Static Program Analysis

Lecture 2: Dataflow Analysis I
(Introduction & Available Expressions/Live Variables Analysis)

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

For p an arbitrary probability:

```
bool c := true;
int i : = 0;
while (c) {
    i := i + 1;
    (c := false [p] c := true)
}
```

This program almost surely terminates. In finite expected time.

How does this work?

- ✓ Visit lecture on

  Probabilistic Programming

- ✓ When? Intro: October 24, 16:15
- ✓ Where? 9U10 (E3)
- ✓ Who? Prof. Joost-Pieter Katoen.
- ✓ Needed: programming, probability, theory.
- ✓ **Lectures (except intro) in December and January**

Outline of Lecture 2

Preliminaries on Dataflow Analysis

An Example: Available Expressions Analysis

Another Example: Live Variables Analysis

Dataflow Analysis: the Approach

- Traditional form of program analysis

Dataflow Analysis: the Approach

- Traditional form of program analysis
- Idea: describe how analysis information flows through program

Dataflow Analysis: the Approach

- Traditional form of program analysis
- Idea: describe how analysis information flows through program
- Distinctions:

dependence on statement order:
: flow-sensitive vs. flow-insensitive analyses

direction of flow:
: forward vs. backward analyses

quantification over paths:
: may (union) vs. must (intersection) analyses

procedures:
: interprocedural vs. intraprocedural analyses

Labelled Programs

- Goal: localisation of analysis information

Labelled Programs

- Goal: localisation of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)

Labelled Programs

- Goal: localisation of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)
- Assume set of labels Lab with meta variable $l \in Lab$ (usually $Lab = \mathbb{N}$)

Labelled Programs

- Goal: localisation of analysis information
- Dataflow information will be associated with
  - `skip` statements
  - assignments
  - tests in conditionals (`if`) and loops (`while`)
- Assume set of labels Lab with meta variable $l \in Lab$ (usually $Lab = \mathbb{N}$)

Definition 2.1 (Labelled WHILE programs)

The syntax of labelled WHILE programs is defined by the following context-free grammar:

$$
\begin{aligned}
a &::= z \mid x \mid a_1{+}a_2 \mid a_1{-}a_2 \mid a_1{*}a_2 \in AExp \\
b &::= t \mid a_1{=}a_2 \mid a_1{>}a_2 \mid \neg b \mid b_1 \wedge b_2 \mid b_1 \vee b_2 \in BExp \\
c &::= [\texttt{skip}]^l \mid [x := a]^l \mid c_1 ; c_2 \mid \\
&\qquad \texttt{if } [b]^l \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end} \mid \texttt{while } [b]^l \texttt{ do } c \texttt{ end} \in Cmd
\end{aligned}
$$

- All labels in $c \in Cmd$ assumed distinct, denoted by Lab_c
- Labelled fragments of c called blocks, denoted by Blk_c

A WHILE Program

Example 2.2

```
x := 6;
y := 7;
z := 0;
while x > 0 do
  x := x - 1;
  v := y;
  while v > 0 do
    v := v - 1;
    z := z + 1
  end
end
```


Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

A WHILE Program with Labels

Example 2.2

$$
\begin{array}{l}
[\mathtt{x}\ \mathtt{:=}\ \mathtt{6}]^1;\\
[\mathtt{y}\ \mathtt{:=}\ \mathtt{7}]^2;\\
[\mathtt{z}\ \mathtt{:=}\ \mathtt{0}]^3;\\
\mathtt{while}\ [\mathtt{x}\ \mathtt{>}\ \mathtt{0}]^4\ \mathtt{do}\\
\quad [\mathtt{x}\ \mathtt{:=}\ \mathtt{x}\ \mathtt{-}\ \mathtt{1}]^5;\\
\quad [\mathtt{v}\ \mathtt{:=}\ \mathtt{y}]^6;\\
\quad \mathtt{while}\ [\mathtt{v}\ \mathtt{>}\ \mathtt{0}]^7\ \mathtt{do}\\
\quad\quad [\mathtt{v}\ \mathtt{:=}\ \mathtt{v}\ \mathtt{-}\ \mathtt{1}]^8;\\
\quad\quad [\mathtt{z}\ \mathtt{:=}\ \mathtt{z}\ \mathtt{+}\ \mathtt{1}]^9\\
\quad \mathtt{end}\\
\mathtt{end}
\end{array}
$$

Representing Control Flow I

Every (labelled) statement has a single entry (given by the initial label) and generally multiple exits (given by the final labels):

Definition 2.3 (Initial and final labels)

The mappings

$$\mathrm{init} : \mathit{Cmd} \to \mathit{Lab} \qquad \text{and} \qquad \mathrm{final} : \mathit{Cmd} \to 2^{\mathit{Lab}}$$

respectively return the initial and final label(s) of a statement:

$c \in \mathit{Cmd}$	$\mathrm{init}(c)$	$\mathrm{final}(c)$
$[\texttt{skip}]^l$	l	$\{l\}$
$[x := a]^l$	l	$\{l\}$
$c_1 ; c_2$	$\mathrm{init}(c_1)$	$\mathrm{final}(c_2)$
$\texttt{if } [b]^l \texttt{ then } c_1 \texttt{ else } c_2 \texttt{ end}$	l	$\mathrm{final}(c_1) \cup \mathrm{final}(c_2)$
$\texttt{while } [b]^l \texttt{ do } c \texttt{ end}$	l	$\{l\}$

Representing Control Flow II

Definition 2.4 (Flow relation)

Given a statement $c \in Cmd$, the (control) flow relation

$$\mathrm{flow}(c) \subseteq Lab \times Lab$$

is defined by

$$\begin{aligned}
\mathrm{flow}([\mathtt{skip}]^l) &:= \emptyset \\
\mathrm{flow}([x := a]^l) &:= \emptyset \\
\mathrm{flow}(c_1\, ;\, c_2) &:= \mathrm{flow}(c_1) \cup \mathrm{flow}(c_2) \cup \mathrm{final}(c_1) \times \{\mathrm{init}(c_2)\} \\
\mathrm{flow}(\mathtt{if}\ [b]^l\ \mathtt{then}\ c_1\ \mathtt{else}\ c_2\ \mathtt{end}) &:= \mathrm{flow}(c_1) \cup \mathrm{flow}(c_2) \cup \{(l, \mathrm{init}(c_1)), (l, \mathrm{init}(c_2))\} \\
\mathrm{flow}(\mathtt{while}\ [b]^l\ \mathtt{do}\ c\ \mathtt{end}) &:= \mathrm{flow}(c) \cup \{(l, \mathrm{init}(c))\} \cup \mathrm{final}(c) \times \{l\}
\end{aligned}$$

Representing Control Flow III

Example 2.5

$c =$ `[z := 1]`^{1}`;`

```
while [x > 0]^2 do
  [z := z*y]^3;
  [x := x-1]^4
end
```

Representing Control Flow III

Example 2.5

$c = [\texttt{z := 1}]^1;$
$\quad \texttt{while } [\texttt{x > 0}]^2 \texttt{ do}$
$\qquad [\texttt{z := z*y}]^3;$
$\qquad [\texttt{x := x-1}]^4$
$\quad \texttt{end}$

$$\mathsf{init}(c) = 1$$
$$\mathsf{final}(c) = \{2\}$$
$$\mathsf{flow}(c) = \{(1,2),(2,3),(3,4),(4,2)\}$$

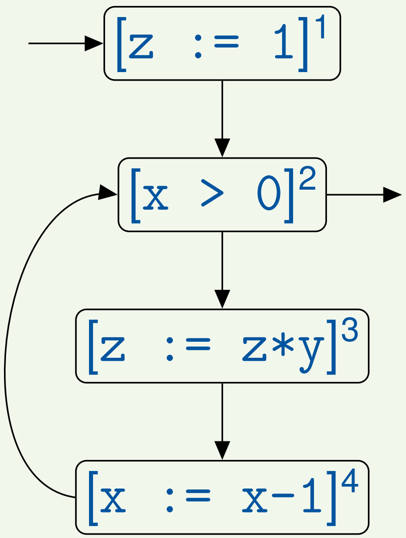
Representing Control Flow III

Example 2.5

$$\begin{aligned} c = {} & [\texttt{z := 1}]^1;\\ & \texttt{while } [\texttt{x > 0}]^2 \texttt{ do}\\ & \quad [\texttt{z := z*y}]^3;\\ & \quad [\texttt{x := x-1}]^4\\ & \texttt{end} \end{aligned}$$

$$\begin{aligned} \mathsf{init}(c) &= 1\\ \mathsf{final}(c) &= \{2\}\\ \mathsf{flow}(c) &= \{(1,2),(2,3),(3,4),(4,2)\} \end{aligned}$$

Visualisation by (control) flow graph:



Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in \mathit{Cmd}$ under consideration has an isolated entry, meaning that
$$\{l \in \mathit{Lab} \mid (l, \mathrm{init}(c)) \in \mathrm{flow}(c)\} = \emptyset$$
(which is the case when c does not start with a `while` loop)

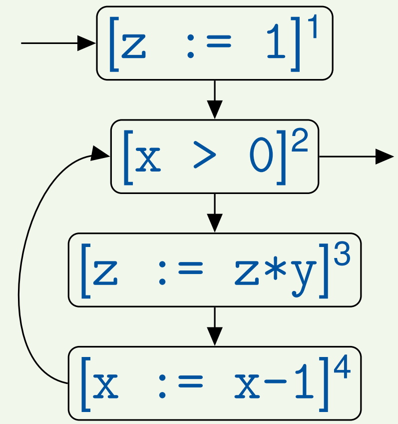
Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in Cmd$ under consideration has an isolated entry, meaning that
$$\{l \in Lab \mid (l, \mathrm{init}(c)) \in \mathrm{flow}(c)\} = \emptyset$$
(which is the case when c does not start with a `while` loop)
- Similarly: $c \in Cmd$ has isolated exits if
$$\{l' \in Lab \mid (l, l') \in \mathrm{flow}(c) \text{ for some } l \in \mathrm{final}(c)\} = \emptyset$$
(which is the case when no final label identifies a loop header)

Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in \mathit{Cmd}$ under consideration has an isolated entry, meaning that
$$\{l \in \mathit{Lab} \mid (l, \mathrm{init}(c)) \in \mathrm{flow}(c)\} = \emptyset$$
(which is the case when c does not start with a `while` loop)
- Similarly: $c \in \mathit{Cmd}$ has isolated exits if
$$\{l' \in \mathit{Lab} \mid (l, l') \in \mathrm{flow}(c) \text{ for some } l \in \mathrm{final}(c)\} = \emptyset$$
(which is the case when no final label identifies a loop header)

Example 2.6 (cf. Ex. 2.5)



has an isolated entry but not isolated exits

Outline of Lecture 2

An Example: Available Expressions Analysis

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

An Example: Available Expressions Analysis

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- Can be used for Common Subexpression Elimination:
  replace subexpression by variable that contains up-to-date value
- Only interesting for non-trivial (i.e., complex) arithmetic expressions

An Example: Available Expressions Analysis

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- Can be used for Common Subexpression Elimination: replace subexpression by variable that contains up-to-date value
- Only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 2.7 (Available Expressions Analysis)

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

An Example: Available Expressions Analysis

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- Can be used for Common Subexpression Elimination: replace subexpression by variable that contains up-to-date value
- Only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 2.7 (Available Expressions Analysis)

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

- `a+b` available at label 3

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- Can be used for Common Subexpression Elimination: replace subexpression by variable that contains up-to-date value
- Only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 2.7 (Available Expressions Analysis)

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

- `a+b` available at label 3
- `a+b` not available at label 5

Goal of Available Expressions Analysis

Available Expressions Analysis

The goal of Available Expressions Analysis is to determine, for each program point, which (complex) expressions *must* have been computed, and not later modified, on all paths to the program point.

- Can be used for Common Subexpression Elimination: replace subexpression by variable that contains up-to-date value
- Only interesting for non-trivial (i.e., complex) arithmetic expressions

Example 2.7 (Available Expressions Analysis)

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

- `a+b` available at label 3
- `a+b` not available at label 5
- possible optimization: $\texttt{while } [\texttt{y > x}]^3 \texttt{ do}$

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $Var_a/Var_b/Var_c$ denotes the set of all variables occurring in $a/b/c$
  - $CExp_b/CExp_c$ denote the sets of all complex arithmetic expressions occurring in b/c

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $Var_a/Var_b/Var_c$ denotes the set of all variables occurring in $a/b/c$
  - $CExp_b/CExp_c$ denote the sets of all complex arithmetic expressions occurring in b/c
- An expression a is killed in a block B if any of the variables in a is modified in B

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $Var_a/Var_b/Var_c$ denotes the set of all variables occurring in $a/b/c$
  - $CExp_b/CExp_c$ denote the sets of all complex arithmetic expressions occurring in b/c
- An expression a is killed in a block B if any of the variables in a is modified in B
- Formally: $\mathsf{kill}_{\mathsf{AE}} : Blk_c \to 2^{CExp_c}$ is defined by

$$\begin{aligned}\mathsf{kill}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{kill}_{\mathsf{AE}}([x := a]^l) &:= \{a' \in CExp_c \mid x \in Var_{a'}\} \\ \mathsf{kill}_{\mathsf{AE}}([b]^l) &:= \emptyset\end{aligned}$$

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $Var_a/Var_b/Var_c$ denotes the set of all variables occurring in $a/b/c$
  - $CExp_b/CExp_c$ denote the sets of all complex arithmetic expressions occurring in b/c
- An expression a is killed in a block B if any of the variables in a is modified in B
- Formally: $\mathsf{kill}_{\mathsf{AE}} : Blk_c \to 2^{CExp_c}$ is defined by

$$\begin{aligned}\mathsf{kill}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset\\ \mathsf{kill}_{\mathsf{AE}}([x := a]^l) &:= \{a' \in CExp_c \mid x \in Var_{a'}\}\\ \mathsf{kill}_{\mathsf{AE}}([b]^l) &:= \emptyset\end{aligned}$$

- An expression a is generated in a block B if it is evaluated in and none of its variables are modified by B

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $Var_a/Var_b/Var_c$ denotes the set of all variables occurring in $a/b/c$
  - $CExp_b/CExp_c$ denote the sets of all complex arithmetic expressions occurring in b/c
- An expression a is killed in a block B if any of the variables in a is modified in B
- Formally: $\mathsf{kill}_{\mathsf{AE}} : Blk_c \to 2^{CExp_c}$ is defined by

$$\begin{aligned}
\mathsf{kill}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset\\
\mathsf{kill}_{\mathsf{AE}}([x := a]^l) &:= \{a' \in CExp_c \mid x \in Var_{a'}\}\\
\mathsf{kill}_{\mathsf{AE}}([b]^l) &:= \emptyset
\end{aligned}$$

- An expression a is generated in a block B if it is evaluated in and none of its variables are modified by B
- Formally: $\mathsf{gen}_{\mathsf{AE}} : Blk_c \to 2^{CExp_c}$ is defined by

$$\begin{aligned}
\mathsf{gen}_{\mathsf{AE}}([\mathtt{skip}]^l) &:= \emptyset\\
\mathsf{gen}_{\mathsf{AE}}([x := a]^l) &:= \{a \mid x \notin Var_a\}\\
\mathsf{gen}_{\mathsf{AE}}([b]^l) &:= CExp_b
\end{aligned}$$

Formalizing Available Expressions Analysis II

Example 2.8 ($\mathsf{kill}_{\mathsf{AE}}/\mathsf{gen}_{\mathsf{AE}}$ functions)

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

Formalizing Available Expressions Analysis II

Example 2.8 ($\mathsf{kill}_{\mathsf{AE}}/\mathsf{gen}_{\mathsf{AE}}$ functions)

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

- $CExp_c = \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$

Formalizing Available Expressions Analysis II

Example 2.8 (kill_{AE}/gen_{AE} functions)

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

- $\mathit{CExp}_c = \{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$
-

Lab_c	$\mathsf{kill}_{AE}(B^l)$	$\mathsf{gen}_{AE}(B^l)$
1	$\emptyset$	$\{\mathtt{a+b}\}$
2	$\emptyset$	$\{\mathtt{a*b}\}$
3	$\emptyset$	$\{\mathtt{a+b}\}$
4	$\{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\mathtt{a+b}\}$

An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system

An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_c$, $\mathrm{AE}_l \subseteq CExp_c$ represents the set of available expressions at the entry of block B^l

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_c$, $\mathrm{AE}_l \subseteq CExp_c$ represents the set of available expressions at the entry of block B^l
- Formally, for $c \in Cmd$ with isolated entry:

$$\mathrm{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathrm{init}(c) \\ \bigcap\{\varphi_{l'}(\mathrm{AE}_{l'}) \mid (l', l) \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$

where $\varphi_{l'} : 2^{CExp_c} \to 2^{CExp_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \mathrm{kill}_{\mathrm{AE}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{AE}}(B^{l'})$$

An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_c$, $\mathrm{AE}_l \subseteq CExp_c$ represents the set of available expressions at the entry of block B^l
- Formally, for $c \in Cmd$ with isolated entry:

$$\mathrm{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathrm{init}(c) \\ \bigcap\{\varphi_{l'}(\mathrm{AE}_{l'}) \mid (l', l) \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$

  where $\varphi_{l'} : 2^{CExp_c} \to 2^{CExp_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \mathrm{kill}_{\mathrm{AE}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{AE}}(B^{l'})$$

- Characterization of analysis:

  flow-sensitive: results depending on order of assignments

  forward: starts in $\mathrm{init}(c)$ and proceeds downwards

  must: $\bigcap$ in equations for AE_l

An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_c$, $\mathrm{AE}_l \subseteq CExp_c$ represents the set of available expressions at the entry of block B^l
- Formally, for $c \in Cmd$ with isolated entry:

$$\mathrm{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathrm{init}(c) \\ \bigcap\{\varphi_{l'}(\mathrm{AE}_{l'}) \mid (l', l) \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$

  where $\varphi_{l'} : 2^{CExp_c} \to 2^{CExp_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \mathrm{kill}_{\mathrm{AE}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{AE}}(B^{l'})$$

- Characterization of analysis:

  flow-sensitive: results depending on order of assignments

  forward: starts in $\mathrm{init}(c)$ and proceeds downwards

  must: $\bigcap$ in equations for AE_l
- Later: solution not necessarily unique

  $\Longrightarrow$ choose greatest one

The Equation System II

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

The Equation System II

Reminder:

$$\mathrm{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathrm{init}(c) \\ \bigcap\{\varphi_{l'}(\mathrm{AE}_{l'}) \mid (l', l) \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathrm{kill_{AE}}(B^{l'})) \cup \mathrm{gen_{AE}}(B^{l'})$$

Example 2.9 (AE equation system)

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

An Example: Available Expressions Analysis

The Equation System II

Reminder:

$$\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap \{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$$

Example 2.9 (AE equation system)

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

$l \in Lab_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\texttt{a+b}\}$
2	$\emptyset$	$\{\texttt{a*b}\}$
3	$\emptyset$	$\{\texttt{a+b}\}$
4	$\{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\texttt{a+b}\}$

An Example: Available Expressions Analysis

The Equation System II

Reminder: $\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$

$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$

Example 2.9 (AE equation system)

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

$l \in Lab_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\mathtt{a+b}\}$
2	$\emptyset$	$\{\mathtt{a*b}\}$
3	$\emptyset$	$\{\mathtt{a+b}\}$
4	$\{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\mathtt{a+b}\}$

Equations:

$$
\begin{aligned}
\mathsf{AE}_1 &= \emptyset\\
\mathsf{AE}_2 &= \varphi_1(\mathsf{AE}_1) = \mathsf{AE}_1 \cup \{\mathtt{a+b}\}\\
\mathsf{AE}_3 &= \varphi_2(\mathsf{AE}_2) \cap \varphi_5(\mathsf{AE}_5)\\
&= (\mathsf{AE}_2 \cup \{\mathtt{a*b}\}) \cap (\mathsf{AE}_5 \cup \{\mathtt{a+b}\})\\
\mathsf{AE}_4 &= \varphi_3(\mathsf{AE}_3) = \mathsf{AE}_3 \cup \{\mathtt{a+b}\}\\
\mathsf{AE}_5 &= \varphi_4(\mathsf{AE}_4) = \mathsf{AE}_4 \setminus \{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}
\end{aligned}
$$

The Equation System II

Reminder: $\mathsf{AE}_l = \begin{cases} \emptyset & \text{if } l = \mathsf{init}(c) \\ \bigcap\{\varphi_{l'}(\mathsf{AE}_{l'}) \mid (l', l) \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$

$\varphi_{l'}(E) = (E \setminus \mathsf{kill}_{\mathsf{AE}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{AE}}(B^{l'})$

Example 2.9 (AE equation system)

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

$l \in Lab_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\mathtt{a+b}\}$
2	$\emptyset$	$\{\mathtt{a*b}\}$
3	$\emptyset$	$\{\mathtt{a+b}\}$
4	$\{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\mathtt{a+b}\}$

Equations:

$$\begin{aligned} \mathsf{AE}_1 &= \emptyset \\ \mathsf{AE}_2 &= \varphi_1(\mathsf{AE}_1) = \mathsf{AE}_1 \cup \{\mathtt{a+b}\} \\ \mathsf{AE}_3 &= \varphi_2(\mathsf{AE}_2) \cap \varphi_5(\mathsf{AE}_5) \\ &= (\mathsf{AE}_2 \cup \{\mathtt{a*b}\}) \cap (\mathsf{AE}_5 \cup \{\mathtt{a+b}\}) \\ \mathsf{AE}_4 &= \varphi_3(\mathsf{AE}_3) = \mathsf{AE}_3 \cup \{\mathtt{a+b}\} \\ \mathsf{AE}_5 &= \varphi_4(\mathsf{AE}_4) = \mathsf{AE}_4 \setminus \{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\} \end{aligned}$$

(Unique) solution:

$$\begin{aligned} \mathsf{AE}_1 &= \emptyset \\ \mathsf{AE}_2 &= \{\mathtt{a+b}\} \\ \mathsf{AE}_3 &= \{\mathtt{a+b}\} \\ \mathsf{AE}_4 &= \{\mathtt{a+b}\} \\ \mathsf{AE}_5 &= \emptyset \end{aligned}$$

Outline of Lecture 2

Another Example: Live Variables Analysis

Goal of Live Variables Analysis

Live Variables Analysis

The goal of Live Variables Analysis is to determine, for each program point, which variables *may* be live at the exit from the point.

Goal of Live Variables Analysis

Live Variables Analysis

The goal of Live Variables Analysis is to determine, for each program point, which variables *may* be live at the exit from the point.

- A variable is called live at the exit from a block if there exists a path from the block to a use of the variable that does not re-define the variable

Another Example: Live Variables Analysis

Goal of Live Variables Analysis

Live Variables Analysis

The goal of Live Variables Analysis is to determine, for each program point, which variables *may* be live at the exit from the point.

- A variable is called live at the exit from a block if there exists a path from the block to a use of the variable that does not re-define the variable
- All variables considered to be live at the end of the program (alternative: restriction to output variables)

Another Example: Live Variables Analysis

Goal of Live Variables Analysis

Live Variables Analysis

The goal of Live Variables Analysis is to determine, for each program point, which variables *may* be live at the exit from the point.

- A variable is called live at the exit from a block if there exists a path from the block to a use of the variable that does not re-define the variable
- All variables considered to be live at the end of the program (alternative: restriction to output variables)
- Can be used for Dead Code Elimination: remove assignments to non-live variables

Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

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

An Example

Example 2.10 (Live Variables Analysis)

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

- x not live at exit from label 1

An Example

Example 2.10 (Live Variables Analysis)

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

- x not live at exit from label 1
- y live at exit from 2

Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

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

- x not live at exit from label 1
- y live at exit from 2
- x live at exit from 3

Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

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

- x not live at exit from label 1
- y live at exit from 2
- x live at exit from 3
- z live at exits from 5 and 6

Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

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

- x not live at exit from label 1
- y live at exit from 2
- x live at exit from 3
- z live at exits from 5 and 6
- possible optimization: remove $[\texttt{x := 2}]^1$

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests and `skip` do not kill

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests and `skip` do not kill
- Formally: $\mathsf{kill}_{\mathsf{LV}} : Blk_c \to 2^{Var_c}$ is defined by

$$\begin{aligned}
\mathsf{kill}_{\mathsf{LV}}([\mathtt{skip}]^l) &:= \emptyset \\
\mathsf{kill}_{\mathsf{LV}}([x := a]^l) &:= \{x\} \\
\mathsf{kill}_{\mathsf{LV}}([b]^l) &:= \emptyset
\end{aligned}$$

Another Example: Live Variables Analysis

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests and `skip` do not kill
- Formally: $\mathsf{kill}_{\mathsf{LV}} : Blk_c \to 2^{Var_c}$ is defined by

$$\begin{aligned} \mathsf{kill}_{\mathsf{LV}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{kill}_{\mathsf{LV}}([x := a]^l) &:= \{x\} \\ \mathsf{kill}_{\mathsf{LV}}([b]^l) &:= \emptyset \end{aligned}$$

- Every reading access generates a live variable

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests and `skip` do not kill
- Formally: $\mathsf{kill}_{\mathrm{LV}} : Blk_c \to 2^{Var_c}$ is defined by

$$\begin{aligned}\mathsf{kill}_{\mathrm{LV}}([\mathtt{skip}]^l) &:= \emptyset\\ \mathsf{kill}_{\mathrm{LV}}([x := a]^l) &:= \{x\}\\ \mathsf{kill}_{\mathrm{LV}}([b]^l) &:= \emptyset\end{aligned}$$

- Every reading access generates a live variable
- Formally: $\mathsf{gen}_{\mathrm{LV}} : Blk_c \to 2^{Var_c}$ is defined by

$$\begin{aligned}\mathsf{gen}_{\mathrm{LV}}([\mathtt{skip}]^l) &:= \emptyset\\ \mathsf{gen}_{\mathrm{LV}}([x := a]^l) &:= Var_a\\ \mathsf{gen}_{\mathrm{LV}}([b]^l) &:= Var_b\end{aligned}$$

Formalizing Live Variables Analysis II

Example 2.11 ($\mathsf{kill}_{\mathsf{LV}}/\mathsf{gen}_{\mathsf{LV}}$ functions)

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

Formalizing Live Variables Analysis II

Example 2.11 ($kill_{LV}$/gen_{LV} functions)

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

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

Formalizing Live Variables Analysis II

Example 2.11 ($\mathsf{kill}_{\mathrm{LV}}$/$\mathsf{gen}_{\mathrm{LV}}$ functions)

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

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

$l \in Lab_c$	$\mathsf{kill}_{\mathrm{LV}}(B^l)$	$\mathsf{gen}_{\mathrm{LV}}(B^l)$
1	$\{\mathtt{x}\}$	$\emptyset$
2	$\{\mathtt{y}\}$	$\emptyset$
3	$\{\mathtt{x}\}$	$\emptyset$
4	$\emptyset$	$\{\mathtt{y}\}$
5	$\{\mathtt{z}\}$	$\{\mathtt{x}\}$
6	$\{\mathtt{z}\}$	$\{\mathtt{y}\}$
7	$\{\mathtt{x}\}$	$\{\mathtt{z}\}$

Another Example: Live Variables Analysis

The Equation System I

- For each $l \in Lab_c$, $\mathrm{LV}_l \subseteq Var_c$ represents the set of live variables at the exit of block B^l

Another Example: Live Variables Analysis

The Equation System I

- For each $l \in Lab_c$, $\mathrm{LV}_l \subseteq Var_c$ represents the set of live variables at the exit of block B^l
- Formally, for a program $c \in Cmd$ with isolated exits:
$$\mathrm{LV}_l = \begin{cases} Var_c & \text{if } l \in \mathrm{final}(c) \\ \bigcup\{\varphi_{l'}(\mathrm{LV}_{l'}) \mid (l, l') \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$
where $\varphi_{l'} : 2^{Var_c} \to 2^{Var_c}$ denotes the transfer function of block $B^{l'}$, given by
$$\varphi_{l'}(V) := (V \setminus \mathrm{kill}_{\mathrm{LV}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{LV}}(B^{l'})$$

Another Example: Live Variables Analysis

The Equation System I

- For each $l \in Lab_c$, $\mathsf{LV}_l \subseteq Var_c$ represents the set of live variables at the exit of block B^l
- Formally, for a program $c \in Cmd$ with isolated exits:

$$\mathsf{LV}_l = \begin{cases} Var_c & \text{if } l \in \mathsf{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l, l') \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$

  where $\varphi_{l'} : 2^{Var_c} \to 2^{Var_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(V) := (V \setminus \mathsf{kill}_{\mathsf{LV}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{LV}}(B^{l'})$$

- Characterization of analysis:

  flow-sensitive: results depending on order of assignments

  backward: starts in $\mathsf{final}(c)$ and proceeds upwards

  may: $\bigcup$ in equations for LV_l

Another Example: Live Variables Analysis

The Equation System I

- For each $l \in Lab_c$, $\mathrm{LV}_l \subseteq Var_c$ represents the set of live variables at the exit of block B^l
- Formally, for a program $c \in Cmd$ with isolated exits:
$$\mathrm{LV}_l = \begin{cases} Var_c & \text{if } l \in \mathrm{final}(c) \\ \bigcup\{\varphi_{l'}(\mathrm{LV}_{l'}) \mid (l, l') \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$
where $\varphi_{l'} : 2^{Var_c} \to 2^{Var_c}$ denotes the transfer function of block $B^{l'}$, given by
$$\varphi_{l'}(V) := (V \setminus \mathrm{kill}_{\mathrm{LV}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{LV}}(B^{l'})$$
- Characterization of analysis:
  flow-sensitive: results depending on order of assignments
  backward: starts in $\mathrm{final}(c)$ and proceeds upwards
  may: $\bigcup$ in equations for LV_l
- Later: solution not necessarily unique
  $\Longrightarrow$ choose least one

Another Example: Live Variables Analysis

The Equation System II

Reminder:

$$\mathrm{LV}_l = \begin{cases} \mathit{Var}_c & \text{if } l \in \mathrm{final}(c) \\ \bigcup\{\varphi_{l'}(\mathrm{LV}_{l'}) \mid (l, l') \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$

$$\varphi_{l'}(V) = (V \setminus \mathrm{kill}_{\mathrm{LV}}(B^{l'})) \cup \mathrm{gen}_{\mathrm{LV}}(B^{l'})$$

The Equation System II

Reminder: $\mathsf{LV}_l = \begin{cases} Var_c & \text{if } l \in \mathsf{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l, l') \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$

$\varphi_{l'}(V) = (V \setminus \mathsf{kill}_{\mathsf{LV}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{LV}}(B^{l'})$

Example 2.12 (LV equation system)

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

The Equation System II

Reminder: $$\mathsf{LV}_l = \begin{cases} \mathit{Var}_c & \text{if } l \in \mathsf{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l,l') \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$
$$\varphi_{l'}(V) = (V \setminus \mathsf{kill}_{\mathsf{LV}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{LV}}(B^{l'})$$

Example 2.12 (LV equation system)

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

$l \in \mathit{Lab}_c$	$\mathsf{kill}_{\mathsf{LV}}(B^l)$	$\mathsf{gen}_{\mathsf{LV}}(B^l)$
1	$\{x\}$	$\emptyset$
2	$\{y\}$	$\emptyset$
3	$\{x\}$	$\emptyset$
4	$\emptyset$	$\{y\}$
5	$\{z\}$	$\{x\}$
6	$\{z\}$	$\{y\}$
7	$\{x\}$	$\{z\}$

Another Example: Live Variables Analysis

The Equation System II

Reminder:
$$\mathsf{LV}_l = \begin{cases} Var_c & \text{if } l \in \mathrm{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l, l') \in \mathrm{flow}(c)\} & \text{otherwise} \end{cases}$$
$$\varphi_{l'}(V) = (V \setminus \mathrm{kill}_{\mathsf{LV}}(B^{l'})) \cup \mathrm{gen}_{\mathsf{LV}}(B^{l'})$$

Example 2.12 (LV equation system)

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

$l \in Lab_c$	$\mathrm{kill}_{\mathsf{LV}}(B^l)$	$\mathrm{gen}_{\mathsf{LV}}(B^l)$
1	$\{\mathtt{x}\}$	$\emptyset$
2	$\{\mathtt{y}\}$	$\emptyset$
3	$\{\mathtt{x}\}$	$\emptyset$
4	$\emptyset$	$\{\mathtt{y}\}$
5	$\{\mathtt{z}\}$	$\{\mathtt{x}\}$
6	$\{\mathtt{z}\}$	$\{\mathtt{y}\}$
7	$\{\mathtt{x}\}$	$\{\mathtt{z}\}$

Equations:
$$\begin{aligned}
\mathsf{LV}_1 &= \varphi_2(\mathsf{LV}_2) = \mathsf{LV}_2 \setminus \{\mathtt{y}\} \\
\mathsf{LV}_2 &= \varphi_3(\mathsf{LV}_3) = \mathsf{LV}_3 \setminus \{\mathtt{x}\} \\
\mathsf{LV}_3 &= \varphi_4(\mathsf{LV}_4) = \mathsf{LV}_4 \cup \{\mathtt{y}\} \\
\mathsf{LV}_4 &= \varphi_5(\mathsf{LV}_5) \cup \varphi_6(\mathsf{LV}_6) \\
&= ((\mathsf{LV}_5 \setminus \{\mathtt{z}\}) \cup \{\mathtt{x}\}) \cup ((\mathsf{LV}_6 \setminus \{\mathtt{z}\}) \cup \{\mathtt{y}\}) \\
\mathsf{LV}_5 &= \varphi_7(\mathsf{LV}_7) = (\mathsf{LV}_7 \setminus \{\mathtt{x}\}) \cup \{\mathtt{z}\} \\
\mathsf{LV}_6 &= \varphi_7(\mathsf{LV}_7) = (\mathsf{LV}_7 \setminus \{\mathtt{x}\}) \cup \{\mathtt{z}\} \\
\mathsf{LV}_7 &= \{\mathtt{x}, \mathtt{y}, \mathtt{z}\}
\end{aligned}$$

The Equation System II

Reminder: $\mathsf{LV}_l = \begin{cases} \mathit{Var}_c & \text{if } l \in \mathsf{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l, l') \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$

$$\varphi_{l'}(V) = (V \setminus \mathsf{kill}_{\mathsf{LV}}(B^{l'})) \cup \mathsf{gen}_{\mathsf{LV}}(B^{l'})$$

Example 2.12 (LV equation system)

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

$l \in \mathit{Lab}_c$	$\mathsf{kill}_{\mathsf{LV}}(B^l)$	$\mathsf{gen}_{\mathsf{LV}}(B^l)$
1	$\{x\}$	$\emptyset$
2	$\{y\}$	$\emptyset$
3	$\{x\}$	$\emptyset$
4	$\emptyset$	$\{y\}$
5	$\{z\}$	$\{x\}$
6	$\{z\}$	$\{y\}$
7	$\{x\}$	$\{z\}$

Equations:

$$\begin{aligned}
\mathsf{LV}_1 &= \varphi_2(\mathsf{LV}_2) = \mathsf{LV}_2 \setminus \{y\} \\
\mathsf{LV}_2 &= \varphi_3(\mathsf{LV}_3) = \mathsf{LV}_3 \setminus \{x\} \\
\mathsf{LV}_3 &= \varphi_4(\mathsf{LV}_4) = \mathsf{LV}_4 \cup \{y\} \\
\mathsf{LV}_4 &= \varphi_5(\mathsf{LV}_5) \cup \varphi_6(\mathsf{LV}_6) \\
&= ((\mathsf{LV}_5 \setminus \{z\}) \cup \{x\}) \cup ((\mathsf{LV}_6 \setminus \{z\}) \cup \{y\}) \\
\mathsf{LV}_5 &= \varphi_7(\mathsf{LV}_7) = (\mathsf{LV}_7 \setminus \{x\}) \cup \{z\} \\
\mathsf{LV}_6 &= \varphi_7(\mathsf{LV}_7) = (\mathsf{LV}_7 \setminus \{x\}) \cup \{z\} \\
\mathsf{LV}_7 &= \{x, y, z\}
\end{aligned}$$

(Unique) solution:

$$\begin{aligned}
\mathsf{LV}_1 &= \emptyset \\
\mathsf{LV}_2 &= \{y\} \\
\mathsf{LV}_3 &= \{x, y\} \\
\mathsf{LV}_4 &= \{x, y\} \\
\mathsf{LV}_5 &= \{y, z\} \\
\mathsf{LV}_6 &= \{y, z\} \\
\mathsf{LV}_7 &= \{x, y, z\}
\end{aligned}$$