Semantics and Verification of Software

Lecture 19: Dataflow Analysis II (Live Variables Analysis)

Thomas Noll

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw10/

Summer Semester 2010

Outline

1. Repetition: Dataflow Analysis
2. Another Example: Live Variables Analysis
3. Heading for a Dataflow Analysis Framework

Labelled Programs

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

  These constructs will be called blocks.
- Assume set of labels L with meta variable $l \in L$ (usually $L = \mathbb{N}$)

Definition (Labelled WHILE programs)

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

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

Here all labels in a statement $c \in Cmd$ are assumed to be distinct.

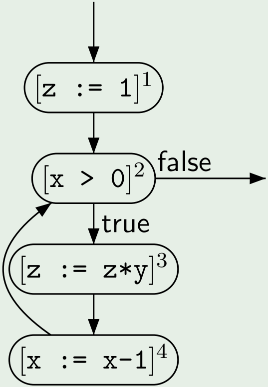
Representing Control Flow

Example

Visualization by (control) flow graph:

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

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



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.

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 to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

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 to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example (Available Expressions Analysis)

```
[x := a+b]^1;
[y := a*b]^2;
while [y > a+b]^3 do
  [a := a+1]^4;
  [x := a+b]^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 to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example (Available Expressions Analysis)

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

- 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 to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example (Available Expressions Analysis)

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

- `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 to avoid recomputations of expressions
- only interesting for non-trivial (i.e., complex) arithmetic expressions

Example (Available Expressions Analysis)

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

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

The Equation System

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 (AE equation system)

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

$l \in L_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
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	$\emptyset$	$\{\texttt{a+b}\}$

Equations:

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

Solution:

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

Outline

Goal of the 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.

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

Goal of the 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)

Goal of the 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

An Example

Example 19.1 (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;
[x := z]^7
```

An Example

Example 19.1 (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
  $[\texttt{z := y*y}]^6$;
$[\texttt{x := z}]^7$

- x not live at exit from label 1

An Example

Example 19.1 (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;
[x := z]^7
```

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

An Example

Example 19.1 (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;
[x := z]^7
```

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

An Example

Example 19.1 (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;
[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

An Example

Example 19.1 (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;
[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: $\mathsf{kill}_{\mathsf{LV}} : Block_c \to 2^{Var_c}$ is defined by

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

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: $\mathsf{kill}_{\mathsf{LV}} : \mathit{Block}_c \to 2^{\mathit{Var}_c}$ is defined by

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

- 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: $\mathsf{kill_{LV}} : Block_c \to 2^{Var_c}$ is defined by
$$\begin{aligned} \mathsf{kill_{LV}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{kill_{LV}}([x \texttt{ := } a]^l) &:= \{x\} \\ \mathsf{kill_{LV}}([b]^l) &:= \emptyset \end{aligned}$$
- Every reading access generates a live variable
- Formally: $\mathsf{gen_{LV}} : Block_c \to 2^{Var_c}$ is defined by
$$\begin{aligned} \mathsf{gen_{LV}}([\mathtt{skip}]^l) &:= \emptyset \\ \mathsf{gen_{LV}}([x \texttt{ := } a]^l) &:= FV(a) \\ \mathsf{gen_{LV}}([b]^l) &:= FV(b) \end{aligned}$$

Formalizing Live Variables Analysis II

Example 19.2 ($\mathsf{kill}_{\mathsf{LV}}/\mathsf{gen}_{\mathsf{LV}}$ functions)

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

Formalizing Live Variables Analysis II

Example 19.2 ($\mathsf{kill}_{\mathsf{LV}}/\mathsf{gen}_{\mathsf{LV}}$ functions)

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

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

Example 19.2 ($\mathsf{kill}_{\mathsf{LV}}/\mathsf{gen}_{\mathsf{LV}}$ functions)

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

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

$l \in L_c$	$\mathsf{kill}_{\mathsf{LV}}(B^l)$	$\mathsf{gen}_{\mathsf{LV}}(B^l)$
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	$\{\mathtt{x}\}$	$\{\mathtt{z}\}$

The Equation System I

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

The Equation System I

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

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

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

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

The Equation System I

- For each $l \in L_c$, $\mathsf{LV}_l \subseteq Var_c$ 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:

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

  may: $\bigcup$ in equation for LV_l

  flow-sensitive: results depending on order of assignments

The Equation System I

- For each $l \in L_c$, $\mathsf{LV}_l \subseteq \mathit{Var}_c$ represents the set of live variables at the exit of block B^l
- Formally, for a program $c \in \mathit{Cmd}$ with isolated exits:
$$\mathsf{LV}_l = \begin{cases} \mathit{Var}_c & \text{if } l \in \mathsf{final}(c) \\ \bigcup\{\varphi_{l'}(\mathsf{LV}_{l'}) \mid (l,l') \in \mathsf{flow}(c)\} & \text{otherwise} \end{cases}$$
where $\varphi_{l'} : 2^{\mathit{Var}_c} \to 2^{\mathit{Var}_c}$ 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:
  - backward: starts in $\mathsf{final}(c)$ and proceeds upwards
  - may: $\bigcup$ in equation for LV_l
  - flow-sensitive: results depending on order of assignments
- Later: solution not necessarily unique
  $\Longrightarrow$ choose least one

The Equation System II

Reminder:

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

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

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 19.3 (LV equation system)

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

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 19.3 (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;
[x := z]^7

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

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 19.3 (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;
    [x := z]^7
```

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

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

The Equation System II

Reminder:

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

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

Example 19.3 (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;
[x := z]^7
```

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

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

Solution:

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

Outline

Similarities between Analysis Problems

- **Observation:** the analyses presented so far have some similarities

Similarities between Analysis Problems

- **Observation:** the analyses presented so far have some similarities

$\Longrightarrow$ Look for underlying framework

Similarities between Analysis Problems

- **Observation:** the analyses presented so far have some similarities

$\Longrightarrow$ Look for underlying framework

- **Advantage:** possibility for designing (efficient) generic algorithms for solving dataflow equations

Similarities between Analysis Problems

- **Observation:** the analyses presented so far have some similarities

$\Longrightarrow$ Look for underlying framework

- **Advantage:** possibility for designing (efficient) generic algorithms for solving dataflow equations
- **Overall pattern:** for $c \in Cmd$ and $l \in L_c$, the analysis information (AI) is described by equations of the form

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

where

- the set of extremal labels, E, is $\{\mathsf{init}(c)\}$ or $\mathsf{final}(c)$
- ι specifies the extremal analysis information
- the combination operator, $\bigsqcup$, is $\bigcap$ or $\bigcup$
- $\varphi_{l'}$ denotes the transfer function of block $B^{l'}$
- the flow relation F is $\mathsf{flow}(c)$ or $\mathsf{flow}^R(c)$ $(:= \{(l', l) \mid (l, l') \in \mathsf{flow}(c)\})$

Characterization of Analyses

- **Direction of information flow:**
  - forward:
    - $F = \mathsf{flow}(c)$
    - AI_l concerns entry of B^l
    - c has isolated entry
  - backward:
    - $F = \mathsf{flow}^R(c)$
    - AI_l concerns exit of B^l
    - c has isolated exits

Characterization of Analyses

- **Direction of information flow:**
  - forward:
    - $F = \mathsf{flow}(c)$
    - AI_l concerns entry of B^l
    - c has isolated entry
  - backward:
    - $F = \mathsf{flow}^R(c)$
    - AI_l concerns exit of B^l
    - c has isolated exits
- **Quantification over paths:**
  - may:
    - $\bigsqcup = \bigcup$
    - property satisfied by some path
    - interested in least solution (later)
  - must:
    - $\bigsqcup = \bigcap$
    - property satisfied by all paths
    - interested in greatest solution (later)

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Introduce partial order for comparing analysis results

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Introduce partial order for comparing analysis results
2. Establish least upper bound as combination operator

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Introduce partial order for comparing analysis results
2. Establish least upper bound as combination operator
3. Ensure monotonicity of transfer functions

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Introduce partial order for comparing analysis results
2. Establish least upper bound as combination operator
3. Ensure monotonicity of transfer functions
4. Guarantee termination of fixpoint iteration (and continuity of functional) by ascending chain condition

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Introduce partial order for comparing analysis results
2. Establish least upper bound as combination operator
3. Ensure monotonicity of transfer functions
4. Guarantee termination of fixpoint iteration (and continuity of functional) by ascending chain condition
5. Optimize fixpoint iteration by worklist algorithm