Static Program Analysis

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

Winter Semester 2016/17

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ws-1617/spa/

Software Modeling and Verification Chair

RWTH AACHEN UNIVERSITY

For p an arbitrary probability:

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

This program almost surely terminates. In finite expected time.

How does this work?

- ✓ Visit lecture on

Probabilistic Programming

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

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
- 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:

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

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

A WHILE Program with Labels

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
```

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

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

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

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

Representing Control Flow II

Definition 2.4 (Flow relation)

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

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

is defined by

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

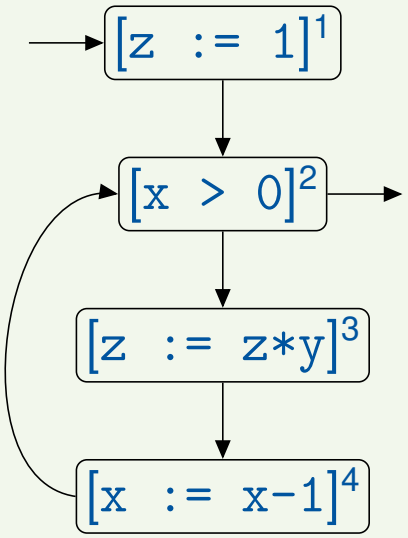
Representing Control Flow III

Example 2.5

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

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

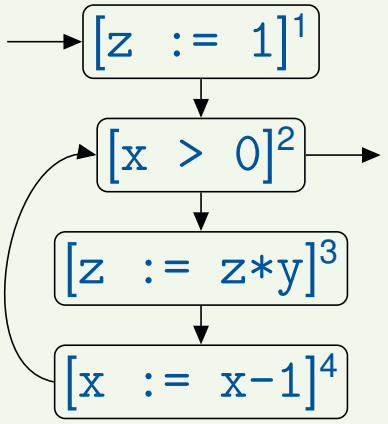
Visualisation by (control) flow graph:



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)



has an isolated entry
but not isolated exits

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

Formalizing Available Expressions Analysis I

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

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

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

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

Formalizing Available Expressions Analysis II

Example 2.8 (kill_{AE}/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
end
```

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

Lab_c	$\text{kill}_{AE}(B^l)$	$\text{gen}_{AE}(B^l)$
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	$\emptyset$	$\{\texttt{a+b}\}$

The Equation System I

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

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

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

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

- Characterization of analysis:

  flow-sensitive: results depending on order of assignments
  forward: starts in $\mathrm{init}(c)$ and proceeds downwards
  must: $\bigcap$ in equations for AE_l

- Later: solution not necessarily unique
  $\Longrightarrow$ choose greatest one

An Example: Available Expressions Analysis

The Equation System II

Reminder:

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

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

Example 2.9 (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
end
```

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

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

(Unique) solution:

$$AE_1 = \emptyset$$
$$AE_2 = \{\texttt{a+b}\}$$
$$AE_3 = \{\texttt{a+b}\}$$
$$AE_4 = \{\texttt{a+b}\}$$
$$AE_5 = \emptyset$$

Another Example: Live Variables Analysis

Goal of Live Variables Analysis

Live Variables Analysis

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

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

Another Example: Live Variables Analysis

An Example

Example 2.10 (Live Variables Analysis)

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

- 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 $[x := 2]^1$

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

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

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

$$\begin{aligned} \text{gen}_{\text{LV}}([\texttt{skip}]^l) &:= \emptyset \\ \text{gen}_{\text{LV}}([x := a]^l) &:= \mathit{Var}_a \\ \text{gen}_{\text{LV}}([b]^l) &:= \mathit{Var}_b \end{aligned}$$

Another Example: Live Variables Analysis

Formalizing Live Variables Analysis II

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

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

- $Var_c = \{x, y, z\}$
- $l \in Lab_c$

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

Another Example: Live Variables Analysis

The Equation System I

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

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

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

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

- Characterization of analysis:

  flow-sensitive: results depending on order of assignments

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

  may: $\bigcup$ in equations for LV_l
- Later: solution not necessarily unique

  $\Longrightarrow$ choose least one

Another Example: Live Variables Analysis

The Equation System II

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

Example 2.12 (LV equation system)

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

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

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

(Unique) solution:
$$\begin{aligned}
\mathrm{LV}_1 &= \emptyset \\
\mathrm{LV}_2 &= \{y\} \\
\mathrm{LV}_3 &= \{x,y\} \\
\mathrm{LV}_4 &= \{x,y\} \\
\mathrm{LV}_5 &= \{y,z\} \\
\mathrm{LV}_6 &= \{y,z\} \\
\mathrm{LV}_7 &= \{x,y,z\}
\end{aligned}$$