Compiler Construction

Lecture 26: Code Optimization

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/cc10/

Winter semester 2010/11

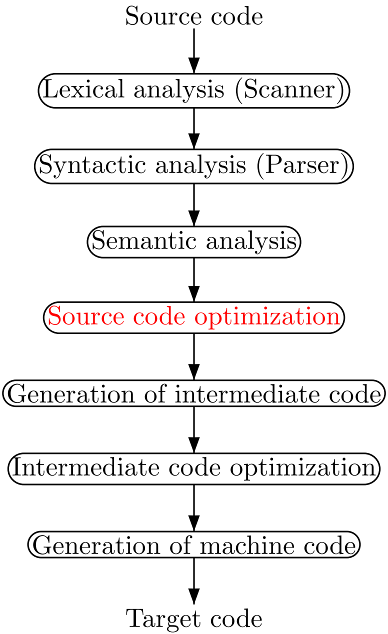
Online Evaluation of CS Curricula

`http://www.campus.rwth-aachen.de/evasys/index.php?mca=online/index/`

- In German
- Losung: `autobahn`
- Advisory service, preparatory CS course
- Conditions of study
- Applied subject
- ...

Outline

Conceptual Structure of a Compiler



Code Optimization

Goal: Make generated code faster and/or more compact

Code Optimization

Goal: Make generated code faster and/or more compact

Common procedure:

- Gather information about program by performing some kind of analysis
- Exploit information to optimize code

Code Optimization

Goal: Make generated code faster and/or more compact

Common procedure:

- Gather information about program by performing some kind of analysis
- Exploit information to optimize code

Here: dataflow analysis

$\Longrightarrow$ attach properties to program statements
that hold every time when statement is executed

Outline

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:

  direction of flow: forward vs. backward analyses

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

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

  procedures: interprocedural vs. intraprocedural analyses

  distinction of procedure calls: context-sensitive vs. context-insensitive analyses

Labeled Programs

- Goal: localization of analysis information

Labeled Programs

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

  These constructs will be called blocks (denotation: *Blk*).

Labeled Programs

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

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

Labeled Programs

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

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

Definition 26.1 (Labeled WHILE programs)

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

$$\begin{aligned}
A &::= z \mid I \mid A_1 \texttt{ + } A_2 \in AExp \\
B &::= A_1 \texttt{ < } A_2 \mid \texttt{not } B \mid B_1 \texttt{ and } B_2 \in BExp \\
C &::= [I \texttt{ := } A]^l \mid C_1 \texttt{;} C_2 \mid \\
&\qquad \texttt{if } [B]^l \texttt{ then } C_1 \texttt{ else } C_2 \mid \texttt{while } [B]^l \texttt{ do } C \in Cmd
\end{aligned}$$

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

A WHILE Program

Example 26.2

```
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;
```

A WHILE Program with Labels

Example 26.2

```
[x := 6]1;
[y := 7]2;
[z := 0]3;
while [x > 0]4 do
  [x := x - 1]5;
  [v := y]6;
  while [v > 0]7 do
    [v := v - 1]8;
    [z := z + 1]9
```

Representing Control Flow I

- Every (labeled) statement has a single entry (given by the initial label) and generally multiple exits (given by the final labels)
- Labels are connected via control-flow edges

Representing Control Flow I

- Every (labeled) statement has a single entry (given by the initial label) and generally multiple exits (given by the final labels)
- Labels are connected via control-flow edges
- Formally:
  - initial label $\mathrm{init} : Cmd \to Lab$
  - final labels $\mathrm{final} : Cmd \to 2^{Lab}$
  - (control) flow relation $\mathrm{flow}(C) \subseteq Lab \times Lab$

Representing Control Flow II

Example 26.3

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

Representing Control Flow II

Example 26.3

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

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

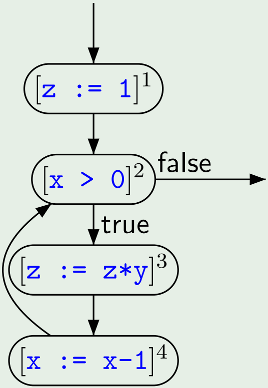
Representing Control Flow II

Example 26.3

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

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

Visualization by flow graph:



Outline

Goal of the 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 the 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., composite) arithmetic expressions

Goal of the 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., composite) arithmetic expressions

Example 26.4 (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 the 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., composite) arithmetic expressions

Example 26.4 (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

Goal of the 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., composite) arithmetic expressions

Example 26.4 (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 the 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., composite) arithmetic expressions

Example 26.4 (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
- possible optimization:
  `while [y > x]^3 do`

Formalizing Available Expressions Analysis I

- Given $C \in Cmd$, $Lab_C/Blk_C/AExp_C$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in C, respectively
- Given $A \in AExp_C$, $\mathrm{Var}(A)$ denotes the set of all variable identifiers occurring in A

Formalizing Available Expressions Analysis I

- Given $C \in Cmd$, $Lab_C / Blk_C / AExp_C$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in C, respectively
- Given $A \in AExp_C$, $\text{Var}(A)$ denotes the set of all variable identifiers occurring in A
- An expression A is killed in a block β if any of the variables in A is modified in β
- Formally: $\text{kill}_{AE} : Blk_C \to 2^{AExp_C}$ is defined by

$$\begin{aligned} \text{kill}_{AE}([I \texttt{ := } A]^l) &:= \{A' \in AExp_C \mid I \in \text{Var}(A')\} \\ \text{kill}_{AE}([B]^l) &:= \emptyset \end{aligned}$$

Formalizing Available Expressions Analysis I

- Given $C \in Cmd$, $Lab_C / Blk_C / AExp_C$ denote the sets of all labels/blocks/complex arithmetic expressions occurring in C, respectively
- Given $A \in AExp_C$, $\mathrm{Var}(A)$ denotes the set of all variable identifiers occurring in A
- An expression A is killed in a block β if any of the variables in A is modified in β
- Formally: $\mathrm{kill}_{AE} : Blk_C \to 2^{AExp_C}$ is defined by
$$\mathrm{kill}_{AE}([I := A]^l) := \{A' \in AExp_C \mid I \in \mathrm{Var}(A')\}$$
$$\mathrm{kill}_{AE}([B]^l) := \emptyset$$
- An expression A is generated in a block β if it is evaluated in and none of its variables are modified by β
- Formally: $\mathrm{gen}_{AE} : Blk_C \to 2^{AExp_C}$ is defined by
$$\mathrm{gen}_{AE}([I := A]^l) := \begin{cases} \{A\} & \text{if } A \in AExp_C, I \notin \mathrm{Var}(A) \\ \emptyset & \text{otherwise} \end{cases}$$
$$\mathrm{gen}_{AE}([B]^l) := AExp_B$$

Formalizing Available Expressions Analysis II

Example 26.5 ($\text{kill}_{AE}/\text{gen}_{AE}$ functions)

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

Example 26.5 ($\text{kill}_{AE}/\text{gen}_{AE}$ functions)

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

- $AExp_C = \{$`a+b`, `a*b`, `a+1`$\}$

Example 26.5 ($\text{kill}_{AE}/\text{gen}_{AE}$ functions)

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

- $AExp_C = \{$`a+b`, `a*b`, `a+1`$\}$
-

Lab_C	$\text{kill}_{AE}(\beta^l)$	$\text{gen}_{AE}(\beta^l)$
1	$\emptyset$	{`a+b`}
2	$\emptyset$	{`a*b`}
3	$\emptyset$	{`a+b`}
4	{`a+b`, `a*b`, `a+1`}	$\emptyset$
5	$\emptyset$	{`a+b`}

The Equation System I

- Analysis itself defined by setting up an equation system

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_C$, $AE_l \subseteq AExp_C$ represents the set of available expressions at the entry of block β^l

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_C$, $AE_l \subseteq AExp_C$ represents the set of available expressions at the entry of block β^l
- Formally:

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

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

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

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_C$, $AE_l \subseteq AExp_C$ represents the set of available expressions at the entry of block β^l
- Formally:

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

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

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

- Characterization of analysis:

  forward: starts in $\text{init}(C)$ and proceeds downwards

  must: $\bigcap$ in equation for AE_l

  flow-sensitive: results depending on order of assignments

The Equation System I

- Analysis itself defined by setting up an equation system
- For each $l \in Lab_C$, $AE_l \subseteq AExp_C$ represents the set of available expressions at the entry of block β^l
- Formally:

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

  where $\varphi_{l'} : 2^{AExp_C} \rightarrow 2^{AExp_C}$ denotes the transfer function of block $\beta^{l'}$, given by

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

- Characterization of analysis:

  forward: starts in $\text{init}(C)$ and proceeds downwards

  must: $\bigcap$ in equation for AE_l

  flow-sensitive: results depending on order of assignments

- In general: solution not necessarily unique

  $\Longrightarrow$ choose greatest one

The Equation System II

Reminder:

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

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

The Equation System II

Reminder:

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

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

Example 26.6 (AE equation system)

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

The Equation System II

Reminder:

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

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

Example 26.6 (AE equation system)

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

$l \in Lab_C$	$\text{kill}_{AE}(\beta^l)$	$\text{gen}_{AE}(\beta^l)$
1	$\emptyset$	{a+b}
2	$\emptyset$	{a*b}
3	$\emptyset$	{a+b}
4	{a+b, a*b, a+1}	$\emptyset$
5	$\emptyset$	{a+b}

The Equation System II

Reminder:

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

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

Example 26.6 (AE equation system)

$C =$

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

$l \in Lab_C$	$\text{kill}_{AE}(\beta^l)$	$\text{gen}_{AE}(\beta^l)$
1	$\emptyset$	{a+b}
2	$\emptyset$	{a*b}
3	$\emptyset$	{a+b}
4	{a+b, a*b, a+1}	$\emptyset$
5	$\emptyset$	{a+b}

Equations:

$$\begin{aligned} AE_1 &= \emptyset \\ AE_2 &= \varphi_1(AE_1) = AE_1 \cup \{\texttt{a+b}\} \\ AE_3 &= \varphi_2(AE_2) \cap \varphi_5(AE_5) \\ &= (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\}) \\ AE_4 &= \varphi_3(AE_3) = AE_3 \cup \{\texttt{a+b}\} \\ AE_5 &= \varphi_4(AE_4) = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\} \end{aligned}$$

The Equation System II

Reminder:

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

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

Example 26.6 (AE equation system)

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

$l \in Lab_C$	$\text{kill}_{AE}(\beta^l)$	$\text{gen}_{AE}(\beta^l)$
1	$\emptyset$	{a+b}
2	$\emptyset$	{a*b}
3	$\emptyset$	{a+b}
4	{a+b, a*b, a+1}	$\emptyset$
5	$\emptyset$	{a+b}

Equations:

$$\begin{aligned} AE_1 &= \emptyset \\ AE_2 &= \varphi_1(AE_1) = AE_1 \cup \{\texttt{a+b}\} \\ AE_3 &= \varphi_2(AE_2) \cap \varphi_5(AE_5) \\ &= (AE_2 \cup \{\texttt{a*b}\}) \cap (AE_5 \cup \{\texttt{a+b}\}) \\ AE_4 &= \varphi_3(AE_3) = AE_3 \cup \{\texttt{a+b}\} \\ AE_5 &= \varphi_4(AE_4) = AE_4 \setminus \{\texttt{a+b}, \texttt{a*b}, \texttt{a+1}\} \end{aligned}$$

Solution:

$$\begin{aligned} AE_1 &= \emptyset \\ AE_2 &= \{\texttt{a+b}\} \\ AE_3 &= \{\texttt{a+b}\} \\ AE_4 &= \{\texttt{a+b}\} \\ AE_5 &= \emptyset \end{aligned}$$

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.

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 of 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 of 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 of 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 26.7 (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 26.7 (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

An Example

Example 26.7 (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 26.7 (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 26.7 (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
- y live at exit from 2
- x live at exit from 3
- z live at exits from 5 and 6

An Example

Example 26.7 (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 $[\texttt{x := 2}]^1$

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

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests do not kill
- Formally: $\text{kill}_{LV} : Blk_C \to 2^{Var_C}$ is defined by
  $$\text{kill}_{LV}([I \texttt{ := } A]^l) := \{I\}$$
  $$\text{kill}_{LV}([B]^l) := \emptyset$$

Formalizing Live Variables Analysis I

- A variable on the left-hand side of an assignment is killed by the assignment; tests do not kill
- Formally: $\text{kill}_{LV} : Blk_C \to 2^{Var_C}$ is defined by
  $$\text{kill}_{LV}([I \texttt{ := } A]^l) := \{I\}$$
  $$\text{kill}_{LV}([B]^l) := \emptyset$$
- 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 do not kill
- Formally: $\mathrm{kill}_{LV} : Blk_C \to 2^{Var_C}$ is defined by
  $$\mathrm{kill}_{LV}([I \texttt{ := } A]^l) := \{I\}$$
  $$\mathrm{kill}_{LV}([B]^l) := \emptyset$$
- Every reading access generates a live variable
- Formally: $\mathrm{gen}_{LV} : Blk_C \to 2^{Var_C}$ is defined by
  $$\mathrm{gen}_{LV}([I \texttt{ := } A]^l) := \mathrm{Var}(A)$$
  $$\mathrm{gen}_{LV}([B]^l) := \mathrm{Var}(B)$$

Example 26.8 ($\mathrm{kill}_{LV}/\mathrm{gen}_{LV}$ functions)

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

Formalizing Live Variables Analysis II

Example 26.8 ($\mathrm{kill}_{LV}/\mathrm{gen}_{LV}$ functions)

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

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

Formalizing Live Variables Analysis II

Example 26.8 ($\text{kill}_{LV}/\text{gen}_{LV}$ functions)

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

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

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

The Equation System I

- For each $l \in Lab_C$, $LV_l \subseteq Var_c$ represents the set of live variables at the exit of block β^l

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 β^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 $\varphi_{l'} : 2^{Var_C} \to 2^{Var_C}$ denotes the transfer function of block $\beta^{l'}$, given by

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

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 β^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 $\varphi_{l'} : 2^{Var_C} \to 2^{Var_C}$ denotes the transfer function of block $\beta^{l'}$, given by

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

- Characterization of analysis:

  backward: starts in 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 $l \in Lab_C$, $LV_l \subseteq Var_c$ represents the set of live variables at the exit of block β^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 $\varphi_{l'} : 2^{Var_C} \to 2^{Var_C}$ denotes the transfer function of block $\beta^{l'}$, given by
$$\varphi_{l'}(V) := (V \setminus \text{kill}_{LV}(\beta^{l'})) \cup \text{gen}_{LV}(\beta^{l'})$$
- Characterization of analysis:
  - backward: starts in final(C) and proceeds upwards
  - may: $\bigcup$ in equation for LV_l
  - flow-sensitive: results depending on order of assignments
- In general: solution not necessarily unique
  $\Longrightarrow$ choose least one

The Equation System II

Reminder:

$$LV_l = \begin{cases} Var_C & \text{if } l \in \text{final}(C) \\ \bigcup\{\varphi_{l'}(LV_{l'}) \mid (l,l') \in \text{flow}(C)\} & \text{otherwise} \end{cases}$$

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

The Equation System II

Reminder:

$$LV_l = \begin{cases} Var_C & \text{if } l \in \text{final}(C) \\ \bigcup\{\varphi_{l'}(LV_{l'}) \mid (l,l') \in \text{flow}(C)\} & \text{otherwise} \end{cases}$$

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

Example 26.9 (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

The Equation System II

Reminder:
$$LV_l = \begin{cases} Var_C & \text{if } l \in \text{final}(C) \\ \bigcup\{\varphi_{l'}(LV_{l'}) \mid (l,l') \in \text{flow}(C)\} & \text{otherwise} \end{cases}$$
$$\varphi_{l'}(V) = (V \setminus \text{kill}_{LV}(\beta^{l'})) \cup \text{gen}_{LV}(\beta^{l'})$$

Example 26.9 (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
```

$l \in Lab_c$	$\text{kill}_{LV}(\beta^l)$	$\text{gen}_{LV}(\beta^l)$
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:

$$LV_l = \begin{cases} Var_C & \text{if } l \in \mathrm{final}(C) \\ \bigcup\{\varphi_{l'}(LV_{l'}) \mid (l,l') \in \mathrm{flow}(C)\} & \text{otherwise} \end{cases}$$

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

Example 26.9 (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
```

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

$$\begin{aligned}
LV_1 &= \varphi_2(LV_2) = LV_2 \setminus \{y\} \\
LV_2 &= \varphi_3(LV_3) = LV_3 \setminus \{x\} \\
LV_3 &= \varphi_4(LV_4) = LV_4 \cup \{y\} \\
LV_4 &= \varphi_5(LV_5) \cup \varphi_6(LV_6) \\
&= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup \\
&\quad ((LV_6 \setminus \{z\}) \cup \{y\}) \\
LV_5 &= \varphi_7(LV_7) = (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_6 &= \varphi_7(LV_7) = (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_7 &= \{x, y, z\}
\end{aligned}$$

The Equation System II

Reminder:

$$LV_l = \begin{cases} Var_C & \text{if } l \in \text{final}(C) \\ \bigcup\{\varphi_{l'}(LV_{l'}) \mid (l,l') \in \text{flow}(C)\} & \text{otherwise} \end{cases}$$

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

Example 26.9 (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
```

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

$$\begin{aligned}
LV_1 &= \varphi_2(LV_2) = LV_2 \setminus \{y\} \\
LV_2 &= \varphi_3(LV_3) = LV_3 \setminus \{x\} \\
LV_3 &= \varphi_4(LV_4) = LV_4 \cup \{y\} \\
LV_4 &= \varphi_5(LV_5) \cup \varphi_6(LV_6) \\
&= ((LV_5 \setminus \{z\}) \cup \{x\}) \cup \\
&\quad ((LV_6 \setminus \{z\}) \cup \{y\}) \\
LV_5 &= \varphi_7(LV_7) = (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_6 &= \varphi_7(LV_7) = (LV_7 \setminus \{x\}) \cup \{z\} \\
LV_7 &= \{x, y, z\}
\end{aligned}$$

Solution:

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

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 $l \in Lab_C$, the analysis information (AI) is described by equations of the form

$$AI_l = \begin{cases} \iota & \text{if } l \in E \\ \bigoplus\{\varphi_{l'}(AI_{l'}) \mid (l', l) \in F\} & \text{otherwise} \end{cases}$$

where

- ι specifies the initial analysis information
- E is $\{\text{init}(C)\}$ or $\text{final}(C)$
- $\bigoplus$ is $\bigcap$ or $\bigcup$
- $\varphi_{l'}$ denotes the transfer function of block $\beta^{l'}$
- F is $\text{flow}(C)$ or $\text{flow}^R(C)$ $(:= \{(l', l) \mid (l, l') \in \text{flow}(C)\})$

Characterization of Analyses

- **Direction of information flow:**
  - forward:
    - $E = \{\text{init}(C)\}$
    - c has isolated entry
    - $F = \text{flow}(C)$
    - AI_l concerns entry of β^l
  - backward:
    - $E = \text{final}(C)$
    - c has isolated exits
    - $F = \text{flow}^R(C)$
    - AI_l concerns exit of β^l

Characterization of Analyses

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

Fixpoint Iteration I

Idea: use fixpoint iteration to solve dataflow equation system

1. For $C \in Cmd$ and $l \in Lab_C$, start with "initial" information AI_l ($AE_l = AExp_C$, $LV_l = \emptyset$)
2. Iteratively evaluate dataflow equations until fixpoint reached

Fixpoint Iteration I

Idea: use fixpoint iteration to solve dataflow equation system

1. For $C \in Cmd$ and $l \in Lab_C$, start with "initial" information AI_l ($AE_l = AExp_C$, $LV_l = \emptyset$)
2. Iteratively evaluate dataflow equations until fixpoint reached

Theoretical foundations:

- Analysis information D forms complete lattice ($D_{AE} = 2^{AExp_C}$, $D_{LV} = 2^{Var_C}$)
  - every subset of D has a least upper/greatest lower bound $\implies$ well-definedness of $\bigoplus$
- ... that satisfies the ascending chain condition
  - $d_1 \overset{\supseteq}{\subseteq} d_2 \overset{\supseteq}{\subseteq} \ldots \implies \exists n : d_n = d_{n+1} = \ldots$
- Combination operator and all transfer functions monotonic
  - $d_1 \overset{\supseteq}{\subseteq} d_2 \implies \varphi(d_1) \overset{\supseteq}{\subseteq} \varphi(d_2)$

$\implies$ Fixpoint effectively computable by iteration

Fixpoint Iteration II

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Fixpoint Iteration II

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Equation system:

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

Fixpoint Iteration II

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Equation system:

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

Fixpoint iteration:

i	1	2	3	4	5
0	$AExp_C$	$AExp_C$	$AExp_C$	$AExp_C$	$AExp_C$

Fixpoint Iteration II

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Equation system:

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

Fixpoint iteration:

i	1	2	3	4	5
0	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$
1	$\emptyset$	$AExp_c$	$AExp_c$	$AExp_c$	$\emptyset$

Fixpoint Iteration II

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Equation system:

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

Fixpoint iteration:

i	1	2	3	4	5
0	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$
1	$\emptyset$	$AExp_c$	$AExp_c$	$AExp_c$	$\emptyset$
2	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$AExp_c$	$\emptyset$

Fixpoint Iteration II

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Equation system:

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

Fixpoint iteration:

i	1	2	3	4	5
0	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$
1	$\emptyset$	$AExp_c$	$AExp_c$	$AExp_c$	$\emptyset$
2	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$AExp_c$	$\emptyset$
3	$\emptyset$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\{\texttt{a+b}\}$	$\emptyset$

Example 26.10 (Available Expressions; cf. Example 26.6)

Program:

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

Equation system:

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

Fixpoint iteration:

i	1	2	3	4	5
0	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$	$AExp_c$
1	$\emptyset$	$AExp_c$	$AExp_c$	$AExp_c$	$\emptyset$
2	$\emptyset$	{a+b}	{a+b}	$AExp_c$	$\emptyset$
3	$\emptyset$	{a+b}	{a+b}	{a+b}	$\emptyset$
4	$\emptyset$	{a+b}	{a+b}	{a+b}	$\emptyset$

Fixpoint Iteration III

Example 26.11 (Live Variables; cf. Example 26.9)

Program:

```
[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
```

Fixpoint Iteration III

Example 26.11 (Live Variables; cf. Example 26.9)

Program:

```
[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
```

Equation system:

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

Fixpoint Iteration III

Example 26.11 (Live Variables; cf. Example 26.9)

Program:

```
[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
```

Equation system:

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

Fixpoint iteration:

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

Fixpoint Iteration III

Example 26.11 (Live Variables; cf. Example 26.9)

Program:

```
[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
```

Equation system:

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

Fixpoint iteration:

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

Fixpoint Iteration III

Example 26.11 (Live Variables; cf. Example 26.9)

Program:

```
[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
```

Equation system:

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

Fixpoint iteration:

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

Fixpoint Iteration III

Example 26.11 (Live Variables; cf. Example 26.9)

Program:

```
[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
```

Equation system:

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

Fixpoint iteration:

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

Outlook

Summer Semester 2011: Static Program Analysis

- More on dataflow analysis
- Constraint-based analysis
- Abstract interpretation
- Pointer analysis