Static Program Analysis

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

Summer Semester 2018

Thomas Noll
Software Modeling and Verification Group
RWTH Aachen University

https://moves.rwth-aachen.de/teaching/ss-18/spa/

Exercises

- Assignments made available via L2P Learning Room
- Requires registration via Campus web page
- Please work on assignments in groups of three

Outline of Lecture 2

Preliminaries on Dataflow Analysis

An Example: Available Expressions Analysis

Another Example: Live Variables Analysis

Heading for a Dataflow Analysis Framework

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:

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

Labelled Programs

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

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 $Lab = \mathbb{N}$)

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:

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

- 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

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
```

A WHILE Program with Labels

Example 2.2

```
[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
  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

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

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

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

Representing Control Flow II

Definition 2.4 (Flow relation)

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

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

is defined by

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

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
```

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

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

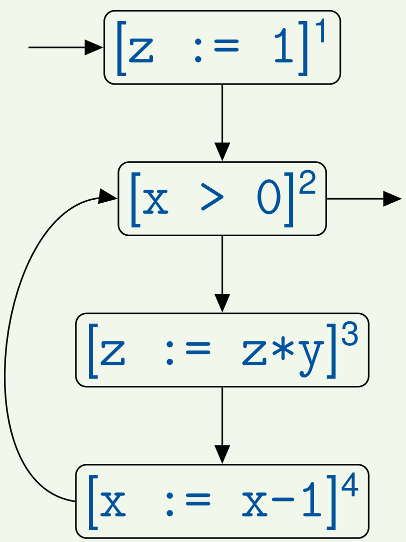
Representing Control Flow III

Example 2.5

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

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

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)

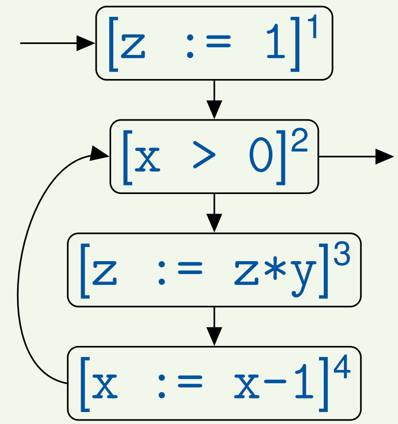
Representing Control Flow IV

- To simplify the presentation we will often assume that the program $c \in Cmd$ under consideration has an isolated entry, meaning that
$$\{l \in Lab \mid (l, \mathrm{init}(c)) \in \mathrm{flow}(c)\} = \emptyset$$
(which is the case when c does not start with a `while` loop)
- Similarly: $c \in Cmd$ has isolated exits if
$$\{l' \in Lab \mid (l, l') \in \mathrm{flow}(c) \text{ for some } l \in \mathrm{final}(c)\} = \emptyset$$
(which is the case when no final label identifies a loop header)

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

Outline of Lecture 2

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

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)

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

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

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

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 optimisation: `while [y > x]^3 do`

Formalising 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

Software Modeling and Verification Chair
RWTH AACHEN UNIVERSITY

Formalising 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: $\mathsf{kill}_{\mathsf{AE}} : Blk_c \to 2^{CExp_c}$ is defined by

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

Formalising 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: $\mathsf{kill}_{\mathsf{AE}} : Blk_c \to 2^{CExp_c}$ is defined by

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

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

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

Formalising Available Expressions Analysis II

Example 2.8 ($\mathsf{kill}_{AE}/\mathsf{gen}_{AE}$ 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
```

Formalising Available Expressions Analysis II

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

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

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

Formalising 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	$\mathsf{kill}_{AE}(B^l)$	$\mathsf{gen}_{AE}(B^l)$
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

An Example: Available Expressions Analysis

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

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

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

An Example: Available Expressions Analysis

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

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

- Characterisation 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

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

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

- Characterisation 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

The Equation System II

Reminder:

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

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

The Equation System II

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 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

An Example: Available Expressions Analysis

The Equation System II

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 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

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

The Equation System II

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 2.9 (AE equation system)

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

$l \in Lab_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
1	$\emptyset$	$\{\mathtt{a+b}\}$
2	$\emptyset$	$\{\mathtt{a*b}\}$
3	$\emptyset$	$\{\mathtt{a+b}\}$
4	$\{\mathtt{a+b}, \mathtt{a*b}, \mathtt{a+1}\}$	$\emptyset$
5	$\emptyset$	$\{\mathtt{a+b}\}$

Equations:

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

The Equation System II

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 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
```

$l \in Lab_c$	$\mathsf{kill}_{\mathsf{AE}}(B^l)$	$\mathsf{gen}_{\mathsf{AE}}(B^l)$
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:
$$\begin{aligned}
\mathsf{AE}_1 &= \emptyset \\
\mathsf{AE}_2 &= \varphi_1(\mathsf{AE}_1) = \mathsf{AE}_1 \cup \{a+b\} \\
\mathsf{AE}_3 &= \varphi_2(\mathsf{AE}_2) \cap \varphi_5(\mathsf{AE}_5) \\
&= (\mathsf{AE}_2 \cup \{a*b\}) \cap (\mathsf{AE}_5 \cup \{a+b\}) \\
\mathsf{AE}_4 &= \varphi_3(\mathsf{AE}_3) = \mathsf{AE}_3 \cup \{a+b\} \\
\mathsf{AE}_5 &= \varphi_4(\mathsf{AE}_4) = \mathsf{AE}_4 \setminus \{a+b, a*b, a+1\}
\end{aligned}$$

(Unique) solution:
$$\begin{aligned}
\mathsf{AE}_1 &= \emptyset \\
\mathsf{AE}_2 &= \{a+b\} \\
\mathsf{AE}_3 &= \{a+b\} \\
\mathsf{AE}_4 &= \{a+b\} \\
\mathsf{AE}_5 &= \emptyset
\end{aligned}$$

Outline of Lecture 2

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

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

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

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 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
```

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

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

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

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

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 optimisation: remove $[\texttt{x := 2}]^1$

Formalising 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: $\mathsf{kill}_{\mathsf{LV}} : Blk_c \rightarrow 2^{Var_c}$ is defined by

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

Formalising 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: $\mathsf{kill}_{\mathsf{LV}} : Blk_c \to 2^{Var_c}$ is defined by

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

- Every reading access generates a live variable
- Formally: $\mathsf{gen}_{\mathsf{LV}} : Blk_c \to 2^{Var_c}$ is defined by

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

Formalising 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
```

Formalising 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
```

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

Formalising 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
```

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

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

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

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

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

Another Example: Live Variables Analysis

The Equation System I

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

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

- Characterisation of analysis:

flow-sensitive: results depending on order of assignments

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

may: $\bigcup$ in equations for LV_l

Another Example: Live Variables Analysis

The Equation System I

- For each $l \in Lab_c$, $\mathrm{LV}_l \subseteq Var_c$ 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

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

- Characterisation 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:

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

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

The Equation System II

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

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

Example 2.12 (LV equation system)

$c =$
```
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
```

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 2.12 (LV equation system)

$c =$
```
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
```

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

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

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
```

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

Equations:

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

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 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
```

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

Equations:

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

(Unique) 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 of Lecture 2

Heading for a Dataflow Analysis Framework

Similarities Between Analysis Problems

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

$\Rightarrow$ Look for underlying framework

Heading for a Dataflow Analysis Framework

Similarities Between Analysis Problems

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

⇒ Look for underlying framework

- **Advantages:**
  - possibility for designing (efficient) generic algorithms for solving dataflow equations
  - enables generic correctness proofs of analyses and algorithms

Similarities Between Analysis Problems

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

$\Rightarrow$ Look for underlying framework

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

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

  where
  - the set of extremal labels, E, is $\{\mathrm{init}(c)\}$ or $\mathrm{final}(c)$
  - 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 $\mathrm{flow}(c)$ or $\mathrm{flow}^R(c)$ $(:= \{(l', l) \mid (l, l') \in \mathrm{flow}(c)\})$

Characterisation of Analyses

Direction of information flow

- **Forward**:
  - $F = \text{flow}(c)$
  - AI_l refers to entry of B^l
  - c has isolated entry
- **Backward**:
  - $F = \text{flow}^R(c)$
  - AI_l refers to exit of B^l
  - c has isolated exits

Characterisation of Analyses

Direction of information flow

- **Forward**:
  - $F = \text{flow}(c)$
  - AI_l refers to entry of B^l
  - c has isolated entry
- **Backward**:
  - $F = \text{flow}^R(c)$
  - AI_l refers to 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)

Heading for a Dataflow Analysis Framework

Solutions as Fixpoints

- **Wanted:** solution of (dataflow) equation system

Solutions as Fixpoints

- **Wanted:** solution of (dataflow) equation system
- **Problem:** recursive dependencies between dataflow variables

Solutions as Fixpoints

- **Wanted:** solution of (dataflow) equation system
- **Problem:** recursive dependencies between dataflow variables
- **Idea:** characterise solution as fixpoint of transformation:

$$(\mathsf{AI}_l = \tau_l)_{l \in Lab_c} \iff \Phi((\mathsf{AI}_l)_{l \in Lab_c}) = (\mathsf{AI}_l)_{l \in Lab_c}$$

where $\Phi\left((\mathsf{AI}_l)_{l \in Lab_c}\right) := (\tau_l)_{l \in Lab_c}$

Solutions as Fixpoints

- **Wanted:** solution of (dataflow) equation system
- **Problem:** recursive dependencies between dataflow variables
- **Idea:** characterise solution as fixpoint of transformation:

$$(\mathsf{AI}_l = \tau_l)_{l \in Lab_c} \iff \Phi((\mathsf{AI}_l)_{l \in Lab_c}) = (\mathsf{AI}_l)_{l \in Lab_c}$$

  where $\Phi\left((\mathsf{AI}_l)_{l \in Lab_c}\right) := (\tau_l)_{l \in Lab_c}$
- **Approach:** approximate fixpoint by iteration

Heading for a Dataflow Analysis Framework

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation

Heading for a Dataflow Analysis Framework

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results

Heading for a Dataflow Analysis Framework

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results
3. Establish least upper bound as combination operator

Heading for a Dataflow Analysis Framework

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results
3. Establish least upper bound as combination operator
4. Ensure monotonicity of transfer functions

Heading for a Dataflow Analysis Framework

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results
3. Establish least upper bound as combination operator
4. Ensure monotonicity of transfer functions
5. Guarantee termination of fixpoint iteration by ascending chain condition

Heading for a Dataflow Analysis Framework

Roadmap

Goal: solve dataflow equation system by fixpoint iteration

1. Characterise solution of equation system as fixpoint of a transformation
2. Introduce partial order for comparing analysis results
3. Establish least upper bound as combination operator
4. Ensure monotonicity of transfer functions
5. Guarantee termination of fixpoint iteration by ascending chain condition
6. Optimise fixpoint iteration by worklist algorithm