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/
For $p$ an arbitrary probability:

```plaintext
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)


✓ 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
Preliminaries on Dataflow Analysis

Dataflow Analysis: the Approach

- Traditional form of program analysis
Preliminaries on Dataflow 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
Preliminaries on Dataflow Analysis

Labelled Programs

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

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 $Lab = \mathbb{N}$)
Preliminaries on Dataflow Analysis

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:

- $a ::= z \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 * a_2 \in AExp$
- $b ::= t \mid a_1 = a_2 \mid a_1 > a_2 \mid \neg b \mid b_1 \land b_2 \mid b_1 \lor b_2 \in BExp$
- $c ::= [skip]' \mid [x := a]' \mid c_1 ; c_2 \mid$
  - if $[b]'$ then $c_1$ else $c_2$ end \mid while $[b]'$ do $c$ end $\in Cmd$
- All labels in $c \in Cmd$ assumed distinct, denoted by $Lab_c$
- Labelled fragments of $c$ called *blocks*, denoted by $Blk_c$
Preliminaries on Dataflow Analysis

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
Preliminaries on Dataflow Analysis

A WHILE Program with Labels

Example 2.2

\[
\begin{align*}
x & := 6^1; \\
y & := 7^2; \\
z & := 0^3; \\
\text{while } [x > 0]^4 \text{ do} \\
& \quad [x := x - 1]^5; \\
& \quad [v := y]^6; \\
& \quad \text{while } [v > 0]^7 \text{ do} \\
& \quad \quad [v := v - 1]^8; \\
& \quad \quad [z := z + 1]^9 \\
& \quad \text{end} \\
& \text{end}
\end{align*}
\]
Preliminaries on Dataflow Analysis

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

\[
\text{init} : \text{Cmd} \rightarrow \text{Lab} \quad \text{and} \quad \text{final} : \text{Cmd} \rightarrow 2^{\text{Lab}}
\]

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

<table>
<thead>
<tr>
<th>$c \in \text{Cmd}$</th>
<th>init$(c)$</th>
<th>final$(c)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>[skip]'</td>
<td>$/$</td>
<td>${/}$</td>
</tr>
<tr>
<td>[$x := a]'</td>
<td>$/$</td>
<td>${/}$</td>
</tr>
<tr>
<td>$c_1; c_2$</td>
<td>init$(c_1)$</td>
<td>final$(c_2)$</td>
</tr>
<tr>
<td>if $[b]'$ then $c_1$ else $c_2$ end</td>
<td>$/$</td>
<td>final$(c_1) \cup$ final$(c_2)$</td>
</tr>
<tr>
<td>while $[b]'$ do $c$ end</td>
<td>$/$</td>
<td>${/}$</td>
</tr>
</tbody>
</table>
Preliminaries on Dataflow Analysis

Representing Control Flow II

**Definition 2.4 (Flow relation)**

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

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

is defined by

- $\text{flow}([\text{skip}]) := \emptyset$
- $\text{flow}([x := a]) := \emptyset$
- $\text{flow}(c_1 ; c_2) := \text{flow}(c_1) \cup \text{flow}(c_2) \cup \text{final}(c_1) \times \{\text{init}(c_2)\}$
- $\text{flow}(\text{if } [b] \text{ then } c_1 \text{ else } c_2 \text{ end}) := \text{flow}(c_1) \cup \text{flow}(c_2) \cup \{(l, \text{init}(c_1)), (l, \text{init}(c_2))\}$
- $\text{flow}(\text{while } [b] \text{ do } c \text{ end}) := \text{flow}(c) \cup \{(l, \text{init}(c))\} \cup \text{final}(c) \times \{l\}$
Example 2.5

\[ c = [z := 1]^{1}; \]
\[ \text{while } [x > 0]^{2} \text{ do} \]
\[ [z := z*y]^{3}; \]
\[ [x := x-1]^{4} \]
end
Preliminaries on Dataflow Analysis

Representing Control Flow III

Example 2.5

c = [z := 1];
  while [x > 0] do
    [z := z*y];
    [x := x-1]
  end

init(c) = 1
final(c) = {2}
flow(c) = {(1, 2), (2, 3), (3, 4), (4, 2)}
Example 2.5

\[ c = [z := 1] \]
\[ \text{while} [x > 0] \text{do} \]
\[ [z := z*y] ; \]
\[ [x := x-1] \]
\[ \text{end} \]

\[ \text{init}(c) = 1 \]
\[ \text{final}(c) = \{2\} \]
\[ \text{flow}(c) = \{(1, 2), (2, 3), (3, 4), (4, 2)\} \]
Preliminaries on Dataflow Analysis

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
  $$\{l \in Lab \mid (l, \text{init}(c)) \in \text{flow}(c)\} = \emptyset$$
  (which is the case when $c$ does not start with a while loop)
Preliminaries on Dataflow Analysis

Representing Control Flow IV

• To simplify the presentation we will often assume that the program \( c \in \text{Cmd} \) under consideration has an isolated entry, meaning that

\[
\{ l \in \text{Lab} \mid (l, \text{init}(c)) \in \text{flow}(c) \} = \emptyset
\]

(which is the case when \( c \) does not start with a \texttt{while} loop)

• Similarly: \( c \in \text{Cmd} \) has isolated exits if

\[
\{ l' \in \text{Lab} \mid (l, l') \in \text{flow}(c) \text{ for some } l \in \text{final}(c) \} = \emptyset
\]

(which is the case when no final label identifies a loop header)
Preliminaries on Dataflow Analysis

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
  \[ \{ l \in Lab \mid (l, \text{init}(c)) \in \text{flow}(c) \} = \emptyset \]
  (which is the case when \( c \) does not start with a while loop)

- Similarly: \( c \in Cmd \) has isolated exits if
  \[ \{ l' \in Lab \mid (l, l') \in \text{flow}(c) \text{ for some } l \in \text{final}(c) \} = \emptyset \]
  (which is the case when no final label identifies a loop header)

Example 2.6 (cf. Ex. 2.5)

\[
\begin{align*}
&[z := 1]^1 \\
&[x > 0]^2 \\
&[z := z*y]^3 \\
&[x := x-1]^4
\end{align*}
\]

has an isolated entry but not isolated exits
An Example: Available Expressions Analysis

Outline of Lecture 2

Preliminaries on Dataflow Analysis

An Example: Available Expressions Analysis

Another Example: Live Variables Analysis
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

Example 2.7 (Available Expressions Analysis)

\[
x := a+b; \\
y := a*b; \\
\text{while } y > a+b \text{ do } \\
a := a+1; \\
x := a+b\]

- \(a+b\) available at label 3
- \(a+b\) not available at label 5
- possible optimization: while \(y > x\)
### 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)

```plaintext
[x := a+b]; [y := a*b];
while [y > a+b] do
  [a := a+1];
  [x := a+b];
end
```

- `a+b` available at label 3
- `a+b` not available at label 5
- Possible optimization:
  ```plaintext
  while [y > x] do
  ```
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)

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

- a+b available at label 3
An Example: Available Expressions Analysis

Goal of 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)

\[
\begin{align*}
  &x := a+b; \quad \text{\textsuperscript{1}} \\
  &y := a*b; \quad \text{\textsuperscript{2}} \\
  \text{while } [y > a+b] \quad \text{\textsuperscript{3}} \text{ do} \\
  &a := a+1; \quad \text{\textsuperscript{4}} \\
  &x := a+b \quad \text{\textsuperscript{5}} \end{align*}
\]

- \(a+b\) available at label 3
- \(a+b\) not available at label 5

13 of 24  Static Program Analysis
Winter Semester 2016/17
Lecture 2: Dataflow Analysis I
(Introduction & Available Expressions/Live Variables Analysis)
An Example: Available Expressions Analysis

Goal of 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)

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

- \(a+b\) available at label 3
- \(a+b\) not available at label 5
- possible optimization:
  while \([y > x]^3\) do
An Example: Available Expressions Analysis

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $\text{Var}_a/\text{Var}_b/\text{Var}_c$ denotes the set of all variables occurring in $a/b/c$
  - $\text{CExp}_b/\text{CExp}_c$ denote the sets of all complex arithmetic expressions occurring in $b/c$
An Example: Available Expressions Analysis

Formalizing Available Expressions Analysis I

• Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  
  – $\text{Var}_a/\text{Var}_b/\text{Var}_c$ denotes the set of all variables occurring in $a/b/c$
  
  – $\text{CExp}_b/\text{CExp}_c$ 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 $B$
An Example: Available Expressions Analysis

Formalizing Available Expressions Analysis I

- Given $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $\text{Var}_a/\text{Var}_b/\text{Var}_c$ denotes the set of all variables occurring in $a/b/c$
  - $\text{CExp}_b/\text{CExp}_c$ 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: $\text{kill}_{AE}: Blk_c \rightarrow 2^{\text{CExp}_c}$ is defined by
  $$\text{kill}_{AE}([\text{skip}]) := \emptyset$$
  $$\text{kill}_{AE}([x := a]) := \{ a' \in \text{CExp}_c \mid x \in \text{Var}_a \}$$
  $$\text{kill}_{AE}([b]) := \emptyset$$
An Example: Available Expressions Analysis

Formalizing Available Expressions Analysis I

- Given \( a \in AExp, b \in BExp, c \in Cmd \),
  - \( \text{Var}_a/\text{Var}_b/\text{Var}_c \) denotes the set of all \textit{variables} occurring in \( a/b/c \)
  - \( \text{CExp}_b/\text{CExp}_c \) denote the sets of all \textit{complex arithmetic expressions} occurring in \( b/c \)
- An expression \( a \) is \textit{killed} in a block \( B \) if any of the variables in \( a \) is modified in \( B \)
- Formally: \( \text{kill}_{AE} : B_{\text{lk}} \rightarrow \mathcal{2}_{\text{CExp}_c} \) is defined by
  \[
  \text{kill}_{AE}(\text{[skip]}) := \emptyset \]
  \[
  \text{kill}_{AE}(\text{[x := a]}) := \{ a' \in \text{CExp}_c \mid x \in \text{Var}_a \} \]
  \[
  \text{kill}_{AE}(\text{[b]}) := \emptyset \]
- An expression \( a \) is \textit{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 $a \in AExp$, $b \in BExp$, $c \in Cmd$,
  - $\text{Var}_a/\text{Var}_b/\text{Var}_c$ denotes the set of all variables occurring in $a/b/c$
  - $\text{CExp}_b/\text{CExp}_c$ 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: $\text{kill}_{AE}: Blk_c \to 2^{CExp_c}$ is defined by
  \[
  \text{kill}_{AE}([\text{skip}]) := \emptyset \\
  \text{kill}_{AE}([x := a]) := \{ a' \in CExp_c \mid x \in \text{Var}_a \} \\
  \text{kill}_{AE}([b]) := \emptyset
  \]
- An expression $a$ is generated in a block $B$ if it is evaluated in and none of its variables are modified by $B$
- Formally: $\text{gen}_{AE}: Blk_c \to 2^{CExp_c}$ is defined by
  \[
  \text{gen}_{AE}([\text{skip}]) := \emptyset \\
  \text{gen}_{AE}([x := a]) := \{ a \mid x \notin \text{Var}_a \} \\
  \text{gen}_{AE}([b]) := CExp_b
  \]
An Example: Available Expressions Analysis

Formalizing Available Expressions Analysis II

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

\[
c = [x := a+b];
[y := a*b];
\text{while } [y > a+b] \text{ do}
\quad [a := a+1];
\quad [x := a+b];
\text{end}
\]

\[
\mathcal{C}\text{Exp}_c = \{a+b, a*b, a+1\}
\]

\[
\mathcal{L}\text{ab}_c \text{kill}_{\text{AE}}(B_l) \text{gen}_{\text{AE}}(B_l) = \emptyset \{a+b\} \emptyset \{a*b\} \emptyset \{a+b\} \{a+b, a*b, a+1\} \emptyset \{a+b\}
\]
An Example: Available Expressions Analysis

Formalizing Available Expressions Analysis II

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

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

\[
\cdot \ CExp_c = \{a+b, a*b, a+1\}
\]
Example 2.8 (\text{kill}_{AE}/\text{gen}_{AE} functions)

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

\[ CExp_c = \{a+b, a*b, a+1\} \]
\[ Lab_c \quad \text{kill}_{AE}(B') \quad \text{gen}_{AE}(B') \]
\[ \begin{array}{lll}
1 & \emptyset & \{a+b\} \\
2 & \emptyset & \{a*b\} \\
3 & \emptyset & \{a+b\} \\
4 & \{a+b, a*b, a+1\} & \emptyset \\
5 & \emptyset & \{a+b\} \\
\end{array} \]
An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system

Later: solution not necessarily unique =⇒ choose greatest one
An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in \text{Lab}_c$, $\text{AE}_l \subseteq \text{CExp}_c$ represents the set of available expressions at the entry of block $B^l$
An Example: Available Expressions Analysis

The Equation System I

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

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

where $\varphi_{l'} : 2^{\text{CExp}_c} \rightarrow 2^{\text{CExp}_c}$ denotes the transfer function of block $B^{l'}$, given by

$$\varphi_{l'}(A) := (A \setminus \text{kill}_{\text{AE}}(B^{l'})) \cup \text{gen}_{\text{AE}}(B^{l'})$$
An Example: Available Expressions Analysis

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_c$, $AE_l \subseteq CExp_c$ represents the set of available expressions at the entry of block $B^l$
- Formally, for $c \in Cmd$ with isolated entry:
  $$AE_l = \begin{cases} \emptyset & \text{if } l = \text{init}(c) \\ \bigcap \{ \varphi_{l'}(AE_{l'}) \mid (l', l) \in \text{flow}(c) \} & \text{otherwise} \end{cases}$$
  where $\varphi_{l'} : 2^{CExp_c} \rightarrow 2^{CExp_c}$ denotes the transfer function of block $B^{l'}$, given by
  $$\varphi_{l'}(A) := (A \setminus \text{kill}_{AE}(B^{l'})) \cup \text{gen}_{AE}(B^{l'})$$
- Characterization of analysis:
  - flow-sensitive: results depending on order of assignments
  - forward: starts in $\text{init}(c)$ 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 $l \in Lab_c$, $AE_l \subseteq CExp_c$ represents the set of available expressions at the entry of block $B^l$
- Formally, for $c \in Cmd$ with isolated entry:
  \[
  AE_l = \begin{cases} 
  \emptyset & \text{if } l = \text{init}(c) \\
  \bigcap \{ \varphi_{l'}(AE_{l'}) \mid (l', l) \in \text{flow}(c) \} & \text{otherwise}
  \end{cases}
  \]
  where $\varphi_{l'} : 2^{CExp_c} \rightarrow 2^{CExp_c}$ denotes the transfer function of block $B^{l'}$, given by
  \[
  \varphi_{l'}(A) := (A \setminus \text{kill}_{AE}(B^{l'})) \cup \text{gen}_{AE}(B^{l'})
  \]
- Characterization of analysis:
  flow-sensitive: results depending on order of assignments
  forward: starts in $\text{init}(c)$ and proceeds downwards
  must: $\bigcap$ in equations for $AE_l$
- Later: solution not necessarily unique
  $\implies$ choose greatest one
An Example: Available Expressions Analysis

The Equation System II

Reminder:

\[ AE_l = \begin{cases} \emptyset & \text{if } l = \text{init}(c) \\ \bigcap \{ \varphi_{l'}(AE_{l'}) \mid (l', l) \in \text{flow}(c) \} & \text{otherwise} \end{cases} \]

\[ \varphi_l(E) = (E \setminus \text{kill}_{AE}(B'')) \cup \text{gen}_{AE}(B'') \]
An Example: Available Expressions Analysis

The Equation System II

Reminder:
\[ AE_l = \begin{cases} \emptyset & \text{if } l = \text{init}(c) \\ \bigcap \{ \varphi_{l'}(AE_{l'}) \mid (l', l) \in \text{flow}(c) \} & \text{otherwise} \end{cases} \]
\[ \varphi_{l'}(E) = (E \setminus \text{kill}_{AE}(B'')) \cup \text{gen}_{AE}(B'') \]

Example 2.9 (AE equation system)

\[
c = \begin{cases} [x := a+b] & \text{1} \\ [y := a*b] & \text{2} \\ \text{while} [y > a+b] \text{ do} & \text{3} \\ [a := a+1] & \text{4} \\ [x := a+b] & \text{5} \end{cases}
\end{cases}
\]

\text{end}

17 of 24 Static Program Analysis
Winter Semester 2016/17
Lecture 2: Dataflow Analysis I
(Introduction & Available Expressions/Live Variables Analysis)
An Example: Available Expressions Analysis

The Equation System II

Reminder:

\[ AE_l = \begin{cases} \emptyset & \text{if } l = \text{init}(c) \\ \varphi_r(\text{AE}_{l'}) \cap (l', l) \in \text{flow}(c) & \text{otherwise} \end{cases} \]

\[ \varphi_r(E) = (E \setminus \text{kill}_{AE}(B'')) \cup \text{gen}_{AE}(B'') \]

Example 2.9 (AE equation system)

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

<table>
<thead>
<tr>
<th>l \in Lab_c</th>
<th>\text{kill}_{AE}(B^l)</th>
<th>\text{gen}_{AE}(B^l)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\emptyset</td>
<td>{a+b}</td>
</tr>
<tr>
<td>2</td>
<td>\emptyset</td>
<td>{a*b}</td>
</tr>
<tr>
<td>3</td>
<td>\emptyset</td>
<td>{a+b}</td>
</tr>
<tr>
<td>4</td>
<td>{a+b, a*b, a+1}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>5</td>
<td>\emptyset</td>
<td>{a+b}</td>
</tr>
</tbody>
</table>
An Example: Available Expressions Analysis

The Equation System II

Reminder:

\[ AE_l = \begin{cases} \emptyset & \text{if } l = \text{init}(c) \\ \bigcap \{ \varphi(E) \text{ } | \text{ } (l', l) \in \text{flow}(c) \} & \text{otherwise} \end{cases} \]

\[ \varphi(E) = (E \setminus \text{kill}_{AE}(B'')) \cup \text{gen}_{AE}(B'') \]

Example 2.9 (AE equation system)

\[ c = [x := a+b]; \]
\[ [y := a*b]; \]
\[ \text{while } [y > a+b] \text{ do} \]
\[ [a := a+1]; \]
\[ [x := a+b]; \]
\[ \text{end} \]

Equations:

\[ AE_1 = \emptyset \]
\[ AE_2 = \varphi_1(AE_1) = AE_1 \cup \{a+b\} \]
\[ AE_3 = \varphi_2(AE_2) \cap \varphi_5(AE_5) \]
\[ = (AE_2 \cup \{a*b\}) \cap (AE_5 \cup \{a+b\}) \]
\[ AE_4 = \varphi_3(AE_3) = AE_3 \cup \{a+b\} \]
\[ AE_5 = \varphi_4(AE_4) = AE_4 \setminus \{a+b, a*b, a+1\} \]

<table>
<thead>
<tr>
<th>(l \in Lab_c)</th>
<th>kill(_{AE}(B'))</th>
<th>gen(_{AE}(B'))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>\emptyset</td>
<td>{a+b}</td>
</tr>
<tr>
<td>2</td>
<td>\emptyset</td>
<td>{a*b}</td>
</tr>
<tr>
<td>3</td>
<td>\emptyset</td>
<td>{a+b}</td>
</tr>
<tr>
<td>4</td>
<td>{a+b, a*b, a+1}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>5</td>
<td>\emptyset</td>
<td>{a+b}</td>
</tr>
</tbody>
</table>
### Available Expressions Analysis

#### The Equation System II

**Reminder:**

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

\[ \varphi_{l'}(E) = (E \setminus \text{kill}_{\text{AE}}(B'')) \cup \text{gen}_{\text{AE}}(B'') \]

---

**Example 2.9 (AE equation system)**

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

<table>
<thead>
<tr>
<th>(l \in \text{Lab}_c)</th>
<th>(\text{kill}_{\text{AE}}(B'))</th>
<th>(\text{gen}_{\text{AE}}(B'))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(\emptyset)</td>
<td>{a+b}</td>
</tr>
<tr>
<td>2</td>
<td>(\emptyset)</td>
<td>{a*b}</td>
</tr>
<tr>
<td>3</td>
<td>(\emptyset)</td>
<td>{a+b}</td>
</tr>
<tr>
<td>4</td>
<td>{a+b, a*b, a+1}</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>5</td>
<td>(\emptyset)</td>
<td>{a+b}</td>
</tr>
</tbody>
</table>

**Equations:**

\[
\begin{align*}
  \text{AE}_1 &= \emptyset \\
  \text{AE}_2 &= \varphi_1(\text{AE}_1) = \text{AE}_1 \cup \{a+b\} \\
  \text{AE}_3 &= \varphi_2(\text{AE}_2) \cap \varphi_5(\text{AE}_5) \\
  &\quad = (\text{AE}_2 \cup \{a*b\}) \cap (\text{AE}_5 \cup \{a+b\}) \\
  \text{AE}_4 &= \varphi_3(\text{AE}_3) = \text{AE}_3 \cup \{a+b\} \\
  \text{AE}_5 &= \varphi_4(\text{AE}_4) = \text{AE}_4 \setminus \{a+b, a*b, a+1\}
\end{align*}
\]

(Unique) solution:

\[
\begin{align*}
  \text{AE}_1 &= \emptyset \\
  \text{AE}_2 &= \{a+b\} \\
  \text{AE}_3 &= \{a+b\} \\
  \text{AE}_4 &= \{a+b\} \\
  \text{AE}_5 &= \emptyset
\end{align*}
\]
Another Example: Live Variables Analysis

Outline of Lecture 2

Preliminaries on Dataflow Analysis

An Example: Available Expressions Analysis

Another Example: Live Variables Analysis
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.
Another Example: Live Variables Analysis

Goal of 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

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

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)

\[
\begin{align*}
  x & := 2 \quad \text{1} \\
  y & := 4 \quad \text{2} \\
  x & := 1 \quad \text{3} \\
  \text{if } [y > 0] & \text{ then} \\
  & \quad \text{z := x} \quad \text{5} \\
  \text{else} & \\
  & \quad \text{z := y*y} \quad \text{6} \\
  \text{end;} & \\
  x & := z \quad \text{7}
\end{align*}
\]
Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

\[
\begin{align*}
& x := 2, \\
& y := 4, \\
& x := 1, \\
& \text{if } y > 0 \text{ then} \quad \text{\bullet } x \text{ not live at exit from label 1} \quad [z := x], \\
& \text{else} \quad [z := y \times y], \\
& \text{end;} \quad \text{\bullet } y \text{ live at exit from 2} \quad [x := z],
\end{align*}
\]
Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

\[
\begin{align*}
  x &:= 2 \quad \text{1} \\
  y &:= 4 \quad \text{2} \\
  x &:= 1 \quad \text{3} \\
  \text{if } [y > 0] &\quad \text{4} \\
    z &:= x \quad \text{5} \\
  \text{else} &\quad \text{5} \\
    z &:= y \times y \quad \text{6} \\
  \end{align*}
\]

- \( x \) 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)

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

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

\[
\begin{align*}
  &x := z^7 \\
\end{align*}
\]
Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

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

- \(x\) 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)

\[
\begin{align*}
  &x := 2; \\
  &y := 4; \\
  &x := 1; \\
  &\text{if } y > 0 \text{ then } \quad \begin{align*}
    &z := x
  \end{align*} \quad \text{end;} \\
  &\text{else } \quad \begin{align*}
    &z := y*y
  \end{align*} \quad \text{end;} \\
  &x := z
\end{align*}
\]

- \(x\) 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 \([x := 2]\)

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
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: \(\text{kill}_{LV} : Blk_c \rightarrow 2^{\text{Var}_c}\) is defined by

\[
\begin{align*}
\text{kill}_{LV}([\text{skip}]) & := \emptyset \\
\text{kill}_{LV}([x := a]) & := \{x\} \\
\text{kill}_{LV}([b]) & := \emptyset
\end{align*}
\]
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 \texttt{skip} do not kill

• Formally: $\text{kill}_{LV}: Blk_c \rightarrow 2^{\var_c}$ is defined by

$$\begin{align*}
\text{kill}_{LV}([\text{skip}]') &:= \emptyset \\
\text{kill}_{LV}([x := a]') &:= \{x\} \\
\text{kill}_{LV}([b]') &:= \emptyset
\end{align*}$$

• Every reading access generates a live variable
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: $\text{kill}_{LV} : \text{Blk}_c \rightarrow 2^{\text{Var}_c}$ is defined by

  \[
  \begin{align*}
  \text{kill}_{LV}([\text{skip}]) & := \emptyset \\
  \text{kill}_{LV}([x := a]) & := \{x\} \\
  \text{kill}_{LV}([b]) & := \emptyset
  \end{align*}
  \]

- Every reading access generates a live variable
- Formally: $\text{gen}_{LV} : \text{Blk}_c \rightarrow 2^{\text{Var}_c}$ is defined by

  \[
  \begin{align*}
  \text{gen}_{LV}([\text{skip}]) & := \emptyset \\
  \text{gen}_{LV}([x := a]) & := \text{Var}_a \\
  \text{gen}_{LV}([b]) & := \text{Var}_b
  \end{align*}
  \]
Another Example: Live Variables Analysis

Formalizing Live Variables Analysis II

Example 2.11 \((\text{kill}_L V/\text{gen}_L V \text{ functions})\)

\[c = [x := 2];\]
\[ [y := 4];\]
\[ [x := 1];\]
\[ \text{if } [y > 0] \text{ then}\]
\[ [z := x];\]
\[ \text{else}\]
\[ [z := y \times y];\]
\[ \text{end};\]
\[ [x := z];\]
Another Example: Live Variables Analysis

Formalizing Live Variables Analysis II

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

\[
c = [x := 2]_1; \\
[y := 4]_2; \\
x := 1]_3; \\
\text{if } [y > 0]_4 \text{ then} \\
\quad [z := x]_5 \\
\text{else} \\
\quad [z := y*y]_6 \\
\text{end}; \\
x := z]_7
\]

\[\bullet \ Var_c = \{x, y, z\}\]
Another Example: Live Variables Analysis

Formalizing Live Variables Analysis II

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

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

\[
\begin{align*}
\text{Var}_c &= \{x, y, z\} \\
\text{I} \in \text{Lab}_c \text{ kill}_{\text{LV}}(B') \text{ gen}_{\text{LV}}(B')
\end{align*}
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{x}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>2</td>
<td>{y}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>3</td>
<td>{x}</td>
<td>\emptyset</td>
</tr>
<tr>
<td>4</td>
<td>\emptyset</td>
<td>{y}</td>
</tr>
<tr>
<td>5</td>
<td>{z}</td>
<td>{x}</td>
</tr>
<tr>
<td>6</td>
<td>{z}</td>
<td>{y}</td>
</tr>
<tr>
<td>7</td>
<td>{x}</td>
<td>{z}</td>
</tr>
</tbody>
</table>
Another Example: Live Variables Analysis

The Equation System I

- For each \( l \in Lab_c \), \( LV_l \subseteq Var_c \) represents the set of live variables at the exit of block \( B^l \)

Formally, for a program \( c \in Cmd \) with isolated exits:

\[
LV_l = \begin{cases} 
Var_c & \text{if } l \in \text{final}(c) \\
\bigcup \{ \phi_{l, l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise}
\end{cases}
\]

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

\[
\phi_{l, l'}(V) = (V \setminus \text{kill}(LV(B^{l'}))) \cup \text{gen}(LV(B^{l'}))
\]

Characterization of analysis:
- flow-sensitive: results depending on order of assignments
- backward: starts in final \( (c) \) and proceeds upwards

Later: solution not necessarily unique \( \Rightarrow \) choose least one
Another Example: Live Variables Analysis

The Equation System I

- For each \( l \in Lab_c, LV_l \subseteq Var_c \) represents the set of live variables at the exit of block \( B^l \)
- Formally, for a program \( c \in Cmd \) with isolated exits:

\[
LV_l = \begin{cases} 
Var_c & \text{if } l \in \text{final}(c) \\
\bigcup \{ \varphi_{l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise}
\end{cases}
\]

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

\[
\varphi_{l'}(V) := (V \setminus \text{kill}_{LV}(B^{l'})) \cup \text{gen}_{LV}(B^{l'})
\]
Another Example: Live Variables Analysis

The Equation System I

- For each $l \in \text{Lab}_c$, $LV_l \subseteq \text{Var}_c$ represents the set of live variables at the exit of block $B^l$.
- Formally, for a program $c \in \text{Cmd}$ with isolated exits:
  \[
  LV_l = \begin{cases} 
  \text{Var}_c & \text{if } l \in \text{final}(c) \\
  \bigcup \{ \varphi_{l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise}
  \end{cases}
  \]
  where $\varphi_{l'} : 2^{\text{Var}_c} \rightarrow 2^{\text{Var}_c}$ denotes the transfer function of block $B^{l'}$, given by
  \[
  \varphi_{l'}(V) := (V \setminus \text{kill}_{LV}(B^{l'})) \cup \text{gen}_{LV}(B^{l'})
  \]
- Characterization of analysis:
  - flow-sensitive: results depending on order of assignments
  - backward: starts in $\text{final}(c)$ and proceeds upwards
  - may: $\bigcup$ in equations for $LV_l$
Another Example: Live Variables Analysis

The Equation System I

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

- Characterization of analysis:
  - **flow-sensitive**: results depending on order of assignments
  - **backward**: starts in \( \text{final}(c) \) and proceeds upwards
    - **may**: \( \bigcup \) in equations for \( LV_l \)
- Later: solution not necessarily unique
  \( \implies \) choose least one
Another Example: Live Variables Analysis

The Equation System II

Reminder: \( LV_l = \begin{cases} \text{Var}_c & \text{if } l \in \text{final}(c) \\ \bigcup \{ \varphi_{l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise} \end{cases} \)

\( \varphi_{l'}(V) = (V \setminus \text{kill}_{LV}(B_{l'})) \cup \text{gen}_{LV}(B_{l''}) \)
Another Example: Live Variables Analysis

The Equation System II

Reminder:
\[
LV_l = \begin{cases} 
\text{Var}_c & \text{if } l \in \text{final}(c) \\
\bigcup \{ \varphi_{l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise}
\end{cases}
\]

\[
\varphi_{l'}(V) = (V \setminus \text{kill}_{LV}(B_{l'})) \cup \text{gen}_{LV}(B_{l''})
\]

Example 2.12 (LV equation system)

\[
c = \begin{bmatrix}
x := 2 \\
y := 4 \\
x := 1 \\
\text{if } [y > 0] \\
\text{then} \\
\text{[z := x]} \\
\text{else} \\
\text{[z := y*y]} \\
\text{end;} \\
\text{[x := z]}
\end{bmatrix}
\]
Another Example: Live Variables Analysis

The Equation System II

Reminder:
\[
LV_l = \begin{cases} 
\mathbf{Var}_c & \text{if } l \in \text{final}(c) \\
\bigcup \{ \varphi_{l'}(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} & \text{otherwise}
\end{cases}
\]
\[
\varphi_{l'}(V) = (V \setminus \text{kill}_{LV}(B')) \cup \text{gen}_{LV}(B'')
\]

Example 2.12 (LV equation system)

\[
c = \begin{bmatrix} x := 2 \end{bmatrix}^1; \\
y := 4^2; \\
x := 1^3; \\
\text{if } [y > 0]^4 \text{ then} \\
\quad z := x^5 \\
\text{else} \\
\quad z := y*y^6 \\
\text{end}; \\
x := z^7 \\
l \in Lab_c \text{ kill}_{LV}(B') \text{ gen}_{LV}(B') \\
1 \quad \{x\} \quad \emptyset \\
2 \quad \{y\} \quad \emptyset \\
3 \quad \{x\} \quad \emptyset \\
4 \quad \emptyset \quad \{y\} \\
5 \quad \{z\} \quad \{x\} \\
6 \quad \{z\} \quad \{y\} \\
7 \quad \{x\} \quad \{z\}
\]
### The Equation System II

**Reminder:**

\[
LV_l = \begin{cases} 
\text{Var}_c & \text{if } l \in \text{final}(c) \\
\bigcup \{ \varphi_r(LV_{l'}) \mid (l, l') \in \text{flow}(c) \} \cup \text{gen}_{LV}(B'') & \text{otherwise}
\end{cases}
\]

\[
\varphi_r(V) = (V \setminus \text{kill}_{LV}(B'')) \cup \text{gen}_{LV}(B'')
\]

#### Example 2.12 (LV equation system)

```plaintext
\begin{align*}
c & = \begin{cases} 
x := 2 \quad \text{if } y > 0 \\
\text{else} \\
x := 1 \quad \text{end;}
\end{cases} \\
y & := 4 \\
x & := 1 \\
\text{if } y > 0 & \text{ then} \\
z & := x \\
\text{else} \\
z & := y*y \\
x & := z
\end{align*}
```

<table>
<thead>
<tr>
<th>l ∈ Lab_c (k)ll_{LV}(B')</th>
<th>gen_{LV}(B')</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>y</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>∅</td>
</tr>
<tr>
<td>5</td>
<td>z</td>
</tr>
<tr>
<td>6</td>
<td>z</td>
</tr>
<tr>
<td>7</td>
<td>x</td>
</tr>
</tbody>
</table>

### Equations:

\[
\begin{align*}
LV_1 & = \varphi_2(LV_2) = LV_2 \setminus \{ y \} \\
LV_2 & = \varphi_3(LV_3) = LV_3 \setminus \{ x \} \\
LV_3 & = \varphi_4(LV_4) = LV_4 \cup \{ y \} \\
LV_4 & = \varphi_5(LV_5) \cup \varphi_6(LV_6) \\
& = ((LV_5 \setminus \{ z \}) \cup \{ x \}) \cup ((LV_6 \setminus \{ z \}) \cup \{ y \}) \\
LV_5 & = \varphi_7(LV_7) = (LV_7 \setminus \{ x \}) \cup \{ z \} \\
LV_6 & = \varphi_7(LV_7) = (LV_7 \setminus \{ x \}) \cup \{ z \} \\
LV_7 & = \{ x, y, z \}
\end{align*}
\]
Another Example: Live Variables Analysis

The Equation System II

Reminder:

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

\[ \varphi_{l'}(V) = (V \setminus \text{kill}_{\text{LV}}(B')) \cup \text{gen}_{\text{LV}}(B') \]

Example 2.12 (LV equation system)

\[ c = \begin{cases} [x := 2] & 1 \\ [y := 4] & 2 \\ [x := 1] & 3 \\ \text{if } [y > 0] & 4 \\ \quad [z := x] & 5 \\ \text{else} & 6 \\ \quad [z := y*y] & 7 \\ \end{cases} \]

\[ l \in \text{Lab}_c \begin{array}{c|c|c} \text{kill}_{\text{LV}}(B') \text{ gen}_{\text{LV}}(B') \hline 1 & x & \emptyset \\ 2 & \{y\} & \emptyset \\ 3 & \{x\} & \emptyset \\ 4 & \emptyset & \{y\} \\ 5 & \{z\} & \{x\} \\ 6 & \{z\} & \{y\} \\ 7 & \{x\} & \{z\} \end{array} \]

Equations:

\[ \text{LV}_1 = \varphi_2(\text{LV}_2) = \text{LV}_2 \setminus \{y\} \]
\[ \text{LV}_2 = \varphi_3(\text{LV}_3) = \text{LV}_3 \setminus \{x\} \]
\[ \text{LV}_3 = \varphi_4(\text{LV}_4) = \text{LV}_4 \cup \{y\} \]
\[ \text{LV}_4 = \varphi_5(\text{LV}_5) \cup \varphi_6(\text{LV}_6) \]
\[ \quad = (\text{LV}_5 \setminus \{z\}) \cup \{x\} \cup (\text{LV}_6 \setminus \{z\}) \cup \{y\} \]
\[ \text{LV}_5 = \varphi_7(\text{LV}_7) = (\text{LV}_7 \setminus \{x\}) \cup \{z\} \]
\[ \text{LV}_6 = \varphi_7(\text{LV}_7) = (\text{LV}_7 \setminus \{x\}) \cup \{z\} \]
\[ \text{LV}_7 = \{x, y, z\} \]

(Unique) solution:

\[ \text{LV}_1 = \emptyset \]
\[ \text{LV}_2 = \{y\} \]
\[ \text{LV}_3 = \{x, y\} \]
\[ \text{LV}_4 = \{x, y\} \]
\[ \text{LV}_5 = \{y, z\} \]
\[ \text{LV}_6 = \{y, z\} \]
\[ \text{LV}_7 = \{x, y, z\} \]