φₛ)[1] – Ex. 4.12)
Another Analysis: Constant Propagation

Goal of 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)

\[
\begin{align*}
[x &:= 1]^1;[y := 1]^2;[z := 1]^3; \\
\text{while } [z > 0]^4 \text{ do} \\
&w := x+y]^5; \\
&\text{if } [w = 2]^6 \text{ then} \\
&[x := y+2]^7 \\
&\text{end} \\
&\text{end}
\end{align*}
\]

- \( y = z = 1 \) at labels 4–7
- \( w, x \) not constant at labels 4–7
- possible optimisations:
  - \([true]^4\)
  - \([w := x+1]^5\)
  - \([x := 3]^7\)
Another Analysis: Constant Propagation

Formalising Constant Propagation Analysis I

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

- set of labels $\text{Lab} := \text{Lab}_c$,
- extremal labels $E := \{\text{init}(c)\}$ (forward problem)
- flow relation $F := \text{flow}(c)$ (forward problem)
- complete lattice $(D, \sqsubseteq)$ where
  - $D := \{\delta \mid \delta : \text{Var}_c \to \mathbb{Z} \cup \{\bot, \top\}\}$
    - $\delta(x) = z \in \mathbb{Z}$: $x$ 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 $\bot \subseteq z \subseteq \top$ (for every $z \in \mathbb{Z}$)

Example 5.9

$\text{Var}_c = \{w, x, y, z\}$, $\delta_1 = (\bot, 1, 2, \top)$, $\delta_2 = (3, 1, 4, \top)$

$\implies \delta_1 \sqcup \delta_2 = (3, 1, \top, \top)$
Another Analysis: Constant Propagation

Formalising Constant Propagation Analysis II

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

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

where

$$
\text{val}_\delta(x) := \delta(x) \\
\text{val}_\delta(z) := z
$$

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

Formalising Constant Propagation Analysis III

Example 5.10

If \( \delta = (\perp, 1, 2, \top) \), then

\[
\varphi_l(\delta) = \begin{cases} 
(0, 1, 2, \top) & \text{if } B^l = (w := 0) \\
(3, 1, 2, \top) & \text{if } B^l = (w := y+1) \\
(\perp, 1, 2, \top) & \text{if } B^l = (w := w+x) \\
(\top, 1, 2, \top) & \text{if } B^l = (w := z+2) 
\end{cases}
\]