

Semantics and Verification of Software

Lecture 15: Dataflow Analysis II (Available Expressions & Live Variables)

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/svsw08/>

Winter semester 2008/09

- 1 Repetition: Dataflow Analysis
- 2 Live Variables Analysis
- 3 Heading for a Dataflow Analysis Framework

- 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:

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

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

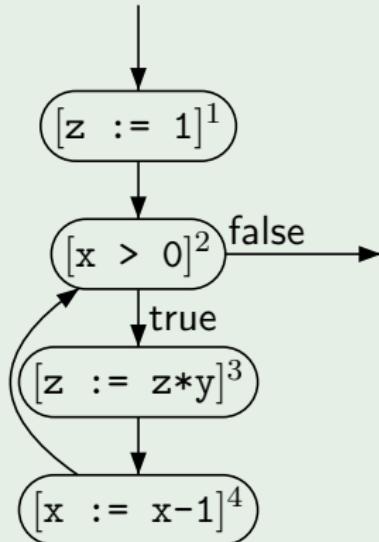
Representing Control Flow

Example

Visualization by **flow graph**:

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

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



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.

- 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
- possible optimization:
 while [y > x]³ do

The Equation System

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

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

Example (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
```

$l \in L_c$	$\text{kill}_{\text{AE}}(B^l)$	$\text{gen}_{\text{AE}}(B^l)$
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:

$$\text{AE}_1 = \emptyset$$

$$\text{AE}_2 = \varphi_1(\text{AE}_1) = \text{AE}_1 \cup \{a+b\}$$

$$\begin{aligned} \text{AE}_3 &= \varphi_2(\text{AE}_2) \cap \varphi_5(\text{AE}_5) \\ &= (\text{AE}_2 \cup \{a*b\}) \cap (\text{AE}_5 \cup \{a+b\}) \end{aligned}$$

$$\text{AE}_4 = \varphi_3(\text{AE}_3) = \text{AE}_3 \cup \{a+b\}$$

$$\text{AE}_5 = \varphi_4(\text{AE}_4) = \text{AE}_4 \setminus \{a+b, a*b, a+1\}$$

Solution: $\text{AE}_1 = \emptyset$

$$\text{AE}_2 = \{a+b\}$$

$$\text{AE}_3 = \{a+b\}$$

$$\text{AE}_4 = \{a+b\}$$

$$\text{AE}_5 = \emptyset$$

- 1 Repetition: Dataflow Analysis
- 2 Live Variables Analysis
- 3 Heading for a Dataflow Analysis Framework

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

Example 15.1 (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 [x := 2]¹

- A variable on the left-hand side of an assignment is **killed** by the assignment; tests and **skip** do not kill
- Formally: $\text{kill}_{\text{LV}} : \text{Block}_c \rightarrow 2^{\text{Var}_c}$ is defined by

$$\text{kill}_{\text{LV}}([\text{skip}]^l) := \emptyset$$

$$\text{kill}_{\text{LV}}([x := a]^l) := \{x\}$$

$$\text{kill}_{\text{LV}}([b]^l) := \emptyset$$

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

$$\text{gen}_{\text{AE}}([\text{skip}]^l) := \emptyset$$

$$\text{gen}_{\text{AE}}([x := a]^l) := \text{FV}(a)$$

$$\text{gen}_{\text{AE}}([b]^l) := \text{FV}(b)$$

Example 15.2 (kill_{LV} / gen_{LV} functions)

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

•	$Var_c = \{x, y, z\}$	
•	$\frac{l \in L_c \text{ kill}_{\text{LV}}(B^l) \text{ gen}_{\text{LV}}(B^l)}{\begin{array}{ccc} 1 & \{x\} & \emptyset \\ 2 & \{y\} & \emptyset \\ 3 & \{x\} & \emptyset \\ 4 & \emptyset & \{y\} \\ 5 & \{z\} & \{x\} \\ 6 & \{z\} & \{y\} \\ 7 & \{x\} & \{z\} \end{array}}$	

The Equation System I

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

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

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

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

- Characterization of analysis:

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

may: \bigcup in equation for LV_l

flow-sensitive: results depending on order of assignments

- Later: solution **not necessarily unique**

\implies choose **least one**

The Equation System II

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

Example 15.3 (LV equation system)

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

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

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

Solution: $\text{LV}_1 = \emptyset$
 $\text{LV}_2 = \{y\}$
 $\text{LV}_3 = \{x, y\}$
 $\text{LV}_4 = \{x, y\}$
 $\text{LV}_5 = \{y, z\}$
 $\text{LV}_6 = \{y, z\}$
 $\text{LV}_7 = \{x, y, z\}$

- 1 Repetition: Dataflow Analysis
- 2 Live Variables Analysis
- 3 Heading for a Dataflow Analysis Framework

Similarities between Analysis Problems

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

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

where

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

- Direction of information flow:

- forward:

- $F = \text{flow}(c)$
 - Al_l concerns entry of B^l
 - c has isolated entry

- backward:

- $F = \text{flow}^R(c)$
 - Al_l concerns exit of B^l
 - c has isolated exits

- Quantification over paths:

- may:

- $\sqcup = \bigcup$
 - property satisfied by some path
 - interested in least solution (later)

- must:

- $\sqcup = \bigcap$
 - property satisfied by all paths
 - interested in greatest solution (later)