Static Program Analysis

Lecture 4: Dataflow Analysis III (The Framework)

Summer Semester 2018

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-18/spa/

Written Exams

- First: Thu Jul 26, 10:00–13:00 AH 1/2
- Second: Tue Sep 18, 10:00–13:00 AH 2

Outline of Lecture 4

Recap: Algebraic Foundations of Dataflow Analysis

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results
3. Establish least upper bound as combination operator
4. Ensure monotonicity of transfer functions
5. Guarantee termination of fixpoint iteration by ascending chain condition
6. Optimise fixpoint iteration by worklist algorithm

Recap: Algebraic Foundations of Dataflow Analysis

Complete Lattices

Since $\{\varphi_{l'}(\mathsf{AI}_{l'}) \mid (l', l) \in F\}$ can contain arbitrary elements, the existence of least upper bounds must be ensured for arbitrary subsets.

Definition (Complete lattice)

A complete lattice is a partial order $(D, \sqsubseteq)$ such that all subsets of D have least upper bounds. In this case,

$$\bot := \bigsqcup \emptyset$$

denotes the least element of D.

Recap: Algebraic Foundations of Dataflow Analysis

The Ascending Chain Condition

Termination of fixpoint iteration is guaranteed by the following condition.

Definition (Ascending Chain Condition)

- A sequence $(d_i)_{i \in \mathbb{N}}$ is called an ascending chain in D if $d_i \sqsubseteq d_{i+1}$ for each $i \in \mathbb{N}$.
- A partial order $(D, \sqsubseteq)$ satisfies the Ascending Chain Condition (ACC) if each ascending chain $d_0 \sqsubseteq d_1 \sqsubseteq \ldots$ eventually stabilises, i.e., there exists $n \in \mathbb{N}$ such that $d_n = d_{n+1} = \ldots$

Notes:

- The finite height property implies ACC, but not vice versa (as there might be non-stabilising descending chains or stabilising chains of unbounded size – see following slide)
- The complete lattice and ACC properties are orthogonal (see following slide)

Domain requirements for dataflow analysis

$(D, \sqsubseteq)$ must be a complete lattice satisfying ACC

Monotonicity of Functions

Monotonicity of transfer functions excludes “oscillating behaviour” in fixpoint iteration.

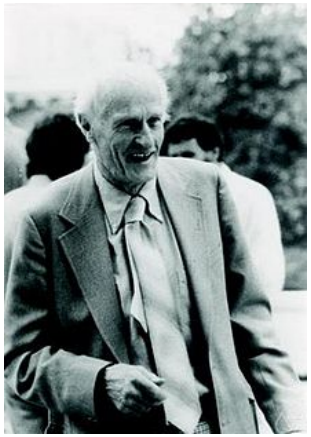
Definition (Monotonicity)

Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be partial orders, and let $\Phi : D \to D'$. Φ is called monotonic (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every $d_1, d_2 \in D$,

$$d_1 \sqsubseteq d_2 \implies \Phi(d_1) \sqsubseteq' \Phi(d_2).$$

Recap: Algebraic Foundations of Dataflow Analysis

The Fixpoint Theorem



Stephen Kleene (1909–1994)

Theorem (Fixpoint Theorem by Kleene)

Let $(D, \sqsubseteq)$ *be a complete lattice satisfying ACC and* $\Phi : D \to D$ *monotonic. Then*

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

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

Outline of Lecture 4

Dataflow Systems I

Definition 4.1 (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: $\{\text{init}(c)\}$ or $\text{final}(c)$),
- a flow relation $F \subseteq Lab \times Lab$ (here: $\text{flow}(c)$ or $\text{flow}^R(c)$),
- 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 family of monotonic transfer functions $\{\varphi_l \mid l \in Lab\}$ of type $\varphi_l : D \to D$.

Dataflow Systems II

Example 4.2

Problem	Available Expressions	Live Variables
E	$\{\text{init}(c)\}$	$\text{final}(c)$
F	$\text{flow}(c)$	$\text{flow}^R(c)$
D	2^{CExp_c}	2^{Var_c}
$\sqsubseteq$	$\supseteq$	$\subseteq$
$\bigsqcup$	$\bigcap$	$\bigcup$
$\bot$	$CExp_c$	$\emptyset$
ι	$\emptyset$	Var_c
φ_l	$\varphi_l(d) = (d \setminus \text{kill}(B^l)) \cup \text{gen}(B^l)$	

Dataflow Systems and Fixpoints

Definition 4.3 (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 $l \in Lab$)

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

Dataflow Systems and Fixpoints

Definition 4.3 (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

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

Dataflow Systems and Fixpoints

Definition 4.3 (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

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

Dataflow Systems and Fixpoints

Definition 4.3 (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 4.4

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

Solving Dataflow Problems by Fixpoint Iteration

- $(D, \sqsubseteq)$ being a complete lattice ensures that Φ_S is well defined

Solving Dataflow Problems by Fixpoint Iteration

- $(D, \sqsubseteq)$ 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 $1 \leq i \leq n$)

Solving Dataflow Problems by Fixpoint Iteration

- $(D, \sqsubseteq)$ 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)

Solving Dataflow Problems by Fixpoint Iteration

- $(D, \sqsubseteq)$ 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:

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

where $\bot_{D^n} = (\underbrace{\bot_D, \ldots, \bot_D}_{n \text{ times}})$

Solving Dataflow Problems by Fixpoint Iteration

- $(D, \sqsubseteq)$ 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:

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

  where $\bot_{D^n} = (\underbrace{\bot_D, \ldots, \bot_D}_{n \text{ times}})$
- If height of $(D, \sqsubseteq)$ is $m \in \mathbb{N}$
  $\Longrightarrow$ height of $(D^n, \sqsubseteq^n)$ is $(m-1) \cdot n + 1$
  $\Longrightarrow$ fixpoint iteration requires at most $m \cdot n$ steps

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

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

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

$$
\begin{aligned}
c = {} & [\texttt{x := a+b}]^1; \\
& [\texttt{y := a*b}]^2; \\
& \texttt{while } [\texttt{y > a+b}]^3 \texttt{ do} \\
& \quad [\texttt{a := a+1}]^4; \\
& \quad [\texttt{x := a+b}]^5 \\
& \texttt{end}
\end{aligned}
$$

Equation system:

$$
\begin{aligned}
AE_1 &= \emptyset \\
AE_2 &= AE_1 \cup \{\texttt{a+b}\} \\
AE_3 &= (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\}) \\
AE_4 &= AE_3 \cup \{\texttt{a+b}\} \\
AE_5 &= AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}
\end{aligned}
$$

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

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

Equation system:

$$AE_1 = \emptyset$$
$$AE_2 = AE_1 \cup \{\texttt{a+b}\}$$
$$AE_3 = (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\})$$
$$AE_4 = AE_3 \cup \{\texttt{a+b}\}$$
$$AE_5 = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$$

Fixpoint iteration:

k	1	2	3	4	5
0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

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

Equation system:

$$AE_1 = \emptyset$$
$$AE_2 = AE_1 \cup \{\texttt{a+b}\}$$
$$AE_3 = (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\})$$
$$AE_4 = AE_3 \cup \{\texttt{a+b}\}$$
$$AE_5 = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$$

Fixpoint iteration:

k	1	2	3	4	5
0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$
1	$\emptyset$	$CExp_c$	$CExp_c$	$CExp_c$	$\emptyset$

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

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

Equation system:

$$AE_1 = \emptyset$$
$$AE_2 = AE_1 \cup \{\texttt{a+b}\}$$
$$AE_3 = (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\})$$
$$AE_4 = AE_3 \cup \{\texttt{a+b}\}$$
$$AE_5 = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$$

Fixpoint iteration:

k	1	2	3	4	5
0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$
1	$\emptyset$	$CExp_c$	$CExp_c$	$CExp_c$	$\emptyset$
2	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$CExp_c$	$\emptyset$

Application to Dataflow Analysis

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

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

Equation system:

$$AE_1 = \emptyset$$
$$AE_2 = AE_1 \cup \{\texttt{a+b}\}$$
$$AE_3 = (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\})$$
$$AE_4 = AE_3 \cup \{\texttt{a+b}\}$$
$$AE_5 = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$$

Fixpoint iteration:

k	1	2	3	4	5
0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$
1	$\emptyset$	$CExp_c$	$CExp_c$	$CExp_c$	$\emptyset$
2	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$CExp_c$	$\emptyset$
3	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\emptyset$

Example: Available Expressions

Example 4.5 (Available Expressions; cf. Example 2.9)

Program:

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

Equation system:

$$AE_1 = \emptyset$$
$$AE_2 = AE_1 \cup \{\texttt{a+b}\}$$
$$AE_3 = (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\})$$
$$AE_4 = AE_3 \cup \{\texttt{a+b}\}$$
$$AE_5 = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$$

Fixpoint iteration:

k	1	2	3	4	5
0	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$	$CExp_c$
1	$\emptyset$	$CExp_c$	$CExp_c$	$CExp_c$	$\emptyset$
2	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$CExp_c$	$\emptyset$
3	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\emptyset$
4	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\emptyset$

Example: Live Variables

Example 4.6 (Live Variables; cf. Example 2.12)

Program:

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

Example: Live Variables

Example 4.6 (Live Variables; cf. Example 2.12)

Program:

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

Equation system:

$$\begin{aligned}
LV_1 &= LV_2 \setminus \{y\} \\
LV_2 &= LV_3 \setminus \{x\} \\
LV_3 &= LV_4 \cup \{y\} \\
LV_4 &= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup ((LV_6 \setminus \{z\}) \cup \{y\}) \\
LV_5 &= (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_6 &= (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_7 &= \{x, y, z\}
\end{aligned}$$

Example: Live Variables

Example 4.6 (Live Variables; cf. Example 2.12)

Program:

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

Equation system:

$$\begin{aligned}
LV_1 &= LV_2 \setminus \{y\} \\
LV_2 &= LV_3 \setminus \{x\} \\
LV_3 &= LV_4 \cup \{y\} \\
LV_4 &= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup ((LV_6 \setminus \{z\}) \cup \{y\}) \\
LV_5 &= (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_6 &= (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_7 &= \{x, y, z\}
\end{aligned}$$

Fixpoint iteration:

k	1	2	3	4	5	6	7
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Example: Live Variables

Example 4.6 (Live Variables; cf. Example 2.12)

Program:

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

Equation system:

$$\begin{aligned}
LV_1 &= LV_2 \setminus \{y\}\\
LV_2 &= LV_3 \setminus \{x\}\\
LV_3 &= LV_4 \cup \{y\}\\
LV_4 &= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup ((LV_6 \setminus \{z\}) \cup \{y\})\\
LV_5 &= (LV_7 \setminus \{x\}) \cup \{z\}\\
LV_6 &= (LV_7 \setminus \{x\}) \cup \{z\}\\
LV_7 &= \{x, y, z\}
\end{aligned}$$

Fixpoint iteration:

k	1	2	3	4	5	6	7
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{z\}$	$\{z\}$	$\{x, y, z\}$

Example: Live Variables

Example 4.6 (Live Variables; cf. Example 2.12)

Program:

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

Equation system:

$$\begin{aligned}
LV_1 &= LV_2 \setminus \{y\}\\
LV_2 &= LV_3 \setminus \{x\}\\
LV_3 &= LV_4 \cup \{y\}\\
LV_4 &= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup ((LV_6 \setminus \{z\}) \cup \{y\})\\
LV_5 &= (LV_7 \setminus \{x\}) \cup \{z\}\\
LV_6 &= (LV_7 \setminus \{x\}) \cup \{z\}\\
LV_7 &= \{x, y, z\}
\end{aligned}$$

Fixpoint iteration:

k	1	2	3	4	5	6	7
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{z\}$	$\{z\}$	$\{x, y, z\}$
2	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{x, y\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\}$

Example: Live Variables

Example 4.6 (Live Variables; cf. Example 2.12)

Program:

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

Equation system:

$$\begin{aligned}
LV_1 &= LV_2 \setminus \{y\}\\
LV_2 &= LV_3 \setminus \{x\}\\
LV_3 &= LV_4 \cup \{y\}\\
LV_4 &= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup ((LV_6 \setminus \{z\}) \cup \{y\})\\
LV_5 &= (LV_7 \setminus \{x\}) \cup \{z\}\\
LV_6 &= (LV_7 \setminus \{x\}) \cup \{z\}\\
LV_7 &= \{x, y, z\}
\end{aligned}$$

Fixpoint iteration:

k	1	2	3	4	5	6	7
0	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
1	$\emptyset$	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{z\}$	$\{z\}$	$\{x, y, z\}$
2	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{x, y\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\}$
3	$\emptyset$	$\{y\}$	$\{x, y\}$	$\{x, y\}$	$\{y, z\}$	$\{y, z\}$	$\{x, y, z\}$

Outline of Lecture 4

Uniqueness of Solutions I

Observation: (non-minimal) solutions of dataflow equation systems are not always unique.

Uniqueness of Solutions I

Observation: (non-minimal) solutions of dataflow equation systems are not always unique.

Example 4.7 (Available Expressions)

```
[z := x+y]^1;
while [true]^2 do
  [skip]^3
end
```

Uniqueness of Solutions I

Observation: (non-minimal) solutions of dataflow equation systems are not always unique.

Example 4.7 (Available Expressions)

```
[z := x+y]^1;
while [true]^2 do
  [skip]^3
end
```

$$\Longrightarrow \quad \begin{aligned} AE_1 &= \emptyset \\ AE_2 &= (AE_1 \cup \{\texttt{x+y}\}) \cap AE_3 \\ AE_3 &= AE_2 \end{aligned}$$

Uniqueness of Solutions I

Observation: (non-minimal) solutions of dataflow equation systems are not always unique.

Example 4.7 (Available Expressions)

$[\texttt{z := x+y}]^1;$
$\texttt{while}\ [\texttt{true}]^2\ \texttt{do}$
$\quad [\texttt{skip}]^3$
$\texttt{end}$

$$\Longrightarrow \quad \begin{aligned} AE_1 &= \emptyset \\ AE_2 &= (AE_1 \cup \{\texttt{x+y}\}) \cap AE_3 \\ AE_3 &= AE_2 \end{aligned}$$

$$\Longrightarrow \quad \begin{aligned} AE_1 &= \emptyset \\ AE_2 &= \{\texttt{x+y}\} \cap AE_3 \\ AE_3 &= AE_2 \end{aligned}$$

Uniqueness of Solutions I

Observation: (non-minimal) solutions of dataflow equation systems are not always unique.

Example 4.7 (Available Expressions)

```
[z := x+y]^1;
while [true]^2 do
  [skip]^3
end
```

$$\Longrightarrow \begin{aligned} AE_1 &= \emptyset \\ AE_2 &= (AE_1 \cup \{x+y\}) \cap AE_3 \\ AE_3 &= AE_2 \end{aligned}$$

$$\Longrightarrow \begin{aligned} AE_1 &= \emptyset \\ AE_2 &= \{x+y\} \cap AE_3 \\ AE_3 &= AE_2 \end{aligned}$$

$\Longrightarrow$ Solutions: $AE_1 = AE_2 = AE_3 = \emptyset$ or
$AE_1 = \emptyset, AE_2 = AE_3 = \{x+y\}$

Uniqueness of Solutions I

Observation: (non-minimal) solutions of dataflow equation systems are not always unique.

Example 4.7 (Available Expressions)

$$
\begin{array}{l}
[\texttt{z := x+y}]^1; \\
\texttt{while } [\texttt{true}]^2 \texttt{ do} \\
\quad [\texttt{skip}]^3 \\
\texttt{end}
\end{array}
\quad
\begin{array}{ll}
\Longrightarrow & \mathsf{AE}_1 = \emptyset \\
& \mathsf{AE}_2 = (\mathsf{AE}_1 \cup \{\texttt{x+y}\}) \cap \mathsf{AE}_3 \\
& \mathsf{AE}_3 = \mathsf{AE}_2 \\
\Longrightarrow & \mathsf{AE}_1 = \emptyset \\
& \mathsf{AE}_2 = \{\texttt{x+y}\} \cap \mathsf{AE}_3 \\
& \mathsf{AE}_3 = \mathsf{AE}_2
\end{array}
$$

$\Longrightarrow$ Solutions: $\mathsf{AE}_1 = \mathsf{AE}_2 = \mathsf{AE}_3 = \emptyset$ or
$\mathsf{AE}_1 = \emptyset, \mathsf{AE}_2 = \mathsf{AE}_3 = \{\texttt{x+y}\}$

Here: greatest solution (w.r.t. $\subseteq$) $\{\texttt{x+y}\}$ (maximal potential for optimisation)

Uniqueness of Solutions II

Example 4.8 (Live Variables)

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

Uniqueness of Solutions II

Example 4.8 (Live Variables)

```
while [x>1]^1 do
  [skip]^2
end;
[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 II

Example 4.8 (Live Variables)

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

$$\Longrightarrow \mathsf{LV}_1 = \mathsf{LV}_2 \cup (\mathsf{LV}_3 \cup \{\mathtt{x}\})$$
$$\mathsf{LV}_2 = \mathsf{LV}_1 \cup \{\mathtt{x}\}$$
$$\mathsf{LV}_3 = \mathsf{LV}_4 \setminus \{\mathtt{y}\}$$
$$\mathsf{LV}_4 = \{\mathtt{x}, \mathtt{y}\}$$
$$\Longrightarrow \mathsf{LV}_3 = \{\mathtt{x}\}$$

Uniqueness of Solutions II

Example 4.8 (Live Variables)

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

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

Uniqueness of Solutions II

Example 4.8 (Live Variables)

```
while [x>1]^1 do
  [skip]^2
end;
[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 II

Example 4.8 (Live Variables)

```
while [x>1]^1 do
  [skip]^2
end;
[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: $LV_1 = LV_2 = (\{x\}$ or $\{x, y\})$,
$LV_3 = \{x\}, LV_4 = \{x, y\}$

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

Outline of Lecture 4

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 $\mathsf{AI}_{l'}$ 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 4.9 (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 $\mathsf{AI}_{l'}$ at no F-predecessor l' changed

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

Algorithm 4.9 (Worklist algorithm)

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

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

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 4.9 (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

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 4.9 (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**

    $(l, l') := \mathbf{head}(W)$; $W := \mathbf{tail}(W)$; *% 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 4.9 (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**

    $(l, l') := \mathbf{head}(W)$; $W := \mathbf{tail}(W)$; *% 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 4.10 (Worklist algorithm)

Available Expression analysis for $c =$ (cf. Examples 2.9 and 4.5)

```
[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:

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

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)

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 4.11 (Counterexample)

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

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

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 4.11 (Counterexample)

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

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
$(3,2)$	$\emptyset$	$\emptyset$	$\{\mathtt{x}\}$

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 4.11 (Counterexample)

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

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
$(3,2)$	$\emptyset$	$\emptyset$	$\{\mathtt{x}\}$
ε	$\emptyset$	$\emptyset$	$\{\mathtt{x}\}$

$\Longrightarrow$ wrong result!

Correctness of Worklist Algorithm

Theorem 4.12 (Correctness of worklist algorithm)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$*, Algorithm 4.9 always terminates and computes* $\mathrm{fix}(\Phi_S)$*.*

Correctness of Worklist Algorithm

Theorem 4.12 (Correctness of worklist algorithm)

Given a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$*, Algorithm 4.9 always terminates and computes* $\mathrm{fix}(\Phi_S)$*.*

Proof.

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